Interaction networks for the identification of boosted $H \to b\bar{b}$ decays

Eric A. Moreno, Thong Q. Nguyen, Jean-Roch Vlimant, Olmo Cerri, Harvey B. Newman, Avikar Periwal, and Maria Spiropulu
California Institute of Technology, Pasadena, California 91125, USA

Javier M. Duarte
University of California San Diego, La Jolla, California 92093, USA and Fermi National Accelerator Laboratory, Batavia, Illinois 60510, USA

Maurizio Pierini
European Organization for Nuclear Research, 1211 Geneva 23, Switzerland

(Received 10 December 2019; accepted 13 June 2020; published 28 July 2020)

We develop an algorithm based on an interaction network to identify high-transverse-momentum Higgs bosons decaying to bottom quark-antiquark pairs and distinguish them from ordinary jets that reflect the configurations of quarks and gluons at short distances. The algorithm's inputs are features of the reconstructed charged particles in a jet and the secondary vertices associated with them. Describing the jet shower as a combination of particle-to-particle and particle-to-vertex interactions, the model is trained to learn a jet representation on which the classification problem is optimized. The algorithm is trained on simulated samples of realistic LHC collisions, released by the CMS Collaboration on the CERN Open Data Portal. The interaction network achieves a drastic improvement in the identification performance with respect to state-of-the-art algorithms.

DOI: 10.1103/PhysRevD.102.012010

I. INTRODUCTION

Jets are collimated showers of hadrons that reflect the configurations of quarks and gluons produced at particle colliders. Each shower, consisting of quarks and gluons emitted by the primary particle, results in an approximately cone-shaped spray of hadrons, which are then observed in particle detectors. Jet identification, or *tagging*, algorithms are designed to identify the nature of the primary particle that initiates a shower by studying the collective features of the hadrons inside the jet.

Traditionally, jet tagging was limited to light-flavor quarks (q), gluons (g), or b quarks. At the CERN Large Hadron Collider (LHC), jet tagging becomes a more complex task as new jet topologies are accessible (see Fig. 1). Due to the large center-of-mass energy available in LHC collisions, heavy particles, such as W, Z, and Higgs bosons (H) and top quarks (t), may be produced with large transverse momentum (p_T). These particles can decay to all-quark final states. Due to the large p_T of the original particle, these quarks are produced within a small solid angle. The overlapping showers produced by these quarks may be reconstructed as a single massive jet. As shown in Fig. 1, the presence of b quarks in the jet gives rise to unique experimental signatures. In particular, b hadrons are characterized by a lifetime of approximately 1.5 ps, which results in a detectable displacement between the proton-collision point and the point where the b hadron decays.

The identification of jets from heavy resonances relies on *jet substructure* techniques, designed to quantify the number of clusters of energetic particles, or *prongs*, inside the jet. The study of jet substructure was pioneered in the 1990s and early 2000s [1–4], but interest skyrocketed after its proposed application to reconstruct Higgs bosons when produced in association with a vector boson [5]. Extensive reviews of these techniques are provided in Refs. [6,7]. Additional discrimination is provided by the reconstructed jet mass, usually computed after a jet grooming algorithm. A review of the techniques used to reconstruct jets and their substructure at the LHC experiments can be found in Ref. [8]. The jet mass plays a special role in physics analyses exploiting jet substructure, as described for instance in Ref. [9]. The jet mass distribution is typically used to separate jets from boosted heavy particles, characterized by a peaking distribution, from the smoothly falling background, due to ordinary quark and gluon jets.

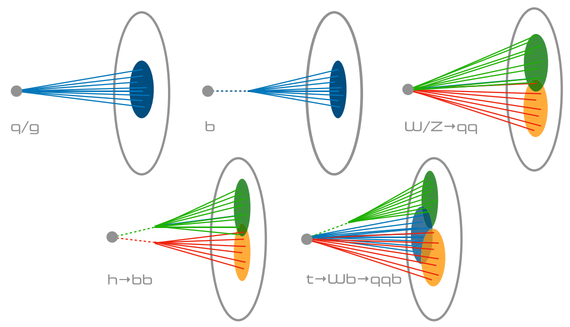


FIG. 1. Pictorial representation of ordinary quark and gluon jets (top left), b jets (top center), and boosted-jet topologies, emerging from high-p_T W and Z bosons (top right), Higgs bosons (bottom left), and top quarks (bottom right) decaying to all-quark final states.

For certain applications, it is desirable to avoid any distortion of the jet mass distribution when applying a jet-tagging selection.

Due to its lifetime, the presence of a b hadron inside of a jet typically results in a reconstructed secondary vertex (SV) that is displaced from the primary vertex (PV). Modern particle detectors are equipped with a vertex detector that can accurately determine SV positions and their separation from the PV, even in a dense environment like a high-p_T jet. This feature is particularly important for tagging a Higgs boson decaying to a bottom quark-antiquark pair ($H \to b\bar{b}$) because all of the relevant jet constituents originate from two displaced vertices.

Recently, several approaches based on deep learning (DL) have been proposed to optimize jet tagging algorithms (see Sec. II), both using expert features with dense layers or raw data representations (e.g., images or lists of particle properties) with more complex architectures. For instance, the LHC collaborations and other researchers have investigated the optimal way to combine substructure, tracking, and vertexing information to enhance the tagging efficiency for high-p_T $H \to b\bar{b}$ decays [10–15]. This is an important task in particle physics because measurements of high-p_T $H \to b\bar{b}$ decays may help resolve the loop-induced and tree-level contributions to the gluon fusion process, providing a complementary approach to study the t Yukawa beyond the $t\bar{t}H$ process [16–19]. These measurements are also sensitive probes for physics beyond the standard model [17,18,20–26]. Finally, improving these measurements is important for measuring the Higgs boson self-coupling through the production of $HH \to b\bar{b}b\bar{b}$ [27–30].

While existing DL approaches have been successfully applied to jet tagging, particle jets involve multiple entities with complex interactions that are not easily encoded as images or lists. Graphs provide a natural representation for such relational information. Traditional machine learning methods use feature engineering and preprocessing to learn from these graphs, which can be time consuming and costly, and may miss important features present in the data. Graph representation learning, including graph convolution networks [31–34] and graph generative models [35,36], leverages DL to learn directly from graph-structured data. In contrast to other DL methods, graph representation learning can (1) handle irregular grids with non-Euclidean geometry [37], (2) encode physics knowledge via graph construction [38], and (3) introduce relational inductive bias into data-driven learning systems [39]. For example, while convolutional neural networks (CNNs) are powerful classifiers that work extremely well for data represented on a grid [40,41], geometric DL algorithms, such as graph neural networks (GNNs) [42,43], are applicable even without an underlying grid structure. Because the data in many scientific domains are not Euclidean, GNNs emerge as a more natural choice.

In this work, we propose to identify $H \to b\bar{b}$ jets with an interaction network (IN), a type of graph network. In Ref. [44], INs were introduced to describe complex physical systems and predict their evolution after a certain amount of time. This was achieved by constructing graph networks to learn the interactions between the physical objects, represented as the nodes of the graph. Just as noted jet substructure variables like $N_2^{\beta=1}$ and $D_2^{\beta=1}$ compute 2-point energy correlation functions between jet constituents to quantify the number of prongs in a jet [45,46], we posit that the ability of INs to learn complex pairwise relationships aids in identifying the patterns present in $H \to b\bar{b}$ decays. Moreover, Ref. [47] showed that the learned features of an IN correlate with known jet substructure variables. It was further demonstrated that the IN architecture outperformed other deep neural networks (DNNs), such as dense, convolutional, and recurrent networks, for a jet-substructure classification task. However, this study was limited because the simulation considered was not fully realistic.

In this paper, we demonstrate that an interaction network with an extended feature representation outperforms state of the art methods for $H \to b\bar{b}$ tagging with GEANT4 -based [48] realistic simulation, while relying on less parameters. In particular, we investigate the use of INs to learn a collective representation of the tracking, vertexing, and substructure properties of the jet and employ this optimized representation to enhance the tagging efficiency. By placing charged particles and secondary vertices on a graph, the network can learn a representation of each particle-to-particle and particle-to-vertex *interaction*, and exploit this information to categorize a given jet as signal ($H \to b\bar{b}$) or background (QCD).

The study is carried out using a sample of fully simulated LHC collision events, released by the CMS Collaboration on the CERN Open Data portal [49]. Previously, many machine learning studies were limited to studies based on generator-level physics with simple detector emulation. The released CMS full-simulation samples allow for a more in depth and realistic study of the efficacy of machine learning methods on high-energy physics experiments. We compare the performance to several different algorithms that we trained with open simulation for $H \to b\bar{b}$ tagging based on the architecture of the deep double-b (DDB) tagger created by the CMS Collaboration [11].

The IN and DDB taggers only rely on information related to charged particles, which (unlike neutral particles) can be traced back to their point of origin: the PV of the high-p_T collision, any SV generated in the collision, or additional PVs originating from simultaneous proton-proton interactions (pileup). This choice makes the algorithm particularly robust against the large pileup contamination expected in future LHC runs since this contamination can be removed via so-called charged hadron subtraction (CHS) [50]. For the IN tagger, we consider an extended representation of each charged particle (secondary vertex), with 22 (12) additional features with respect to the nominal DDB tagger (as discussed in Sec. III). To enable a fair comparison between network architectures, we also report results for an extended variant of the DDB tagger, the deep double-$b+$ (DDB+) tagger, which consumes the same information as the IN tagger.

This paper is structured as follows: we discuss related work in Sec. II. Section III gives a brief description of the datasets used. Sections IV and V describe the IN architecture and the algorithms used to decorrelate its score from the jet mass distribution. Section VI describes our reconstruction and training of the DDB and DDB+ algorithms. Results are presented in Sec. VII and conclusions are given in Sec. VIII.

II. RELATED WORK

The use of DNNs has recently found a great deal of success in particle physics [6,51], especially jet tagging. Driving this innovation are increasingly complex architectures that are tailored to particular domains, including CNNs [52–54], which are well suited to computer vision, and recurrent neural networks (RNNs) [55,56] like long short-term memory units (LSTMs) [57] and gated recurrent units (GRUs) [58], which are appropriate for natural language processing. Several classification algorithms have been studied in the context of jet tagging at the LHC using CNNs [59–63] and physics-inspired DNN models [63–66]. Recurrent and recursive layers have been used to define jet classifiers starting from a list of reconstructed particle momenta [67–70]. Recently, several different approaches, applied to the specific case of t jet identification have been compared [71] on a public t jet tagging dataset [72]. This study found ParticleNet [73], a GNN based on the dynamic graph CNN [34] to be the best performing for that task. In Ref. [47], it was shown that the area under the receiver operating characteristic (ROC) curve (AUC), accuracy, and background rejection at a 30% true positive rate (TPR) of a simple IN architecture trained with the same dataset is within 1%, 0.5%, and 40% of those of ParticleNet, while using 70% fewer parameters. Unsupervised, semisupervised, and weakly supervised methods have also been proposed, mainly to tag t jets or jets coming from postulated new particles [74–84]. Finally, others have also explored the CMS open data and simulation to study jet properties and jet classification algorithms in a realistic setting [85–90].

For the task of identifying $H \to b\bar{b}$ specifically, several machine learning approaches have been applied. In generator-level studies, Ref. [15] uses images, representing both the H candidate jet and the full event, as inputs to a CNN. In conditions more closely resembling real data, the CMS Collaboration created a boosted decision tree based on expert chosen features to identify the presence of two b hadrons within a single anti-k_T [91,92] $R = 0.8$ jet (AK8 jet) [10]. This approach was extended using a deep neural network and additional particle-level and vertex level information, the DDB tagger [11]. Other more generic CMS algorithms, also based on deep neural networks and known as the boosted event shape tagger (BEST) and the DeepAK8 tagger, were created to classify the decays of multiple heavy resonances, including H, Z, W, and t [13]. The ATLAS Collaboration has also designed an algorithm to identify two b hadrons within an anti-k_T $R = 1$ jet using b tagging of track-based subjets [14]. For the task of $H \to b\bar{b}$ identification, the CMS DDB tagger, DeepAK8 algorithm, and the ATLAS tagger achieve similar state-of-the-art performance.

Graph networks [47,71,73,93] and the related particle flow networks [94] have recently been used for other kinds of jet tagging, matching or exceeding the performances of other DL approaches, for event classification [95,96], for charged particle tracking in a silicon detector [97,98], for mitigation of the effects pileup [99], and for particle reconstruction in irregular calorimeters [98,100–102] and the IceCube experiment [96].

While applying GNNs is natural for particle physics data, one issue we confront in this paper is how to deal with heterogeneous hierarchical data, i.e., data composed of different sets of elements with different numbers and types of features. The primary original contributions of this paper are (1) designing an IN with data comprising a heterogeneous graph with two types of graph nodes: particles and SVs), (2) demonstrating that an IN achieves competetive performance on public, realistic simulation for the task of $H \to b\bar{b}$ tagging with fewer trainable parameters in a way that is robust to the effects of pileup, and (3) comparing and evaluating mass decorrelation methods.

III. DATA SAMPLES

The CMS open data and simulation are available from the CERN Open Data Portal [49], including releases of 2010, 2011, and 2012 CMS collision data as well as 2011, 2012, and 2016 CMS simulated data.

Samples of $H \to b\bar{b}$ jets are available from simulated events containing Randall-Sundrum gravitons [103] decaying to two Higgs bosons, which subsequently decay to $b\bar{b}$ pairs. The event generation was done by the CMS Collaboration with MADGRAPH5_aMCATNLO2.2.2 at leading order, with graviton masses ranging between 0.6 and 4.5 TeV. Generation of this process enables better sampling of events with large Higgs boson p_T. The main source of background originates from multijet events. The background dataset was generated with PYTHIA8.205 [104] in different bins of the average p_T of the final-state partons ($\hat{p}_T$). The parton showering and hadronization was performed with PYTHIA8.205 [104], using the CMS underlying event tune CUETP8M1 [105] and the NNPDF 2.3 [106] parton distribution functions. Pileup interactions are modeled by overlaying each simulated event with additional minimum bias collisions, also generated with PYTHIA8.205. The CMS detector response is modeled by GEANT4 [48].

The outcome of the default CMS reconstruction workflow is provided in the open simulation [107]. In particular, particle candidates are reconstructed using the particle-flow (PF) algorithm [108]. Charged particles from pileup interactions are removed using the CHS algorithm. Jets are clustered from the remaining reconstructed particles using the anti-k_T algorithm [91,92] with a jet-size parameter $R = 0.8$. The standard CMS jet energy corrections are applied to the jets. In order to remove soft, wide-angle radiation from the jet, the soft-drop (SD) algorithm [5,109] is applied, with angular exponent $\beta = 0$, soft cutoff threshold $z_{\mathrm{cut}} < 0.1$, and characteristic radius $R_0 = 0.8$ [110]. The SD mass (m_{SD}) is then computed from the four-momenta of the remaining constituents.

A signal $H \to b\bar{b}$ jet is defined as a jet geometrically matched to the generator-level Higgs boson and both b quark daughters. Jets from QCD multijet events are used to define a sample of fake $H \to b\bar{b}$ candidates.

The dataset is reduced by requiring the AK8 jets to have $300 < p_T < 2400$ GeV, $|\eta| < 2.4$, and $40 < m_{\mathrm{SD}} < 200$ GeV. After this reduction, the dataset consists of 3.9 million $H \to b\bar{b}$ jets and 1.9 million inclusive QCD jets. Charged particles are required to have $p_T > 0.95$ GeV and reconstructed secondary vertices (SVs) are associated with the AK8 jet using $\Delta R = \sqrt{\Delta\phi^2 + \Delta\eta^2} < 0.8$. The dataset is divided into blocks of features, referring to different objects. Different blocks are used as input by the models described in the rest of the paper.

The IN uses 30 features related to charged particles (see III in App. C). The IN also uses 14 SV features listed in Table IV. The DDB tagger [11] uses a subset of the above features (8 features for each particle and 2 features for each SV), chosen to minimize the correlation with the jet mass. In addition, the DDB tagger uses 27 high-level features (HLF) listed in Table V and first used in a previous version of the algorithm, described in Ref. [10]. To isolate the effects of the different architecture, the DDB+ tagger uses the same inputs as the IN tagger, while retaining the architecture of the DDB tagger. The charged particles (SVs) are sorted in descending order of the 2D impact parameter significance (2D flight distance significance) and only the first 60 (5) are considered.

IV. THE INTERACTION NETWORK MODEL

The IN is based on two input collections comprising N_p particles, each represented by a feature vector of length P, and N_v vertices, each represented by a feature vector of length S. Although kinematic features of neutral particles could also be taken into account with an additional input graph, we verified that doing so does not significantly improve the performance for this task as shown in Sec. VII. Further, excluding neutral particles has the benefit of improved robustness to pileup. For a single jet, the input consists of an X and a Y matrix, with sizes $P \times N_p$ and $S \times N_v$, respectively. The X matrix contains the input features (columns) of the charged particles (rows), while the Y matrix contains the input features of the SVs.

A particle graph $\mathcal{G}_p$ is constructed by connecting each particle to every other particle through $N_{pp} = N_p(N_p - 1)$ directed edges. Similarly, a particle-vertex graph $\mathcal{G}_{pv}$ is constructed by connecting each vertex to each particle through $N_{pv} = N_p N_v$ directed edges. As described below, we only consider those edges that are received by particles

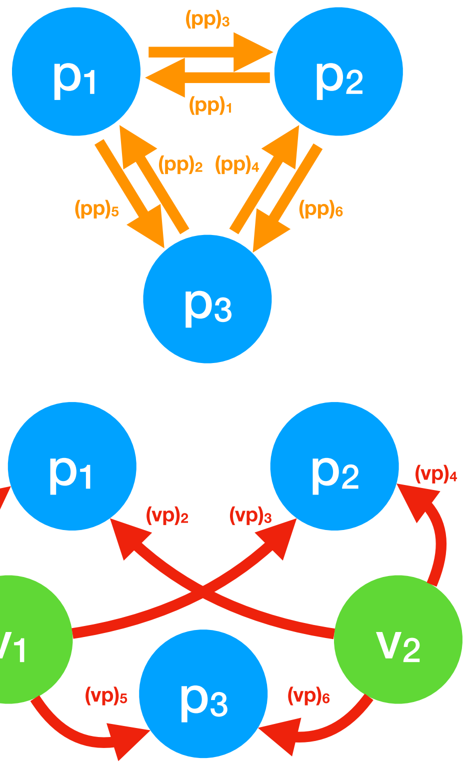


FIG. 2. Two example graphs with 3 particles and 2 vertices and the corresponding edges.

because the final aggregation is performed over the particles. These graphs are pictorially represented in Fig. 2 for the case of three particles and two vertices. As shown in the figure, the graph nodes and edges are arbitrarily enumerated. The result of the graph processing is independent of the labeling order, as described below.

For the graph $\mathcal{G}_p$, a receiving matrix (R_R) and a sending matrix (R_S) are defined, both of size $N_p \times N_{pp}$. The element $(R_R)_{ij}$ is set to 1 when the ith particle receives the jth edge and is 0 otherwise. Similarly, the element $(R_S)_{ij}$ is set to 1 when the ith particle sends the jth edge and is 0 otherwise. For the second graph, the corresponding adjacency matrices R_K (of size $N_p \times N_{vp}$) and R_V (of size $N_v \times N_{vp}$) are defined. In the example of Fig. 2, the R_R, R_S, R_K, and R_V matrices would be written as:

$$R_R = \begin{array}{c|cccccc} & (pp)_1 & (pp)_2 & (pp)_3 & (pp)_4 & (pp)_5 & (pp)_6 \\ \hline p_1 & 1 & 1 & 0 & 0 & 0 & 0 \\ p_2 & 0 & 0 & 1 & 1 & 0 & 0 \\ p_3 & 0 & 0 & 0 & 0 & 1 & 1 \end{array}, \tag{1}$$

$$R_S = \begin{array}{c|cccccc} & (pp)_1 & (pp)_2 & (pp)_3 & (pp)_4 & (pp)_5 & (pp)_6 \\ \hline p_1 & 0 & 0 & 1 & 0 & 1 & 0 \\ p_2 & 1 & 0 & 0 & 0 & 0 & 1 \\ p_3 & 0 & 1 & 0 & 1 & 0 & 0 \end{array}, \tag{2}$$

$$R_K = \begin{array}{c|cccccc} & (vp)_1 & (vp)_2 & (vp)_3 & (vp)_4 & (vp)_5 & (vp)_6 \\ \hline p_1 & 1 & 1 & 0 & 0 & 0 & 0 \\ p_2 & 0 & 0 & 1 & 1 & 0 & 0 \\ p_3 & 0 & 0 & 0 & 0 & 1 & 1 \end{array}, \tag{3}$$

$$R_V = \begin{array}{c|cccccc} & (vp)_1 & (vp)_2 & (vp)_3 & (vp)_4 & (vp)_5 & (vp)_6 \\ \hline v_1 & 1 & 0 & 1 & 0 & 1 & 0 \\ v_2 & 0 & 1 & 0 & 1 & 0 & 1 \end{array}. \tag{4}$$

Each column of an adjacency matrix corresponds to a directional connection from one particle to another, $(pp)_i$, or from a vertex and to a particle, $(vp)_j$. Column entries that are 1 in a given row in the receiving matrix R_R indicate that the corresponding particle receives that connection. Likewise, if a column entry is 1 in a given row in the sending matrix R_S, the corresponding particle is the sender for that connection. Because the fully connected particle graph we consider has no self-connections, i.e., no particle sends and receives the same connection, the rows of R_R and R_S do not share any of the same nonzero column entries. For the R_R and R_V adjacency matrices, we only consider those connections that are sent to particles because the final aggregation is performed over the particles. We tested a version of the IN architecture in which we considered connections that are sent to vertices as well and aggregated separately before being processed by the final network, but found no significant improvement.

The data flow of the IN model is pictorially represented in Fig. 3. The input processing starts by creating the $2P \times N_{pp}$ particle-particle interaction matrix B_{pp} and the $(P+S) \times N_{vp}$ particle-vertex interaction matrix B_{vp} defined as

$$B_{pp} = \begin{pmatrix} X \cdot R_R \\ X \cdot R_S \end{pmatrix}, \tag{5}$$

$$B_{vp} = \begin{pmatrix} X \cdot R_K \\ Y \cdot R_V \end{pmatrix}, \tag{6}$$

where $\cdot$ indicates the ordinary matrix product. Each column of B_{pp} consists of the $2P$ features of the sending and receiving nodes of each particle-particle interaction, while each column of B_{vp} consists of the $P+S$ features of each particle-vertex one.

Processing each column of B_{pp} by the function f_R^{pp}, one builds an internal representation of the particle-particle interaction with a function $f_R^{pp}: \mathbb{R}^{2P} \mapsto \mathbb{R}^{D_E}$, where D_E is the size of the internal representation. This results in an *effect matrix* E_{pp} with dimensions $D_E \times N_{pp}$. We similarly

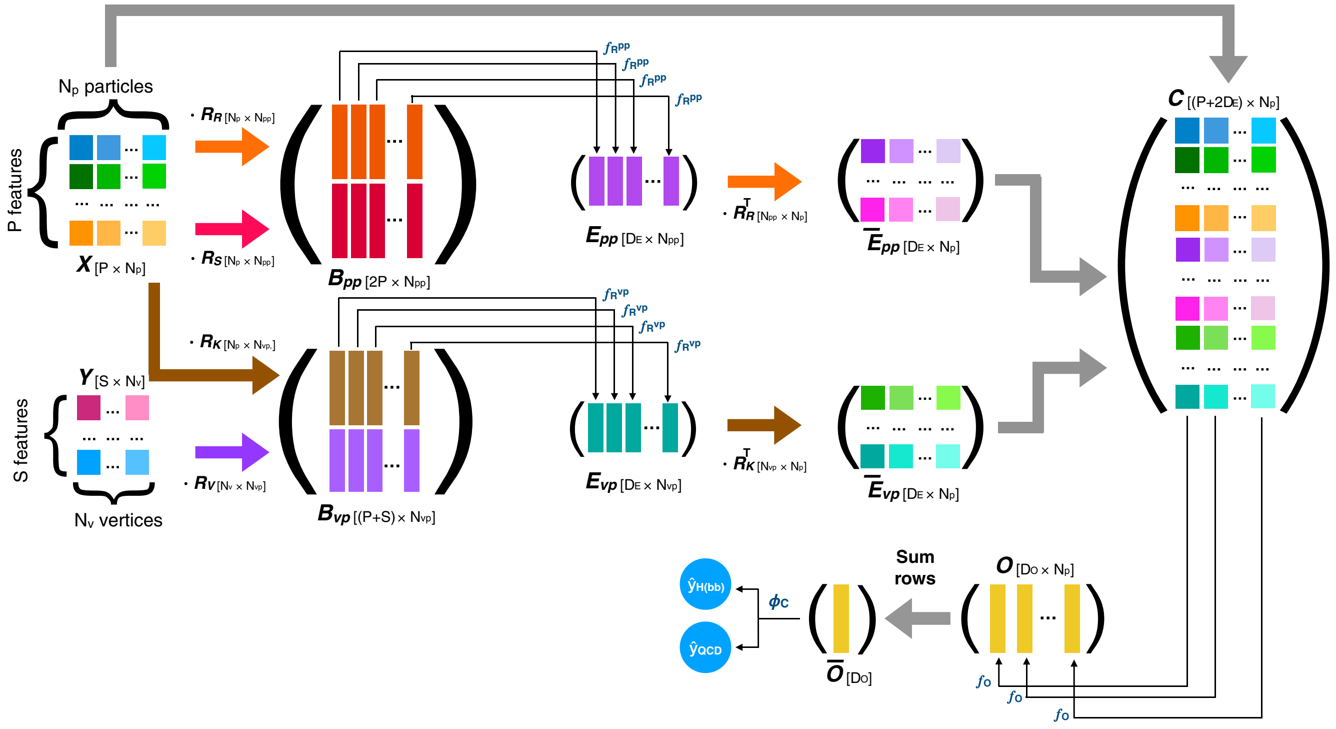


FIG. 3. Illustration of the IN classifier. The particle feature matrix X is multiplied by the receiving and sending matrices R_R and R_S to build the particle-particle interaction feature matrix B_{pp}. Similarly, the particle feature matrix X and the vertex feature matrix Y are multiplied by the adjacency matrices R_K and R_V, respectively, to build the particle-vertex interaction feature matrix B_{vp}. These pairs are then processed by the interaction functions f_R^{pp} and f_R^{vp}, and the postinteraction function f_O, which are expressed as neural networks and learned in the training process. This procedure creates a learned representation of each particle's postinteraction features, given by N_p vectors of size D_O. The N_p vectors are summed, giving D_O features for the entire jet, which is given as input to a classifier ϕ_C, also represented by a neural network. More details on the various steps are given in the text.

build the E_{vp} matrix, with dimensions $D_E \times N_{vp}$, using a function $f_R^{vp}: \mathbb{R}^{P+S} \mapsto \mathbb{R}^{D_E}$.

We then propagate the particle-particle interactions back to the particles receiving them, by building $\bar{E}_{pp} = E_{pp} R_R^\top$ with dimension $D_E \times N_p$. We also build $\bar{E}_{vp} = E_{vp} R_V^\top$ with dimension $D_E \times N_p$, which collects the information of the particle-vertex interactions for each particle and across all of the vertices.

The next step consists of building the C matrix, with dimensions $(P + 2D_E) \times N_p$, by combining the input information for each particle (X) with the learned representation of the particle-particle ($\bar{E}_{pp}$) and particle-vertex ($\bar{E}_{vp}$) interactions:

$$C = \begin{pmatrix} X \\ \bar{E}_{pp} \\ \bar{E}_{vp} \end{pmatrix}. \tag{7}$$

The final aggregator combines the input and interaction information to build the post-interaction representation of the graph, summarized by the matrix O, with dimensions $D_O \times N_p$. The aggregator consists of a function $f_O: \mathbb{R}^{P+2D_E} \mapsto \mathbb{R}^{D_O}$, which computes the elements of the O matrix The elements of the O matrix are computed by a function $f_O: \mathbb{R}^{P+2D_E} \mapsto \mathbb{R}^{D_O}$, which returns the post-interaction representation for each of the input nodes. As is done for f_R^{pp} and f_R^{vp}, f_O is applied to each column of C.

We stress the fact that the by-column processing applied by the f_R^{pp}, f_R^{vp}, and f_O functions and the sum across interactions by defining the $\bar{E}_{pp}$ and $\bar{E}_{vp}$ matrices are essential ingredients to make the outcome of the IN tagger independent of the order used to label the N_p input particles and N_v input vertices. In other words, while the representations of the R_R, R_S, R_K, and R_V matrices depend on the adopted labeling convention, the final representation of each particle does not.

The learned representation of the post-interaction graph, given by the elements of the O matrix, can be used to solve the specific task at hand. Depending on the task, the final function that computes the classifier output may be chosen to preserve the permutation invariance of the input particles and vertices. In this case, we first sum along each row (corresponding to a sum over particles) of O to produce a feature vector $\bar{O}$ with length D_O for the jet as a whole.

This is passed to a function $\phi_C : \mathbb{R}^{D_O} \mapsto \mathbb{R}^N$, which produces the output of the classifier.

The training of the IN is performed with the CMS open simulation with 2016 conditions. The input dataset is split into training, validation, and test samples with percentages of 80%, 10%, and 10%, respectively.

We use PyTorch [111,112] to implement and train the classifier on one NVIDIA GeForce GTX 1080 GPU. We also convert the interaction network into a TensorFlow model, as discussed in Appendix B. The model is implemented with each of f_R^{pp} and f_R^{vp} expressed as a sequence of 3 dense layers of sizes (60,30,20) with a rectified linear unit (ReLU) activation function after each layer. The function f_O is a similar sequence of dense layers of sizes (60,30,24) with ReLU activations. We use up to $N_p = 60$ charged particles and $N_v = 5$ secondary vertices as inputs to the IN tagger. Given the size of these layers, the total number of trainable parameters is 18,144. We train the model using the Adam optimizer [113] with an initial learning rate of 10^{-4} and a batch size of 128 for up to 200 epochs, enforcing early stopping [114] on the validation loss with a patience of 5 epochs. The size of the batch is constrained by the required memory utilization of the GPU. The training takes approximately 25 minutes per epoch on the GPU and stopped after 110 epochs.

For the baseline algorithm, we minimize the categorical cross-entropy loss function for this classification task L_C and let the network exploit all of the discriminating information in the dataset.

To determine the impact of neutral particles, we also train an augmented all-particle IN model, which consumes an additional input set with 10 kinematic features for up to 100 charged or neutral particles, listed in Table VI. This additional input set is processed by the model in a similar way to the SV input set: the set of all particles is fully connected to the set of charged particles. The effect matrix for these interactions is computed by an independent neural network and then appended to an enlarged C matrix, now of size $(P + 3D_E) \times N_p$, before being processed by the network f_O. The remaining steps of the model proceed as described above. The total number of trainable parameters for this model is 24,254.

V. DECORRELATION WITH THE JET MASS

Many possible applications of a jet tagging algorithm would require the final score to be uncorrelated from the jet mass, so that a selection based on the tagger score does not change the jet mass distribution. This is particularly relevant for the background distribution, but is required to some extent also for the signal one. Several techniques exist to deliver a tagger with minimal effects on the jet mass distribution. For taggers based on high-level features, one could remove those features more correlated to the jet mass or divide those correlated features by the jet mass. For taggers based on a more *raw* representation of the jet (as in this case), one could perform an adversarial training [115–119]. One could also reweight or remove background events such that the background m_{SD} distribution is indistinguishable from the signal m_{SD} distribution [120]. Finally, one could also define a mass-dependent threshold based on simulation as in the "designing decorrelated taggers" (DDT) procedure proposed in Ref. [121]. We test and compare all three methods in Appendix A. We found the DDT method to be the most robust and performant deocorrelation procedure. As such, we use it as the nominal decorrelation method in the following results.

A. Designing decorrelated taggers

Following the DDT procedure [121], the tagger threshold for a given false positive rate (FPR) or "working point" is determined as a function of m_{SD}. By creating a m_{SD}-dependent tagger threshold, the background jet m_{SD} distribution for events passing and failing this threshold can be made identical. In practice, this is done by considering the distribution of the network score versus the jet m_{SD} for the training dataset. A quantile regression was used to find the threshold on the network score as a function of m_{SD} distribution that would correspond to a fixed quantile (the chosen 1−FPR value). By construction, this procedure results in near-perfect mass decorrelation.

In this case, a gradient boosted regressor [122,123] with the following parameters was used:

(i) α-quantile of $1 - \mathrm{FPR}$,
(ii) number of estimators of 500,
(iii) minimum number of samples at a leaf node of 50,
(iv) minimum number of samples to split an internal node of 2500,
(v) maximum depth of 5,
(vi) validation set of 20%,
(vii) early stopping with tolerance of 10.

VI. DEEP DOUBLE-B TAGGER MODELS

The DDB tagger is a convolutional and recurrent neural network model developed by CMS [11] to identify boosted $H \to b\bar{b}$ jets. We reconstruct this model based on publicly available information from the CMS Collaboration as follows. The model takes as input 27 HLFs used in Ref. [10], as well as 8 particle-specific features of up to 60 charged particles, and 2 properties of up to 5 SVs associated with the jet (see Appendix C). Each block of inputs is treated as a one-dimensional list, with batch normalization [124] applied directly to the input layers. For each collection of charged particles and SVs, separate 1D convolutional layers [125], with a kernel size of 1, are applied: 2 hidden layers with 32 filters each and ReLU [126] activation. The outputs are then separately fed into two gated recurrent units (GRUs) with 50 output nodes each and ReLU activations. Finally, the GRU outputs are concatenated with the HLFs and processed by a dense layer with 100 nodes and ReLU activation, and another final dense layer with 2 output nodes with softmax activation. Dropout [127] (with a rate of 10%) is used in each layer to prevent overfitting. The nominal DDB tagger model has 40,344 trainable parameters, 32% of which are found in the fully connected layers.

We define a variant of this model, the DDB+ model, which takes as input all 30 features of charged particles and all 14 features of the SVs. In this variant, we do not consider the HLFs. Thus, the final dense layer only receives the GRU outputs from processing the low-level charged particle and SV information. This extended DDB+ tagger algorithm has 38,746 trainable parameters. The number of parameters is less overall because the increase in the size of the convolutional and recurrent layers is compensated by the decrease in the size of the fully connected layers.

We train the DDB and DDB+ models using the CMS open simulation dataset with Keras [128] over up to 200 epochs with an early stopping patience of 5 epochs and a batch size of 4096 using the Adam optimizer with an initial learning rate of 10^{-3}. For both models, one training epoch takes about 3 minutes and training stops after approximately 50 epochs. In this case, the larger batch size is possible due to the smaller GPU memory utilization of the model during training. We find consistent performance for different batch size choices with no evidence of overfitting with larger batch sizes.

In order to decorrelate the tagger output from the jet mass, we use the same DDT procedure described in Sec. V A applied to both the DDB and DDB+ taggers.

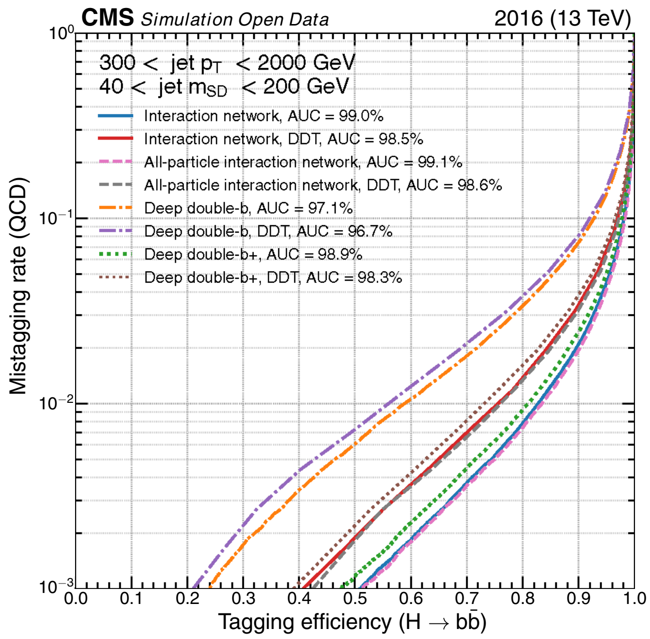
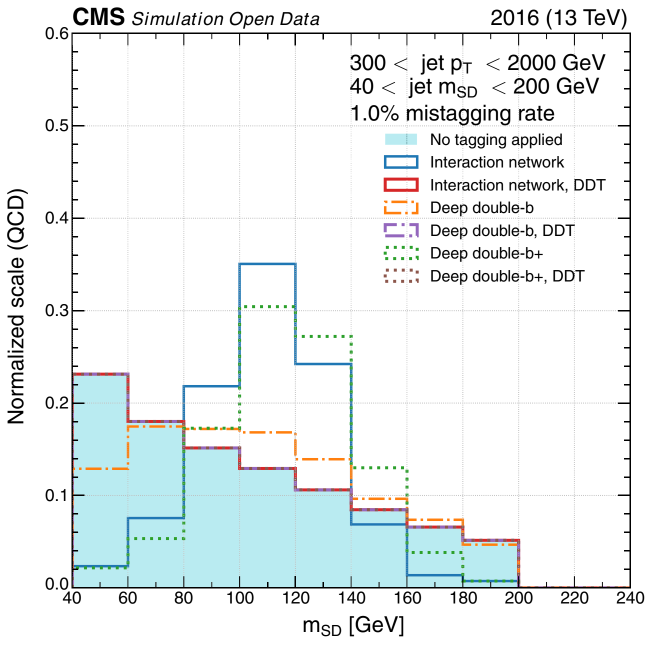


FIG. 4. Performance of the IN, all-particle IN, DDB, and DDB+ algorithms quantified with a ROC curve of FPR (QCD mistagging rate) versus TPR ($H \to b\bar{b}$ tagging efficiency). The performance of each baseline algorithm is compared to that of the algorithms after applying the DDT procedure to decorrelate the tagger score from the jet mass. This decorrelation results in a smaller TPR for a given FPR.



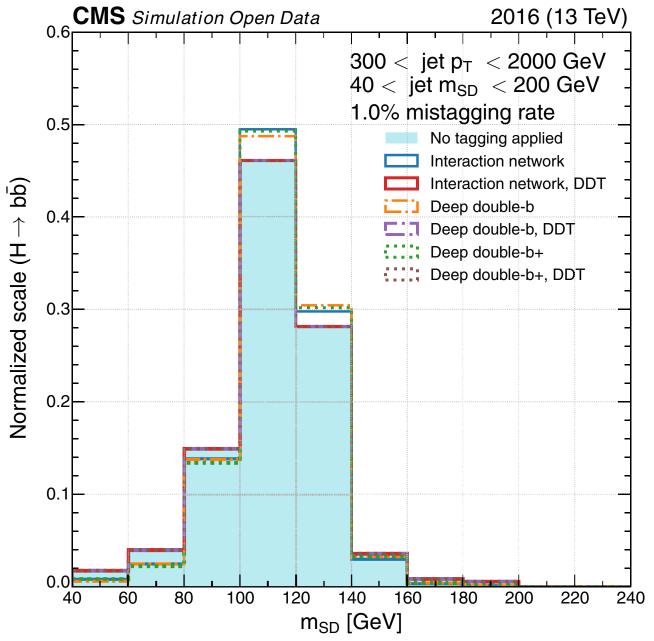


FIG. 5. An illustration of the "sculpting" of the background jet mass distribution (top) and the signal jet mass distribution (bottom) after applying a threshold on the tagger score corresponding to a 1% FPR for several different algorithms. The unmodified interaction network is highly correlated with the jet mass, but after applying the methods described in the text, the correlation is reduced for the background while the peak of the signal distribution is still retained.

TABLE I. Performance metrics of the different baseline and decorrelated models, including accuracy, area under the ROC curve, background rejection at a true positive rate of 30% an 50%, and true positive rate and mass decorrelation metric $1/D_{\mathrm{JS}}$ at a false positive rate of 1%. For the DDT models, the corresponding accuracy is listed for the tagger after the decorrelation is performed for a FPR of 50%.

Model	Parameters	Accuracy	AUC	$1/\varepsilon_b$	$1/\varepsilon_b$	ε_s	$1/D_{\mathrm{JS}}$
				@ $\varepsilon_s = 30\%$	@ $\varepsilon_s = 50\%$	@ $\varepsilon_b = 1\%$	@ $\varepsilon_b = 1\%$
			Baseline models				
Interaction network	18,144	95.5%	99.0%	4616.9	1028.8	82.8%	4.5
Deep double-*b*	40,344	91.7%	97.2%	578.0	165.3	60.6%	75.3
Deep double-*b*+	38,746	95.3%	98.8%	3863.1	852.7	81.5%	4.4
			Decorrelated models				
Interaction network, DDT	18,144	93.2%	98.5%	2258.7	540.0	75.6%	29,265.3
Deep double-*b*, DDT	40,344	86.8%	96.7%	456.6	136.8	55.9%	48,099.0
Deep double-*b*+, DDT	38,746	92.9%	98.3%	1973.8	466.6	72.9%	15,171.2

VII. RESULTS

In Fig. 4 the performance of the IN, all-particle IN, DDB, and DDB+ algorithms are quantified in a ROC curve. The axes are the TPR, or $H \to b\bar{b}$ tagging efficiency and the false positive rate, or QCD mistagging rate. As shown in Fig. 4, the IN provides an improved performance with respect to the DDB and DDB+ taggers. At a 1% FPR, the IN tagger outperforms the DDB and DDB+ taggers by 37% and 2% in TPR, respectively. Likewise, at a 50% TPR, the IN tagger yields a factor of 6 or 1.2 better background rejection (1/FPR) than the DDB or DDB+ tagger, respectively. Thus, while the additional inputs provide a significant improvement for the DDB+ model, the IN architecture is also important to achieve a better performance with significantly less parameters than the DDB+ model.

We verified that one could match the performance obtained by the IN with a DDB-inspired architecture and expanding the model size. With 150,786 trainable parameters, a DDB architecture achieves the same performance as the IN at the cost of 8 times more parameters. Because of this the IN model holds an advantage in terms of memory usage during inference over this alternative model.

Figure 4 also shows that there is only a modest improvement in the AUC and accuracy by including information in the IN model from neutral particles. For this reason and to preserve robustness to increased pileup, in the following results, we consider the original IN model that excludes neutral particles.

Figure 5 shows an illustration of how the signal and background jet mass distributions change after applying a threshold on the different baseline and DDT-decorrelated tagger scores. Following Ref. [119], we quantify the impact of these algorithms on the mass decorrelation by computing the Jensen-Shannon (JS) divergence:

$$D_{\mathrm{JS}}(P\|Q) = \frac{1}{2}D_{\mathrm{KL}}(P\|M) + \frac{1}{2}D_{\mathrm{KL}}(Q\|M), \qquad (8)$$

where $M = \frac{1}{2}(P+Q)$ is the average of the normalized m_{SD} distributions of the background jets passing (P) and failing (Q) a given tagger score and $D_{\mathrm{KL}}(P\|Q) = \sum_i P_i \log(P_i/Q_i)$ is the Kullback-Leibler (KL) divergence. Larger values of the metric $1/D_{\mathrm{JS}}$ correspond to a better decorrelation.

After applying the mass decorrelation techniques, the performance of each of the taggers worsens slightly but the

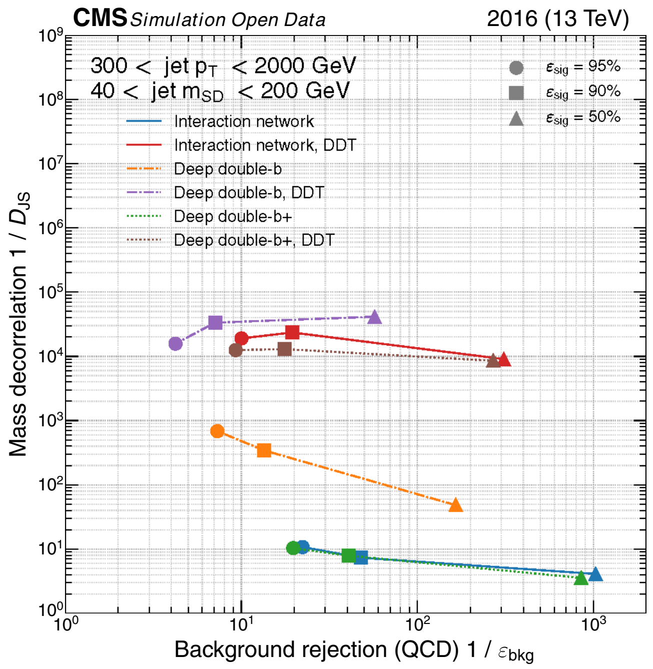


FIG. 6. The mass decorrelation metric $1/D_{\mathrm{JS}}$ as a function of background rejection for the baseline and decorrelated IN, DDB, and DDB+ taggers. The decorrelation is quantified as the inverse of the JS divergence between the background mass distribution passing and failing a given threshold cut on the classifier score. Greater values of this metric correspond to better mass decorrelation. The background rejection is quantified as the inverse of the FPR, while the signal efficiency is equal to the TPR.

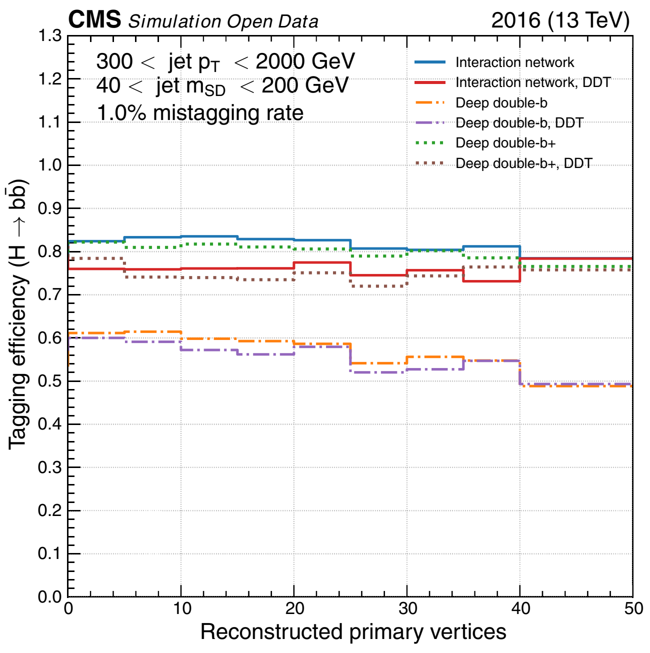


FIG. 7. TPR of the baseline and decorrelated IN, DDB, and DDB+ taggers as a function of the number of reconstructed PVs for a 1% FPR.

IN algorithm still significantly outperforms the DDB and DDB+ taggers. Figure 6 displays the trade-off between the background rejection and $1/D_{\mathrm{JS}}$ at different TPRs for the baseline and DDT-decorrelated algorithms. At a 50% TPR, the decorrelated IN algorithm achieves a significantly better $1/D_{\mathrm{JS}}$ by a factor of about 2,200 while the background rejection decreases by a factor of about 3.3 compared to the baseline IN algorithm. At a 1% FPR, the DDT-decorrelated IN tagger has a TPR of 75.6% compared to the DDT-decorrelated DDB (DDB+) tagger with a 55.9% (72.9%) TPR, corresponding to an improvement of 35% (4%). Table I summarizes different performance metrics for the three considered models and their decorrelated versions. For the DDT models, the corresponding accuracy is listed for the tagger after the decorrelation is performed for a FPR of 50%.

To quantify the dependence on the number of pileup interactions, Fig. 7 shows the performance of the different algorithms as a function of the number of primary vertices in the event, which scales linearly with the number of pileup collisions. Using only charged particles and secondary vertices as input, the IN tagger is robust against an increasing number of pileup interactions, exhibiting behavior similar to the DDB and DDB+ taggers.

VIII. CONCLUSIONS

We presented a novel technique using a graph representation of the jet's constituents and secondary vertices based on an interaction network to identify Higgs bosons decaying to bottom quark-antiquark pairs ($H \to b\bar{b}$) in LHC collisions. This model can operate on a variable number of jet constituents and secondary vertices and does not depend on the ordering schemes of these objects. The interaction network was trained on an open simulation dataset released by the CMS Collaboration in the CERN Open Data Portal. A significant improvement in performance is observed with respect to two alternative taggers based on the deep double-*b* tagger created by the CMS Collaboration. By design, the interaction network uses extended low-level input features for particles and vertices, offers a more flexible representation of jet data, and is robust against the noise generated by pileup collisions. Even when trained with the same set of input features, the interaction network architecture outperforms the deep double-*b* architecture. Thus, while part of the improvement is due to the extended input representation, additional improvement comes from the interaction network architecture, despite using on half as many parameters. The interaction network algorithm implementation and its training code are available at Ref. [129].

Together with the best-performing models, we presented additional models, obtained by applying different decorrelation techniques between the network score and the jet-mass distribution. This was done to minimize the selection bias of the classifier output towards any values of the jet mass, which would make the algorithms suitable for physics analyses relying on the jet mass as a discriminating variable. As expected, the decorrelation procedure results in a reduction of the $H \to b\bar{b}$ identification performance. Nevertheless, the decorrelated interaction network model outperforms the decorrelated deep double-*b* models.

Once applied to a full data analysis, this graph-based tagging algorithm could contribute a substantial improvement to the experimental precision of $H \to b\bar{b}$ measurements, including those sensitive to beyond the standard model physics and the Higgs boson self-coupling. These results motivate further exploration of applications based on interaction networks (and graph neural networks in general) for object tagging and other similar tasks in experimental high energy physics.

ACKNOWLEDGMENTS

This work was possible thanks to the commitment of the CMS Collaboration to release its simulation data through the CERN Open Data Portal. We would like to thank our CMS colleagues and the CERN Open Data team for their effort to promote open access to science. In particular, we thank Kati Lassila-Perini for her precious help. We are grateful to Caltech and the Kavli Foundation for their support of undergraduate student research in cross-cutting areas of machine learning and domain sciences. This work was conducted at "*iBanks*," the AI GPU cluster at Caltech. We acknowledge NVIDIA, SuperMicro and the Kavli Foundation for their support of *iBanks*. This project is partially supported by the United States Department of Energy, Office of High Energy Physics Research under Caltech Contract No. DE-SC0011925. M. P. is supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (Grant Agreement No. 772369). J. M. D. is supported by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the U.S. Department of Energy, Office of Science, Office of High Energy Physics. E. A. M is supported by the Taylor W. Lawrence Research Fellowship and Mellon Mays Research Fellowship.

APPENDIX A: ADDITIONAL MASS DECORRELATION METHODS

In this appendix, we describe and compare two additional mass decorrelation methods to the DDT procedure described in Sec. V. In one method, we train two neural networks simultaneously: the original classifier and an additional network intended to regress the jet mass, known as the adversary. The original classifier is trained to maximally confuse the adversary. After training, the effect is the classifier is not able to discriminate the jet mass. In the other method, we train the classifier with sample weights, such that the QCD background jet mass distribution is reweighted to be identical to that of the $H \to b\bar{b}$ signal. We then compare these procedures to the DDT method.

1. Adversarial training

The secondary adversary network is constructed that consists of three hidden layers each with 64 nodes. The adversary is trained simultaneously with the classifier (interaction network) using the summed postinteraction feature vector $\bar{\mathbf{O}}$ as its input. From this input, the adversary is trained to predict a one-hot encoding of the pivot feature m_{SD}, which we aim to decorrelate from the classifier output. The chosen one-hot encoding corresponds to 40 m_{SD} bins from 40 to 200 GeV. The training begins by initializing the weights from the best classifier training. The adversary is then pretrained for 10 epochs using the Adam algorithm with an initial learning rate of 10^{-4}. During each epoch, the classifier is first trained by minimizing the total loss

$$L = L_{\mathrm{C}} - \lambda L_{\mathrm{adversary}}. \tag{A1}$$

Subsequently, the adversary is trained by minimizing $L_{\mathrm{adversary}}$ using only the background QCD samples. To balance tagging performance and m_{SD} correlation, $\lambda = 10$ was chosen.

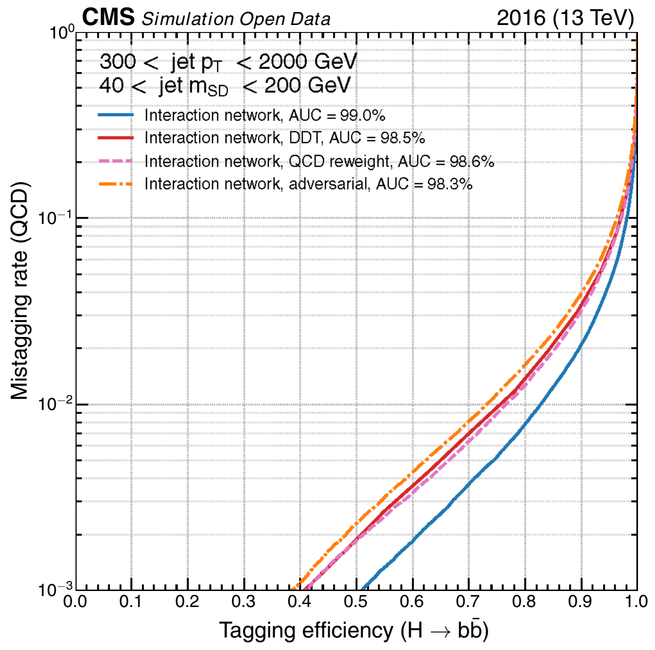


FIG. 8. Performance of IN algorithm compared to the same after applying the three techniques described in the text to decrease the degree to which the tagger score is dependent on the mass of the jet. This results in a lower performance because the algorithm is forced to have reduced correlation with the jet mass.

2. Sample reweighting

While adversarial training requires a complicated tuning process, sample reweighting is a simpler way to achieve the same goal. Individual QCD events are weighted in the loss function based on their mass bin as to match the signal jet mass distribution of the training sample. Given a background event in certain mass bin, with the number of background and signal events in that bin denoted as N_b^{bin} and N_s^{bin}, respectively, the event is weighted by $w_{\mathrm{bin}} = N_s^{\mathrm{bin}}/N_b^{\mathrm{bin}}$.

3. Results

Figure 8 shows a comparison of the ROC curves for the baseline IN algorithm and the versions that were decorrelated using the DDT procedure, adversarial training, and sample reweighting. Table II summarizes a variety of performance metrics for the decorrelated algorithms

TABLE II. Performance metrics of the three decorrelated IN models, including accuracy, area under the ROC curve, background rejection at a true positive rate of 30% and 50%, and true positive rate and mass decorrelation metric $1/D_{\mathrm{JS}}$ at a false positive rate of 1%.

				$1/\varepsilon_b$	$1/\varepsilon_b$	ε_s	$1/D_{\mathrm{JS}}$
Model	Parameters	Accuracy	AUC	@ $\varepsilon_s = 30\%$	@ $\varepsilon_s = 50\%$	@ $\varepsilon_b = 1\%$	@ $\varepsilon_b = 1\%$
Interaction network, adversarial	18,144	94.6%	98.6%	2381.0	540.1	76.5%	124.6
Interaction network, QCD reweight.	18,144	93.4%	98.3%	1864.9	436.2	73.2%	2051.0
Interaction network, DDT	18,144	93.2%	98.5%	2258.7	540.0	75.6%	29,265.3

including $1/D_{\mathrm{JS}}$, which quantifies the success of the decorrelation procedure for a given FPR. As shown in Fig. 8 and Table II, the DDT procedure provides the best decorrelation in terms of $1/D_{\mathrm{JS}}$, and comparable to the best accuracy, AUC, background rejection, and tagging efficiency.

APPENDIX B: MODEL IMPLEMENTED IN ONNX AND TENSORFLOW

In order to integrate the IN algorithm into experimental workflows, it is often necessary to provide the algorithm in other formats. For example, the CMS experimental software framework CMSSW [130] currently only supports ONNX [131], TensorFlow [132], and MXNET [133] models. To perform this conversion, we first translate the PyTorch model into an ONNX representation using the built-in exporter. Then the conversion to TensorFlow is performed with the dedicated TensorFlow backend for ONNX [134]. The trained model in all three formats is available at Ref. [129].

APPENDIX C: DATASET FEATURES

The charged particle features used by the IN and DDB taggers are listed in Table III. The SV features used by both taggers are listed in Table IV, and the high-level features used only by the reconstructed DDB tagger are shown in Table V. Finally, additional features of charged or neutral particles are listed in Table VI to demonstrate the change in the performance of the IN model by including neutral particles.

TABLE III. Charged particle features. The IN and DDB+ models use all of the features, while the DDB algorithm uses the subset of features indicated in bold.

Variable	Description
`track_ptrel`	p_{T} of the charged particle divided by the p_{T} of the AK8 jet
`track_erel`	Energy of the charged particle divided by the energy of the AK8 jet
`track_phirel`	$\Delta\phi$ between the charged particle and the AK8 jet axis
`track_etarel`	$\Delta\eta$ between the charged particle and the AK8 jet axis
`track_deltaR`	ΔR between the charged particle and the AK8 jet axis
`track_drminsv`	ΔR between the associated SVs and the charged particle
`track_drsubjet1`	ΔR between the charged particle and the first soft drop subjet
`track_drsubjet2`	ΔR between the charged particle and the second soft drop subjet
`track_dz`	Longitudinal impact parameter of the track, defined as the distance of closest approach of the track trajectory to the PV projected on to the z direction
`track_dzsig`	Longitudinal impact parameter significance of the track
`track_dxy`	Transverse (2D) impact parameter of the track, defined as the distance of closest approach of the track trajectory to the beam line in the transverse plane to the beam
`track_dxysig`	Transverse (2D) impact parameter of the track
`track_normchi2`	Normalized χ^2 of the track fit
`track_quality`	Track quality: `undefQuality` $= -1$, `loose` $= 0$, `tight` $= 1$, `highPurity` $= 2$, `confirmed` $= 3$, `looseSetWithPV` $= 5$, `highPuritySetWithPV` $= 6$, `discarded` $= 7$, `qualitySize` $= 8$
`track_dptdpt`	Track covariance matrix entry (p_{T}, p_{T})
`track_detadeta`	Track covariance matrix entry (η, η)
`track_dphidphi`	Track covariance matrix entry (ϕ, ϕ)
`track_dxydxy`	Track covariance matrix entry (d_{xy}, d_{xy})
`track_dzdz`	Track covariance matrix entry (d_z, d_z)
`track_dxydz`	Track covariance matrix entry (d_{xy}, d_z)
`track_dphidz`	Track covariance matrix entry (d_ϕ, d_z)
`track_dlambdadz`	Track covariance matrix entry (λ, d_z)
`trackBTag_EtaRel`	$\Delta\eta$ between the track and the AK8 jet axis
`trackBTag_PtRatio`	Component of track momentum perpendicular to the AK8 jet axis, normalized to the track momentum
`trackBTag_PParRatio`	Component of track momentum parallel to the AK8 jet axis, normalized to the track momentum
`trackBTag_Sip2dVal`	Transverse (2D) signed impact parameter of the track
`trackBTag_Sip2dSig`	Transverse (2D) signed impact parameter significance of the track
`trackBTag_Sip3dVal`	3D signed impact parameter of the track
`trackBTag_Sip3dSig`	3D signed impact parameter significance of the track
`trackBTag_JetDistVal`	Minimum track approach distance to the AK8 jet axis

TABLE IV. Secondary vertex features. The IN and DDB+ models use all of the features, while the DDB algorithm uses the subset of features indicated in bold.

Variable	Description
`sv_ptrel`	p_T of the SV divided by the p_T of the AK8 jet
`sv_erel`	Energy of the SV divided by the energy of the AK8 jet
`sv_phirel`	$\Delta\phi$ between the SV and the AK8 jet axis
`sv_etarel`	$\Delta\eta$ between the SV and the AK8 jet axis
`sv_deltaR`	ΔR between the SV and the AK8 jet axis
`sv_pt`	p_T of the SV
`sv_mass`	Mass of the SV
`sv_ntracks`	Number of tracks associated with the SV
`sv_normchi2`	Normalized χ^2 of the SV fit
`sv_costhetasvpv`	$\cos\theta$ between the SV and the PV
`sv_dxy`	Transverse (2D) flight distance of the SV
`sv_dxysig`	Transverse (2D) flight distance significance of the SV
`sv_d3d`	3D flight distance of the SV
`sv_d3dsig`	3D flight distance significance of the SV

TABLE V. High-level features used by the DDB algorithm.

Variable	Description
`fj_jetNTracks`	Number of tracks associated with the AK8 jet
`fj_nSV`	Number of SVs associated with the AK8 jet ($\Delta R < 0.7$)
`fj_tau0_trackEtaRel_0`	Smallest track $\Delta\eta$ relative to the jet axis, associated to the first N-subjettiness axis
`fj_tau0_trackEtaRel_1`	Second smallest track $\Delta\eta$ relative to the jet axis, associated to the first N-subjettiness axis
`fj_tau0_trackEtaRel_2`	Third smallest track $\Delta\eta$ relative to the jet axis, associated to the first N-subjettiness axis
`fj_tau1_trackEtaRel_0`	Smallest track $\Delta\eta$ relative to the jet axis, associated to the second N-subjettiness axis
`fj_tau1_trackEtaRel_1`	Second smallest track $\Delta\eta$ relative to the jet axis, associated to the second N-subjettiness axis
`fj_tau1_trackEtaRel_2`	Third smallest track $\Delta\eta$ relative to the jet axis, associated to the second N-subjettiness axis
`fj_tau_flightDistance2dSig_0`	Transverse (2D) flight distance significance between the PV and the SV with the smallest uncertainty on the 3D flight distance associated to the first N-subjettiness axis
`fj_tau_flightDistance2dSig_1`	Transverse (2D) flight distance significance between the PV and the SV with the smallest uncertainty on the 3D flight distance associated to the second N-subjettiness axis
`fj_tau_vertexDeltaR_0`	ΔR between the first N-subjettiness axis and SV direction
`fj_tau_vertexEnergyRatio_0`	SV energy ratio for the first N-subjettiness axis, defined as the total energy of all SVs associated with the first N-subjettiness axis divided by the total energy of all the tracks associated with the AK8 jet that are consistent with the PV
`fj_tau_vertexEnergyRatio_1`	SV energy ratio for the second N-subjettiness axis
`fj_tau_vertexMass_0`	SV mass for the first N-subjettiness axis, defined as the invariant mass of all tracks from SVs associated with the first N-subjettiness axis
`fj_tau_vertexMass_1`	SV mass for the second N-subjettiness axis
`fj_trackSip2dSigAboveBottom_0`	Track 2D signed impact parameter significance of the first track lifting the combined invariant mass of the tracks above the b hadron threshold mass (5.2 GeV)
`fj_trackSip2dSigAboveBottom_1`	Track 2D signed impact parameter significance of the second track lifting the combined invariant mass of the tracks above the b hadron threshold mass (5.2 GeV)
`fj_trackSip2dSigAboveCharm_0`	Track 2D signed impact parameter significance of the first track lifting the combined invariant mass of the tracks above the c hadron threshold mass (1.5 GeV)
`fj_trackSipdSig_0`	Largest track 3D signed impact parameter significance
`fj_trackSipdSig_1`	Second largest track 3D signed impact parameter significance
`fj_trackSipdSig_2`	Third largest track 3D signed impact parameter significance
`fj_trackSipdSig_3`	Fourth largest track 3D signed impact parameter significance
`fj_trackSipdSig_0_0`	Largest track 3D signed impact parameter significance associated to the first N-subjettiness axis

(Table continued)

TABLE V. *(Continued)*

Variable	Description
fj_trackSipdSig_0_1	Second largest track 3D signed impact parameter significance associated to the first N-subjettiness axis
fj_trackSipdSig_1_0	Largest track 3D signed impact parameter significance associated to the second N-subjettiness axis
fj_trackSipdSig_1_1	Second largest track 3D signed impact parameter significance associated to the second N-subjettiness axis
fj_z_ratio	z ratio variable as defined in Ref. [10]

TABLE VI. Additional features for charged or neutral particles. The all-particle IN model uses these features.

Variable	Description
pfcand_ptrel	p_T of the charged or neutral particle divided by the p_T of the AK8 jet
pfcand_erel	Energy of the charged or neutral particle divided by the energy of the AK8 jet
pfcand_phirel	$\Delta\phi$ between the charged or neutral particle and the AK8 jet axis
pfcand_etarel	$\Delta\eta$ between the charged or neutral particle and the AK8 jet axis
pfcand_deltaR	ΔR between the charged or neutral particle and the AK8 jet axis
pfcand_puppiw	Pileup per particle identification (PUPPI) weight [135] for the charged or neutral particle
pfcand_drminsv	ΔR between the associated SVs and the charged or netural particle
pfcand_drsubjet1	ΔR between the charged or neutral particle and the first soft drop subjet
pfcand_drsubjet2	ΔR between the charged or neutral particle and the second soft drop subjet
pfcand_hcalFrac	Fraction of energy of the charged or neutral particle deposited in the hadron calorimeter

[1] M. H. Seymour, Tagging a heavy Higgs boson, in *ECFA Large Hadron Collider Workshop, Aachen, Germany 1990, Proceedings* (1991), p. 557.

[2] M. H. Seymour, Searches for new particles using cone and cluster jet algorithms: A comparative study, Z. Phys. C **62**, 127 (1994).

[3] M. H. Seymour, The average number of subjets in a hadron collider jet, Nucl. Phys. **B421**, 545 (1994).

[4] J. M. Butterworth, B. E. Cox, and J. R. Forshaw, *WW* scattering at the CERN LHC, Phys. Rev. D **65**, 096014 (2002).

[5] J. M. Butterworth, A. R. Davison, M. Rubin, and G. P. Salam, Jet Substructure as a New Higgs Search Channel at the LHC, Phys. Rev. Lett. **100**, 242001 (2008).

[6] A. J. Larkoski, I. Moult, and B. Nachman, Jet substructure at the Large Hadron Collider: A review of recent advances in theory and machine learning, Phys. Rep. **841**, 1 (2020).

[7] S. Marzani, G. Soyez, and M. Spannowsky, *Looking Inside Jets: An Introduction to Jet Substructure and Boosted-Object Phenomenology*, Lect. Notes Phys. Vol. 958 (Springer, New York, 2019).

[8] R. Kogler *et al.*, Jet substructure at the Large Hadron Collider: Experimental review, Rev. Mod. Phys. **91**, 045003 (2019).

[9] CMS Collaboration, A multi-dimensional search for new heavy resonances decaying to boosted WW, WZ, or ZZ boson pairs in the dijet final state at 13 TeV, Eur. Phys. J. C **80**, 237 (2020).

[10] CMS Collaboration, Identification of heavy-flavour jets with the CMS detector in pp collisions at 13 TeV, J. Instrum. **13**, P05011 (2018).

[11] CMS Collaboration, Performance of deep tagging algorithms for boosted double quark jet topology in proton-proton collisions at 13 TeV with the Phase-0 CMS detector, CMS Detector Performance Note CMS-DP-2018-046, 2018.

[12] CMS Collaboration, Boosted jet identification using particle candidates and deep neural networks, CMS Detector Performance Note CMS-DP-2017-049, 2017.

[13] CMS Collaboration, Identification of heavy, energetic, hadronically decaying particles using machine-learning techniques, J. Instrum. **15**, P06005 (2020).

[14] ATLAS Collaboration, Identification of boosted Higgs bosons decaying into b-quark pairs with the ATLAS detector at 13 TeV, Eur. Phys. J. C **79**, 836 (2019).

[15] J. Lin, M. Freytsis, I. Moult, and B. Nachman, Boosting $H \to b\bar{b}$ with machine learning, J. High Energy Phys. 10 (2018) 101.

[16] CMS Collaboration, Inclusive Search for a Highly Boosted Higgs Boson Decaying to a Bottom Quark-Antiquark Pair, Phys. Rev. Lett. **120**, 071802 (2018).

[17] CMS Collaboration, Measurement and interpretation of differential cross sections for Higgs boson production at $\sqrt{s} = 13$ TeV, Phys. Lett. B **792**, 369 (2019).
[18] C. Grojean, E. Salvioni, M. Schlaffer, and A. Weiler, Very boosted Higgs in gluon fusion, J. High Energy Phys. 05 (2014) 022.
[19] K. Becker *et al.*, Precise predictions for boosted Higgs production, arXiv:2005.07762.
[20] ATLAS Collaboration, Search for boosted resonances decaying to two b-quarks and produced in association with a jet at $\sqrt{s} = 13$ TeV with the ATLAS detector, ATLAS Conference Note ATLAS-CONF-2018-052 (2018).
[21] S. Dawson, I. Lewis, and M. Zeng, Usefulness of effective field theory for boosted Higgs production, Phys. Rev. D **91**, 074012 (2015).
[22] M. Schlaffer, M. Spannowsky, M. Takeuchi, A. Weiler, and C. Wymant, Boosted Higgs shapes, Eur. Phys. J. C **74**, 3120 (2014).
[23] M. Grazzini, A. Ilnicka, M. Spira, and M. Wiesemann, Effective field theory for Higgs properties parametrisation: The transverse momentum spectrum case, in *52nd Rencontres de Moriond on QCD and High Energy Interactions* (2017), p. 23.
[24] M. Grazzini, A. Ilnicka, M. Spira, and M. Wiesemann, Modeling BSM effects on the Higgs transverse-momentum spectrum in an EFT approach, J. High Energy Phys. 03 (2017) 115.
[25] F. Bishara, U. Haisch, P. F. Monni, and E. Re, Constraining Light-Quark Yukawa Couplings from Higgs Distributions, Phys. Rev. Lett. **118**, 121801 (2017).
[26] Y.-Y. Li, R. Nicolaidou, and S. Paganis, Exclusion of heavy, broad resonances from precise measurements of WZ and VH final states at the LHC, Eur. Phys. J. C **79**, 348 (2019).
[27] J. Amacker *et al.*, Higgs self-coupling measurements using deep learning and jet substructure in the $b\bar{b}b\bar{b}$ final state, arXiv:2004.04240.
[28] A. Dainese *et al.*, eds., Report on the physics at the HL-LHC, and perspectives for the HE-LHC, CERN Yellow Reports: Monographs Vol. **7** (CERN, Geneva, 2019).
[29] F. Kling T. Plehn, and P. Schichtel, Maximizing the significance in Higgs boson pair analyses, Phys. Rev. D **95**, 035026 (2017).
[30] M. Grazzini, G. Heinrich, S. Jones, S. Kallweit, M. Kerner, J. M. Lindert, and J. Mazzitelli, Higgs boson pair production at NNLO with top quark mass effects, J. High Energy Phys. 05 (2018) 059.
[31] M. Niepert, M. Ahmed, and K. Kutzkov, Learning convolutional neural networks for graphs, in *Proceedings of the 33rd International Conference on Machine Learning, Proceedings of Machine Learning Research*, edited by M. F. Balcan and K. Q. Weinberger (PMLR, New York, 2016), p. 2014.
[32] T. N. Kipf and M. Welling, Semi-supervised classification with graph convolutional networks, in *5th International Conference on Learning Representations, ICLR 2017, Conference Track Proceedings* (OpenReview, Amherst, 2017).
[33] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, PointNet: Deep learning on point sets for 3D classification and segmentation, in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, New York, 2017).
[34] Y. Wang *et al.*, Dynamic graph CNN for learning on point clouds, ACM Trans. Graph. **38**, 146 (2019).
[35] A. Grover, A. Zweig, and S. Ermon, Graphite: Iterative generative modeling of graphs, in *Proceedings of the 36th International Conference on Machine Learning, Proceedings of Machine Learning Research*, Vol. 97, edited by K. Chaudhuri and R. Salakhutdinov (PMLR, Long Beach, 2019), p. 2434, http://proceedings.mlr.press/v97/.
[36] J. You, R. Ying, X. Ren, W. L. Hamilton, and J. Leskovec, GraphRNN: Generating realistic graphs with deep auto-regressive models, in *Proceedings of the 35th International Conference on Machine Learning, Proceedings of Machine Learning Research*, Vol. 80 (PMLR, Stockholm, 2018), p. 5708, see http://proceedings.mlr.press/v80/.
[37] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, Spectral networks and locally connected networks on graphs, in *2nd International Conference on Learning Representations, ICLR 2014, Conference Track Proceedings* (ICLR, Banf, 2014), arXiv:1312.6203.
[38] D. Zheng, V. Luo, J. Wu, and J. B. Tenenbaum, Unsupervised learning of latent physical properties using perception-prediction networks, in *Proceedings of the 34th Conference on Uncertainty in Artificial Intelligence*, edited by A. Globerson and R. Silva (AUAI Press, Corvallis, 2018), p. 497, http://auai.org/uai2018/proceedings/uai2018proceedings.pdf.
[39] P. W. Battaglia *et al.*, Relational inductive biases, deep learning, and graph networks, arXiv:1806.01261.
[40] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE **11**, 2278 (1998).
[41] K. He, X. Zhang, S. Ren, and J. Sun, Deep residual learning for image recognition, in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, New York, 2016), p. 770.
[42] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, Geometric deep learning: Going beyond euclidean data, IEEE Signal Process. Mag. **34**, 18 (2017).
[43] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, Gated graph sequence neural networks, in *4th International Conference on Learning Representations, ICLR 2016, Conference Track Proceedings* (ICLR, San Juan, 2016), arXiv:1511.05493.
[44] P. W. Battaglia, R. Pascanu, M. Lai, D. J. Rezende, and K. Kavukcuoglu, Interaction networks for learning about objects, relations and physics, in *Advances in Neural Information Processing Systems 29*, edited by D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (Curran Associates, Inc., Red Hook, New York, 2016), p. 4502.
[45] A. J. Larkoski, G. P. Salam, and J. Thaler, Energy correlation functions for jet substructure, J. High Energy Phys. 06 (2013) 108.
[46] I. Moult, L. Necib, and J. Thaler, New angles on energy correlation functions, J. High Energy Phys. 12 (2016) 153,

[47] E. A. Moreno *et al.*, JEDI-net: A jet identification algorithm based on interaction networks, Eur. Phys. J. C **80**, 58 (2020).

[48] GEANT4 Collaboration, GEANT4—A simulation toolkit, Nucl. Instrum. Methods Phys. Res., Sect. A **506**, 250 (2003).

[49] CERN Open Data Portal, http://opendata.cern.ch (2014).

[50] H. Kirschenmann (CMS Collaboration), Jet performance in CMS, Proc. Sci., EPS-HEP2013 (**2013**) 433.

[51] D. Guest, K. Cranmer, and D. Whiteson, Deep learning and its application to LHC physics, Annu. Rev. Nucl. Part. Sci. **68**, 161 (2018).

[52] Y. LeCun and Y. Bengio, Convolutional networks for images, speech, and time series, in *The Handbook of Brain Theory and Neural Networks*, edited by M. A. Arbib (MIT Press, Cambridge, Massachusetts, 1995), Vol. 3361, p. 255.

[53] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, Face recognition: a convolutional neural-network approach, IEEE Trans. Neural Networks **8**, 98 (1997).

[54] A. Krizhevsky, I. Sutskever, and G. E. Hinton, ImageNet classification with deep convolutional neural networks, in *Advances in Neural Information Processing Systems 25* (Curran Associates, Inc., Red Hook, New York, 2012), p. 1097.

[55] R. J. Williams and D. Zipser, A learning algorithm for continually running fully recurrent neural networks, Neural Comput. **1**, 270 (1989).

[56] A. Graves, A.-R. Mohamed, and G. Hinton, Speech recognition with deep recurrent neural networks, in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing* (IEEE, New York, 2013), p. 6645.

[57] S. Hochreiter and J. Schmidhuber, Long short–term memory, Neural Comput. **9**, 1735 (1997).

[58] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling, in *Deep Learning and Representation Learning Workshop at the 28th Neural Information Processing Systems* (dlworkshop.org, Montreal, 2014), arXiv:1412.3555.

[59] L. de Oliveira, M. Kagan, L. Mackey, B. Nachman, and A. Schwartzman, Jet-images—Deep learning edition, J. High Energy Phys. 07 (2016) 069.

[60] S. Macaluso and D. Shih, Pulling out all the tops with computer vision and deep learning, J. High Energy Phys. 10 (2018) 121.

[61] G. Kasieczka, T. Plehn, M. Russell, and T. Schell, Deep-learning top taggers or the end of QCD? J. High Energy Phys. 05 (2017) 006.

[62] P. T. Komiske, E. M. Metodiev, and M. D. Schwartz, Deep learning in color: Towards automated quark/gluon jet discrimination, J. High Energy Phys. 01 (2017) 110.

[63] P. Baldi, K. Bauer, C. Eng, P. Sadowski, and D. Whiteson, Jet substructure classification in high-energy physics with deep neural networks, Phys. Rev. D **93**, 094034 (2016).

[64] K. Datta and A. J. Larkoski, Novel jet observables from machine learning, J. High Energy Phys. 03 (2018) 086.

[65] A. Butter, G. Kasieczka, T. Plehn, and M. Russell, Deep-learned top tagging with a Lorentz layer, SciPost Phys. **5**, 028 (2018).

[66] P. T. Komiske, E. M. Metodiev, and J. Thaler, Energy flow polynomials: A complete linear basis for jet substructure, J. High Energy Phys. 04 (2018) 013.

[67] G. Louppe, K. Cho, C. Becot, and K. Cranmer, QCD-aware recursive neural networks for jet physics, J. High Energy Phys. 01 (2019) 057.

[68] S. Egan *et al.*, Long Short-Term Memory (LSTM) networks with jet constituents for boosted top tagging at the LHC, arXiv:1711.09059.

[69] T. Cheng, Recursive neural networks in quark/gluon tagging, Comput. Software Big Sci. **2**, 3 (2018).

[70] D. Guest, J. Collado, P. Baldi, S.-C. Hsu, G. Urban, and D. Whiteson, Jet flavor classification in high-energy physics with deep neural networks, Phys. Rev. D **94**, 112002 (2016).

[71] G. Kasieczka *et al.*, The machine learning landscape of top taggers, SciPost Phys. **7**, 014 (2019).

[72] G. Kasieczka, T. Plehn, J. Thompson, and M. Russel, Top quark tagging reference dataset https://doi.org/10.5281/zenodo.2603256 (2019), Zenodo.

[73] H. Qu and L. Gouskos, ParticleNet: Jet tagging via particle clouds, Phys. Rev. D **101**, 056019 (2020).

[74] T. Heimel, G. Kasieczka, T. Plehn, and J. M. Thompson, QCD or What? SciPost Phys. **6**, 030 (2019).

[75] M. Farina, Y. Nakai, and D. Shih, Searching for new physics with deep autoencoders, Phys. Rev. D **101**, 075021 (2020).

[76] B. M. Dillon, D. A. Faroughy, and J. F. Kamenik, Uncovering latent jet substructure, Phys. Rev. D **100**, 056002 (2019).

[77] J. H. Collins, K. Howe, and B. Nachman, Anomaly Detection for Resonant New Physics with Machine Learning, Phys. Rev. Lett. **121**, 241803 (2018).

[78] J. H. Collins, K. Howe, and B. Nachman, Extending the search for new resonances with machine learning, Phys. Rev. D **99**, 014038 (2019).

[79] L. M. Dery, B. Nachman, F. Rubbo, and A. Schwartzman, Weakly supervised classification in high energy physics, J. High Energy Phys. 05 (2017) 145.

[80] E. M. Metodiev, B. Nachman, and J. Thaler, Classification without labels: Learning from mixed samples in high energy physics, J. High Energy Phys. 10 (2017) 174,

[81] P. T. Komiske, E. M. Metodiev, B. Nachman, and M. D. Schwartz, Learning to classify from impure samples with high-dimensional data, Phys. Rev. D **98**, 011502 (2018).

[82] T. Cohen, M. Freytsis, and B. Ostdiek, (Machine) Learning to do more with less, J. High Energy Phys. 02 (2018) 034.

[83] B. Nachman and D. Shih, Anomaly detection with density estimation, Phys. Rev. D **101**, 075042 (2020).

[84] A. Andreassen, B. Nachman, and D. Shih, Simulation assisted likelihood-free anomaly detection, Phys. Rev. D **101**, 095004 (2020).

[85] A. Tripathee, W. Xue, A. Larkoski, S. Marzani, and J. Thaler, Jet substructure studies with CMS open data, Phys. Rev. D **96**, 074003 (2017).

[86] A. Larkoski, S. Marzani, J. Thaler, A. Tripathee, and W. Xue, Exposing the QCD Splitting Function with CMS Open Data, Phys. Rev. Lett. **119**, 132003 (2017).

[87] M. Andrews, M. Paulini, S. Gleyzer, and B. Poczos, End-to-end physics event classification with CMS open data: Applying image-based deep learning to detector data for the direct classification of collision events at the LHC, Comput. Software Big Sci. **4**, 6 (2020).

[88] M. Andrews *et al.*, End-to-end jet classification of quarks and gluons with the CMS open data, arXiv:1902.08276.

[89] P. T. Komiske, E. M. Metodiev, and J. Thaler, Metric Space of Collider Events, Phys. Rev. Lett. **123**, 041801 (2019).

[90] P. T. Komiske, R. Mastandrea, E. M. Metodiev, P. Naik, and J. Thaler, Exploring the space of jets with CMS open sata, Phys. Rev. D **101**, 034009 (2020).

[91] M. Cacciari, G. P. Salam, and G. Soyez, The anti-k_t jet clustering algorithm, J. High Energy Phys. 04 (2008) 063.

[92] M. Cacciari, G. P. Salam, and G. Soyez, FastJet user manual, Eur. Phys. J. C **72**, 1896 (2012).

[93] I. Henrion, J. Brehmer, J. Bruna, K. Cho, K. Cranmer, G. Louppe, and G. Rochette, Neural message passing for jet physics, in *Deep Learning for Physical Sciences Workshop at the 31st Conference on Neural Information Processing Systems* (dl4physicalsciences.github.io, Long Beach, 2017).

[94] P. T. Komiske, E. M. Metodiev, and J. Thaler, Energy flow networks: Deep sets for particle jets, J. High Energy Phys. 01 (2019) 121.

[95] M. Abdughani, J. Ren, L. Wu, and J. M. Yang, Probing stop pair production at the LHC with graph neural networks, J. High Energy Phys. 08 (2019) 055.

[96] N. Choma *et al.*, Graph neural networks for IceCube signal classification, arXiv:1809.06166.

[97] S. Farrell *et al.*, Novel deep learning methods for track reconstruction, in 4th International Workshop Connecting The Dots (2018), arXiv:1810.06111.

[98] X. Ju *et al.*, Graph neural networks for particle reconstruction in high energy physics detectors, in *Machine Learning and the Physical Sciences Workshop at the 33rd Conference on Neural Information Processing Systems* (ml4physicalsciences.github.io, Vancouver, 2019), arXiv:2003.11603.

[99] J. Arjona Martínez, O. Cerri, M. Spiropulu, J. R. Vlimant, and M. Pierini, Pileup mitigation at the Large Hadron Collider with graph neural networks, Eur. Phys. J. Plus **134**, 333 (2019).

[100] S. R. Qasim, J. Kieseler, Y. Iiyama, and M. Pierini, Learning representations of irregular particle-detector geometry with distance-weighted graph networks, Eur. Phys. J. C **79**, 608 (2019).

[101] J. Kieseler, Object condensation: One-stage grid-free multi-object reconstruction in physics detectors, graph and image data, arXiv:2002.03605.

[102] L. Gray, T. Klijnsma, and S. Ghosh, A dynamic reduction network for point clouds, arXiv:2003.08013.

[103] L. Randall and R. Sundrum, Large Mass Hierarchy from a Small Extra Dimension, Phys. Rev. Lett. **83**, 3370 (1999).

[104] T. Sjöstrand *et al.*, An introduction to PYTHIA 8.2, Comput. Phys. Commun. **191**, 159 (2015).

[105] CMS Collaboration, Event generator tunes obtained from underlying event and multiparton scattering measurements, Eur. Phys. J. C **76**, 155 (2016).

[106] NNPDF Collaboration, Parton distributions with LHC data, Nucl. Phys. **B867**, 244 (2013).

[107] CMS Collaboration, J. Duarte, Sample with jet, track and secondary vertex properties for Hbb tagging ML studies, HiggsToBBNTuple_HiggsToBB_QCD_RunII_13TeV_MC, https://doi.org/10.7483/OPENDATA.CMS.JGJX.MS7Q (2019). CERN Open Data Portal.

[108] CMS Collaboration, Particle-flow reconstruction and global event description with the cms detector, J. Instrum. **12**, P10003 (2017).

[109] M. Dasgupta, A. Fregoso, S. Marzani, and G. P. Salam, Towards an understanding of jet substructure, J. High Energy Phys. 09 (2013) 029.

[110] A. J. Larkoski, S. Marzani, G. Soyez, and J. Thaler, Soft drop, J. High Energy Phys. 05 (2014) 146.

[111] N. Ketkar, Introduction to PyTorch, in *Deep Learning with Python* (Springer, New York, 2017), p. 195.

[112] A. Paszke *et al.*, PyTorch: An imperative style, high-performance deep learning library, in *Advances in Neural Information Processing Systems 32*, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett (Curran Associates, Inc., RedHook, New York, 2019), p. 8026.

[113] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, in *3rd International Conference for Learning Representations, ICLR 2015* (ICLR, San Diego, 2015), arXiv:1412.6980.

[114] Y. Yao, L. Rosasco, and A. Caponnetto, On early stopping in gradient descent learning, Constr. Approx. **26**, 289 (2007).

[115] H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, and M. Marchand, Domain-adversarial neural networks, in *Second Workshop on Transfer and Multi-Task Learning: Theory meets Practice at the 28th Conference on Neural Information Processing Systems* (MTL, Montreal, 2014), arXiv:1412.4446.

[116] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. March, and V. Lempitsky, Domain-adversarial training of neural networks, J. Mach. Learn. Res. **17**, 1 (2016), http://jmlr.org/papers/v17/15-239.html.

[117] G. Louppe, M. Kagan, and K. Cranmer, Learning to pivot with adversarial networks, in *Advances in Neural Information Processing Systems 30*, edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Curran Associates, Inc., Red Hook, New York, 2017), p. 981, arXiv:1611.01046.

[118] C. Shimmin, P. Sadowski, P. Baldi, E. Weik, D. Whiteson, E. Goul, and A. Søgaard, Decorrelated jet substructure tagging using adversarial neural networks, Phys. Rev. D **96**, 074034 (2017).

[119] ATLAS Collaboration, Performance of mass-decorrelated jet substructure observables for hadronic two-body decay tagging in ATLAS, ATLAS Public Note ATL-PHYS-PUB-2018-014, 2018.

[120] L. Bradshaw, R. K. Mishra, A. Mitridate, and B. Ostdiek, Mass agnostic jet taggers, SciPost Phys. **8**, 011 (2020).

[121] J. Dolen, P. Harris, S. Marzani, S. Rappoccio, and N. Tran, Thinking outside the ROCs: Designing Decorrelated Taggers (DDT) for jet substructure, J. High Energy Phys. 05 (2016) 156.

[122] J. H. Friedman, Stochastic gradient boosting, Computational Statistics and Data Analysis **38**, 367 (2002).

[123] J. H. Friedman, Greedy function approximation: A gradient boosting machine, Ann. Stat. **29**, 1189 (2001).

[124] S. Ioffe and C. Szegedy, Batch Normalization: Accelerating deep network training by reducing internal covariate shift, in *Proceedings of the 32nd International Conference on International Conference on Machine Learning*, Vol. 37, edited by F. Bach and D. Blei (PMLR, Lille, 2015), p. 448.

[125] S. Kiranyaz, T. Ince, R. Hamila, and M. Gabbouj, Convolutional neural networks for patient-specific ECG classification, in *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)* (IEEE, New York, 2015), p. 2608.

[126] A. F. Agarap, Deep learning using rectified linear units (ReLU), arXiv:1803.08375.

[127] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. **15**, 1929 (2014), http://www.jmlr.org/papers/v15/srivastava14a.html.

[128] F. Chollet *et al.*, Keras, https://keras.io (2015).

[129] E. Moreno, J. Duarte, and A. Periwal, eric-moreno/IN: v1.5, https://github.com/eric-moreno/IN (2020).

[130] CMS Collaboration, CMS physics: Technical design report Volume 1: Detector performance and software, CMS Technical Design Report CERN-LHCC-2006-001, 2006.

[131] Open Neural Network Exchange Collaboration, ONNX, https://onnx.ai/ (2017).

[132] M. Abadi *et al.*, TensorFlow: Large-scale machine learning on heterogeneous distributed systems, http://download.tensorflow.org/paper/whitepaper2015.pdf (2015).

[133] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang, MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems, in *Workshop on Machine Learning Systems at the 29th Conference on Neural Information Processing Systems* (LearningSys, Montreal, 2015), arXiv:1512.01274.

[134] A. Jacob, T. Jin, G.-T. Bercea, and W. Hu, onnx/onnx-tensorflow: tf-1.x, https://github.com/onnx/onnx-tensorflow (2020).

[135] D. Bertolini, P. Harris, M. Low, and N. Tran, Pileup per particle identification, J. High Energy Phys. 10 (2014) 059.