FERMILAB-PUB-19-492-CMS-E

# 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, Maria Spiropulu**
California Institute of Technology

**Javier M. Duarte**
University of California San Diego
Fermi National Accelerator Laboratory (FNAL)

**Maurizio Pierini**
European Organization for Nuclear Research (CERN)

## Abstract

We develop a jet identification algorithm based on an interaction network, designed to identify high-momentum Higgs bosons decaying to bottom quark-antiquark pairs, distinguish them from ordinary jets originating from the hadronization of quarks and gluons. The algorithm's inputs are features of the reconstructed charged particles in a jet and the secondary vertices associated to 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 accurate 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.

## 1 Introduction

Jets are collimated showers of particles resulting from the hadronization 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 much more complex task, with new jet topologies becoming accessible (see Fig. 1). Due to the large center-of-mass energy available in LHC collisions, heavy particles, such as W, Z, or Higgs (H) bosons or top (t) quarks may be produced with high transverse momentum ($p_{\mathrm{T}}$). These particles can decay to all-quark final states. Due to the large $p_{\mathrm{T}}$ 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 comes with an additional peculiarity. In fact, b hadrons are characterized by a lifetime of

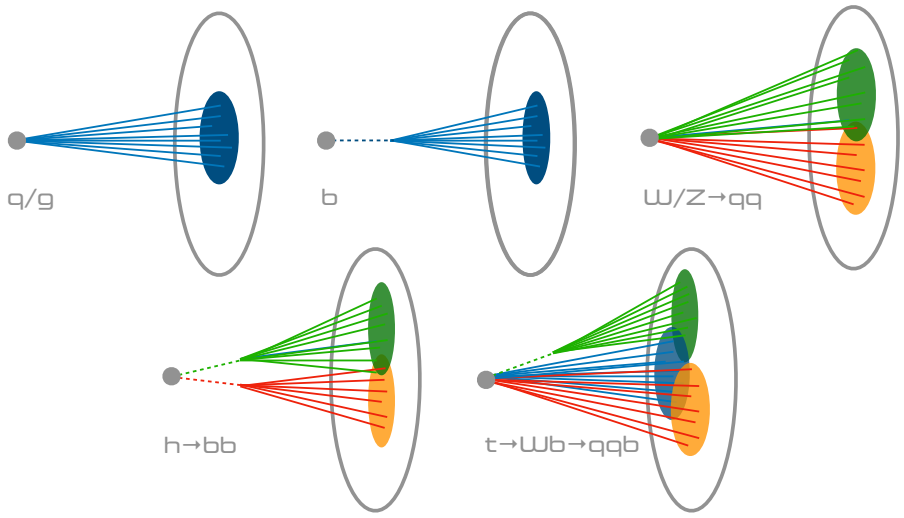


Figure 1: Pictorial representation of ordinary quark and gluon jets (top left), b jets (top center), and boosted-jet topology, emerging from high-$p_{\mathrm{T}}$ W/Z bosons (top right), H bosons (bottom left), and top quarks (bottom right) decaying to all-quarks final states.

approximately 1.5 ps, which induces a detectable displacement between the proton-collision point and the point where the b quark decays.

The identification of jets from heavy resonances relies on *jet substructure* techniques, designed to highlight the presence of clusters of particles, or *prongs*, inside the jet. An extensive review of these techniques is provided in Ref. [1]. 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. [2]. The jet mass plays a special role in physics analyses exploiting jet substructure, as described for instance in Ref. [3]. 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. 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 results in a clean experimental signature: a secondary vertex (SV), displaced from the primary vertex (PV). Modern particle detectors are equipped with a vertex detector and can accurately determine SV positions and their separation from the PV, even in a dense environment like a high-$p_{\mathrm{T}}$ jet. This feature is particularly important for tagging a Higgs boson decaying to a bottom quark-antiquark pair ($\mathrm{H} \to \mathrm{b\bar{b}}$) because all of the jet constituents originate from two displaced vertices.

Recently, several approaches based on deep learning have been proposed to optimize jet tagging algorithms (see Sec. 2), 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 CMS and ATLAS collaborations have investigated the optimal way to combine substructure, tracking, and vertexing information to enhance the tagging efficiency for high-$p_{\mathrm{T}}$ $\mathrm{H} \to \mathrm{b\bar{b}}$ decays [4–7]. This is an important task in particle physics because measurements of high-$p_{\mathrm{T}}$ $\mathrm{H} \to \mathrm{b\bar{b}}$ decays may help resolve the loop induced and tree-level contributions to the gluon fusion process [8, 9] and provide an alternative approach to study the top quark Yukawa coupling in addition to the $\mathrm{t\bar{t}H}$ process [10, 11].

In this work, we propose to identify $\mathrm{H} \to \mathrm{b\bar{b}}$ jets with an interaction network (IN). In Ref. [12], 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. Although there is no direct analogy between a set of physical objects evolving in time and the particle

constituents of a jet, we showed in Ref. [13] that the IN architecture outperforms other deep neural networks (dense, convolutional, and recurrent networks) for a jet-substructure classification task. In this paper, we extend this result to the case of $H \to b\bar{b}$ tagging. 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 [14]. 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 a state-of-the-art $H \to b\bar{b}$ tagging algorithm in CMS, the deep double-b (DDB) tagger [5].

The IN tagger only relies on information related to charged particles, which (unlike neutral particles) can be tracked back to their point of origin: the PV of the high-$p_T$ collision, any SV generated in the collision, or additional PVs originated by simultaneous proton-proton collisions (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 charge hadron subtraction (CHS) [15]. On the other hand, we consider an extended representation of each charged particle, with 22 additional features with respect to the DDB tagger (as discussed in Section 3). As a result of this, we obtain a sizable improvement in tagging performance despite ignoring neutral particles.

This paper is structured as follows: we discuss related works in section 2. Section 3 gives a brief description of the data sets used. Sections 4 and 5 describe the IN architecture and the algorithms used to decorrelate its score from the jet mass distribution. Section 6 describes the baseline DDB algorithm, respectively. Results are presented in section 7. Conclusions are given in section 8.

## 2 Related work

Deep learning has recently found a great deal of success in particle physics [1, 16]. Deep Neural Networks (DNNs) are artificial neural networks with multiple feed-forward hidden layers, each of which takes input features and produces a more abstract and composite representation as an output. Networks of this kind have produced progresses in many fields, including computer vision and natural language processing. Driving the innovation in these fields are increasingly complex architectures that are well-suited to a particular domain, including convolutional neural networks (CNNs) [17–19], recurrent neural networks (RNNs) [20, 21], long short-term memory units (LSTMs) [22], and gated recurrent units (GRUs) [23]. Jet tagging is one of the most popular LHC-related tasks to which DL solutions have been applied. Several classification algorithms have been studied in the context of jet tagging at the LHC [24–29] using CNNs, or physics-inspired DNN models [30–32]. Recurrent and recursive layers have been used to define jet classifiers starting from a list of reconstructed particle momenta [33–35]. Recently, these different approaches, applied to the specific case of top quark (t) jet identification have been compared [36]. Unsupervised methods have also been proposed, mainly to tag top jets or jets coming from new postulated new particles [37–39].

Graph networks have very recently been used for jet tagging, matching the performances of other deep learning approaches [40, 41, 13], for event classification [42, 43], charged particle tracking in a silicon detector [44], pileup subtraction at the LHC [45], and particle reconstruction in irregular calorimeters [46, 13, 41] as well as in the IceCube experiment [43].

Particles, distributed sensor networks and power grids are examples of problems that involve multiple entities with complex interactions. Graphs provide a natural representation for encoding such relational information. Traditional machine learning methods use feature engineering to learn from graphs, which is slow and costly. Graph representation learning,

including graph convolution networks [47–50] and graph generative models [51, 52], leverages deep learning graph representation to learn directly from structured data. In contrast to existing deep learning methods, graph representation learning can (1) handle irregular grids with non-Euclidean geometry [53], (2) encode physics knowledge via graph construction [54], and (3) introduce relational inductive bias into data-driven learning systems [55]. Convolutional neural networks are powerful classifiers that work extremely well for images [56, 57], where data are represented on a pixel grid. However, in many scientific problems the data itself is not Euclidean. Geometric deep learning algorithms, such as graph neural networks [58, 59], that are invariant to the underlying grid structure, emerge as a more optimal choice for such data.

## 3 Data samples

The CMS Open Data are available from the CERN Open Data Portal [14], including releases of 2010, 2011, and 2012 CMS collision data as well as 2011, 2012, and 2016 CMS simulated data.

Samples of $\mathrm{H} \to \mathrm{b\bar{b}}$ jets are available from simulated events containing Randall-Sundrum gravitons [60] decaying to two Higgs bosons, which subsequently decay to $\mathrm{b\bar{b}}$ pairs. The event generation was done with MADGRAPH5_aMCATNLO 2.2.2 at leading order, with graviton masses ranging between 0.6 and 4.5 TeV. The main source of background originates from multijet events. The background data set was generated with PYTHIA 8.205 [61] in different bins of the average $p_\mathrm{T}$ of the final-state partons ($\hat{p}_\mathrm{T}$). The parton showering and hadronization was performed with PYTHIA 8.205 [61], using the CMS underlying event tune CUETP8M1 [62] and the NNPDF 2.3 [63] parton distribution functions. Pileup interactions are modelled by overlaying each simulated event with additional minimum bias collisions, also generated with PYTHIA 8.205.

The outcome of the default CMS reconstruction workflow is provided in the Open Data release [64]. In particular, particle candidates are reconstructed using the particle-flow (PF) algorithm [65]. Charged particles from pileup interactions are removed using the CHS algorithm. Jets are clustered from the remaining reconstructed particles using the anti-$k_\mathrm{T}$ algorithm [66, 67] with a jet-size parameter $R = 0.8$ (AK8 jets). 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 [68, 8] is applied, with angular exponent $\beta = 0$, soft cutoff threshold $z_\mathrm{cut} < 0.1$, and characteristic radius $R_0 = 0.8$ [69]. The soft-drop mass ($m_\mathrm{SD}$) is then computed from the four-momenta of the remaining constituents.

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

The data set is reduced by requiring the AK8 jets to have $300 < p_\mathrm{T} < 2400\,\mathrm{GeV}$, $|\eta| < 2.4$, and $40 < m_\mathrm{SD} < 200\,\mathrm{GeV}$. Charged particles are required to have $p_\mathrm{T} > 0.95\,\mathrm{GeV}$ and reconstructed secondary vertices (SVs) are associated with the AK8 jet using $\Delta R = \sqrt{\Delta\phi^2 + \Delta\eta^2} < 0.8$. The data set is divided in 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 Tab. 2 in App. B). The IN also uses 14 SV features listed in Tab. 3. The DDB tagger [5] 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 Tab. 4 and first used in a previous version of the algorithm, described in Ref. [4]. For both the IN and the DDB tagger, 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.

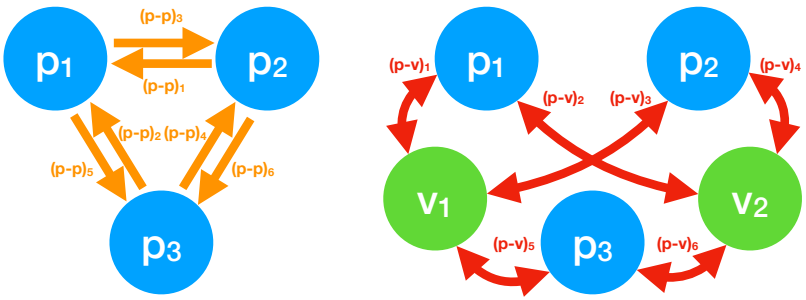


Figure 2: Two example graphs with 3 particles and 2 vertices and the corresponding edges.

## 4 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 $S$. The input consists of an ensemble of $X$ and $Y$ matrices, with sizes $P \times N_p$ and $P \times N_v$, respectively.

A particle graph $\mathcal{G}_p$ is constructed by connecting each particle to each 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 particle to each vertex through $N_{pv} = N_p N_v$ undirected edges. This is pictorially represented in Fig. 2 for the case of a three particles and two vertices. As shown in the figure, the graph nodes and edges are arbitrary 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 $i^{\text{th}}$ particle receives the $j^{\text{th}}$ edge and is 0 otherwise. Similarly, the element $(R_S)_{ij}$ is set to 1 when the $i^{\text{th}}$ particle sends the $j^{\text{th}}$ 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:

$$R_R = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \end{array} \begin{array}{c} \begin{array}{cccccc} (p-p)_1 & (p-p)_2 & (p-p)_3 & (p-p)_4 & (p-p)_5 & (p-p)_6 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \end{array} \tag{1}$$

$$R_S = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \end{array} \begin{array}{c} \begin{array}{cccccc} (p-p)_1 & (p-p)_2 & (p-p)_3 & (p-p)_4 & (p-p)_5 & (p-p)_6 \end{array} \\ \begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array} \tag{2}$$

$$R_K = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \end{array} \begin{array}{c} \begin{array}{cccccc} (p-v)_1 & (p-v)_2 & (p-v)_3 & (p-v)_4 & (p-v)_5 & (p-v)_6 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \end{array} \tag{3}$$

$$R_V = \begin{array}{c} \\ v_1 \\ v_2 \end{array} \begin{array}{c} \begin{array}{cccccc} (p-v)_1 & (p-v)_2 & (p-v)_3 & (p-v)_4 & (p-v)_5 & (p-v)_6 \end{array} \\ \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \end{array}. \tag{4}$$

The data flow of our 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 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 $\overline{E}_{pp} = E_{pp} R_R^\top$ with dimension $D_E \times N_p$. We also build $\overline{E}_{vp} = E_{vp} R_V^\top$ 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 $(P + 2D_E) \times N_O$, by combining the input information for each particle ($X$) with the learned representation of the particle-particle ($\overline{E}_{pp}$) and particle-vertex ($\overline{E}_{vp}$) interactions:

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

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 $\overline{E}_{pp}$ and $\overline{E}_{vp}$ 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, represented 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 $\overline{O}$ 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 data simulation with 2016 conditions. The input data set consists of 3.9 million $H \to b\bar{b}$ jets and 1.9 million inclusive QCD jets, split into training, validation, and test samples with proportions of 80%, 10%, and 10%, respectively.

We use PyTorch [70] to implement and train the classifier on one GTX 1080 GPU [1]. 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 ReLU activation function after each layer. The function $f_O$ is a dense sequence of sizes $(60, 30, 24)$ in a similar fashion. We use up to $N_p = 60$ charged

[1] We also convert the interaction network into a TensorFlow model, as discussed in App. A.

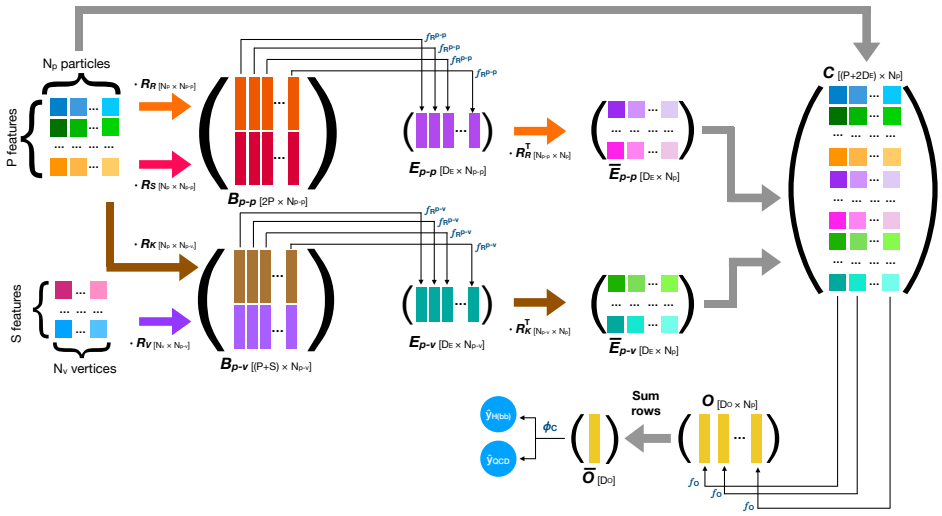


Figure 3: Illustration of the IN classifier. The particle feature matrix $X$ is multiplied by the receiving and sending matrices $R_R$ and $R_S$ to build the particle-particle interaction feature matrix $B_{pp}$. Similarly, the particle feature matrix $X$ and the vertex feature matrix $Y$ are multiplied 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 post-interaction 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 post-interaction 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 $\phi_C$, also represented by a neural network. More details on the various steps are given in the text.

particles and $N_v = 5$ secondary vertices as inputs to the IN tagger. We train the model using the Adam optimizer [71] with an initial learning rate of $10^{-4}$ and a minibatch size of 128 for up to 100 epochs, enforcing early stopping [72] on the validation loss with a patience of 5 epochs.

As a baseline, we minimize the categorical cross-entropy loss for the classification task $L_C$ and we let the network exploit all the discriminating information in the data set.

## 5 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 our case), one could perform an adversarial training [73–75]. One could also reweight or remove background events such that the background $m_{\mathrm{SD}}$ distribution is indistinguishable from the signal $m_{\mathrm{SD}}$ distribution. Finally, one could also define a mass-dependent threshold based on simulation as in the "designing decorrelated taggers" (DDT) procedure proposed in Ref. [76]. We present results for the latter three approaches.

### 5.1 Adversarial training

A 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 post-interaction feature vector $\overline{\mathbf{O}}$ as its input. From this input, the adversary is trained to predict a one-hot encoding of the pivot feature $m_{\mathrm{SD}}$, which we aim to decorrelate from the classifier output. The chosen one-hot encoding corresponds to 40 $m_{\mathrm{SD}}$ bins from 40 to 200 GeV. The training begins from by initializing the weights from the best classifier training. The adversary is then pre-trained 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

$$L = L_{\mathrm{C}} - \lambda L_{\mathrm{adversary}}. \tag{8}$$

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

### 5.2 Sample reweighting

While adversarial training requires a complicate 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 $N_{\mathrm{bkg}}^{\mathrm{bin}}$ and $N_{\mathrm{sig}}^{\mathrm{bin}}$, respectively, the event is weighted by $w_{\mathrm{bin}} = N_{\mathrm{sig}}^{\mathrm{bin}}/N_{\mathrm{bkg}}^{\mathrm{bin}}$.

### 5.3 Designing decorrelated taggers

Following the DDT procedure [76], the tagger threshold for a given FPR or "working point" is determined as a function of $m_{\mathrm{SD}}$. By creating a $m_{\mathrm{SD}}$-dependent tagger threshold, the background jet $m_{\mathrm{SD}}$ distribution for events passing and failing this threshold can be made identical. In practice, this is done considering the distribution of the network score vs. the jet $p_{\mathrm{T}}$ and $m_{\mathrm{SD}}$ for the training dataset. A quantile regression was used to find the threshold on the network score as a function of $p_{\mathrm{T}}$ and $m_{\mathrm{SD}}$ distribution that would correspond to a fixed quantile (the chosen 1-FPR value). By construction, this procedure results in near-perfect mass-decorrelation.

In our case, a gradient boosted regressor [77, 78] with the following parameters was used:

- $\alpha$-quantile = 1 − FPR,
- number of estimators of 250,
- minimum number of samples at a leaf node of 3,
- minimum number of samples to split an internal node of 3,
- maximum depth of 5,
- validation set of 20%,
- early stopping with tolerance = 5.

## 6 Deep double-b tagger model

The DDB tagger is the deep neural network algorithm currently in use by the CMS collaboration to identify $\mathrm{H} \to \mathrm{b\overline{b}}$ jets. Since this tagger is trained on a dataset similar to the one considered for this study, we adopt it as a proxy for a typical state-of-the-art algorithm. The DDB model is based on the 27 HLFs used in Ref. [4], 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 App. B). Each block of inputs is treated as a one-dimensional list, with batch normalization [79] applied directly to the input layers. For each collection of charged particles and SVs, separate 1D convolutional layers [80], with a kernel size of 1, are trained: 2 hidden layers with 32 filters each and ReLU [81] activation. The filters act on each particle or vertex individually. The compressed and transformed outputs are then separately fed into two gated recurrent units (GRUs) with 50 output nodes each and ReLU activation. The outputs of the GRUs are concatenated with the HLFs and then processed by a dense layer with 100

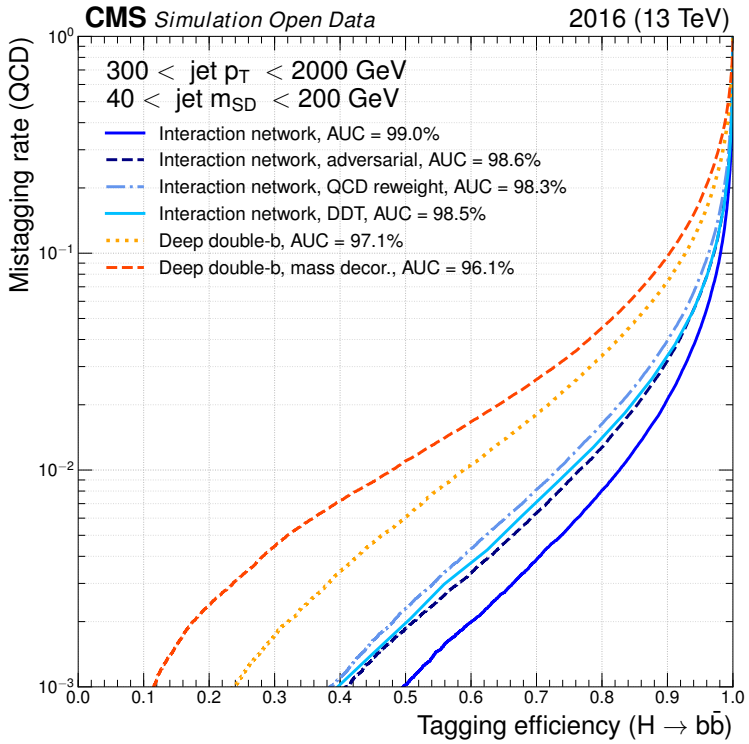


Figure 4: Performance of interaction network after applying adversarial training used to decrease the degree to with the interaction network is dependent on the mass of the jet. This results in a lower performance because the algorithm is forced to decorrelate jet mass.These results are compared with the ROC curves for the deep double-b tagger, with and without a mass sculpting technique using Kullback–Leibler divergence.

nodes and ReLU activation, and another final dense layer with 2 output nodes with softmax activation. Dropout [82] (with a rate of 10%) is used in each layer to prevent overfitting.

The data set used for training consists of CMS simulation of $H \to b\bar{b}$ and corresponding to 2017 data-taking conditions. The training was performed with Keras [83] over 100 epochs with a batch size of 4096 using the Adam optimizer [71]. In order to decorrelate the tagger output from the jet mass, the network was trained for an additional 20 epochs, with a custom loss function penalty term which penalizes the mass sculpting. Namely, the Kullback-Liebler (KL) divergence,

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_i P_i \log \frac{P_i}{Q_i}, \tag{9}$$

is used between two $m_{\mathrm{SD}}$ distributions: one weighted by the network's output probability for signal and the other weighted by the network's output probability for background. The KL divergence is computed for the distributions for true signal events $P_{\mathrm{sig}}(m_{\mathrm{SD}})$ and true background events $P_B(m_{\mathrm{SD}})$, separately. The total loss function is then

$$L = L_{\mathrm{C}} + \lambda D_{\mathrm{KL}} \left( P_{\mathrm{bkg}}^{\text{sig weighted}}(m_{\mathrm{SD}}) \parallel P_{\mathrm{bkg}}^{\text{bkg weighted}}(m_{\mathrm{SD}}) \right) \tag{10}$$

$$+ \lambda D_{\mathrm{KL}} \left( P_{\mathrm{sig}}^{\text{sig weighted}}(m_{\mathrm{SD}}) \parallel P_{\mathrm{sig}}^{\text{bkg weighted}}(m_{\mathrm{SD}}) \right), \tag{11}$$

where $\lambda = 2$ was chosen.

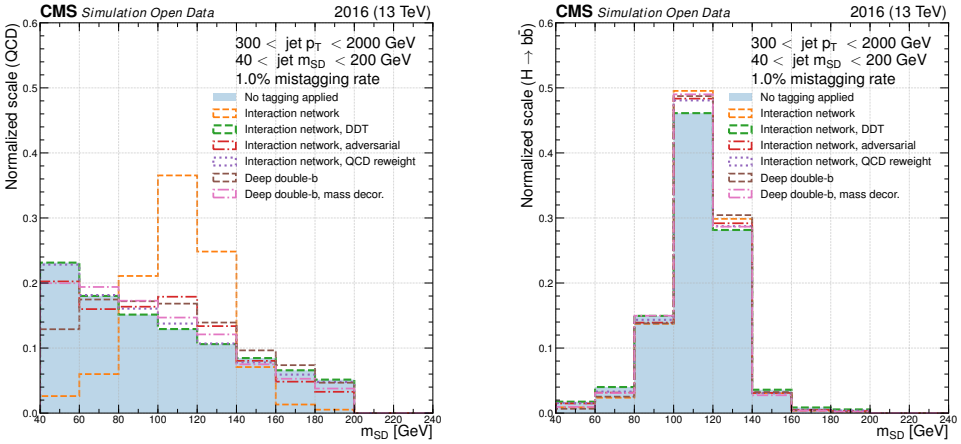


Figure 5: An illustration of the "sculpting" of the background jet mass distribution (left) and the signal jet mass distribution (right) after applying a threshold on the tagger score corresponding to a 1% false positive rate for several different algorithms. The unmodified interaction network is highly 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.

The DDB model was trained using CMS simulated events with 2017 running conditions. We verified that a similar tagging performance is obtained when the training procedure is repeated on the 2016 data set considered for this study.

## 7 Results

As shown in Fig. 4, the IN provides an improved performance with respect to the DDB tagger. At 1% FPR, the IN tagger outperforms the DDB tagger by 40% in true positive rate (TPR). Likewise, at 80% TPR, the IN tagger yields a factor of 4 smaller false positive rate (FPR) than the the DDB tagger.

Fig. 5 shows an illustration of how the signal and background jet mass distributions change after applying a threshold on the tagger score for different decorrelation procedures. Following Ref. [84], we quantify the impacts of these algorithms on the mass decorrelation by computing the Jensen-Shannon (JS) divergence:

$$D_{\mathrm{JS}}(P \parallel Q) = \frac{1}{2} D_{\mathrm{KL}}(P \parallel M) + \frac{1}{2} D_{\mathrm{KL}}(Q \parallel M) \ , \tag{12}$$

where $M = \frac{1}{2}(P + Q)$ is the average of the normalized $m_{\mathrm{SD}}$ distributions of the background jets passing ($P$) and failing ($Q$) a given tagger score. As shown in Fig. 6, the DDT procedure provides the best decorrelation of the IN tagger followed by the reweighted training and the adversarial training, respectively.

After applying the mass decorrelation techniques, the performance worsens slightly but still significantly outperforms the DDB taggers, as shown in Fig. 4. At 1% FPR, the DDT-decorrelated IN tagger has a TPR of 76% compared to the decorrelated DDB tagger with a 48% TPR, corresponding to an improvement of 55%. Table 1 summarizes different performance metrics for the four considered models.

In addition, we show the performance of the proposed algorithm as a function of the number of primary vertices in the event (see Fig. 7), scaling linearly with the number pileup. Using only charged particles and secondary vertices as input, the IN is also robust against an increasing number of pileup interactions.

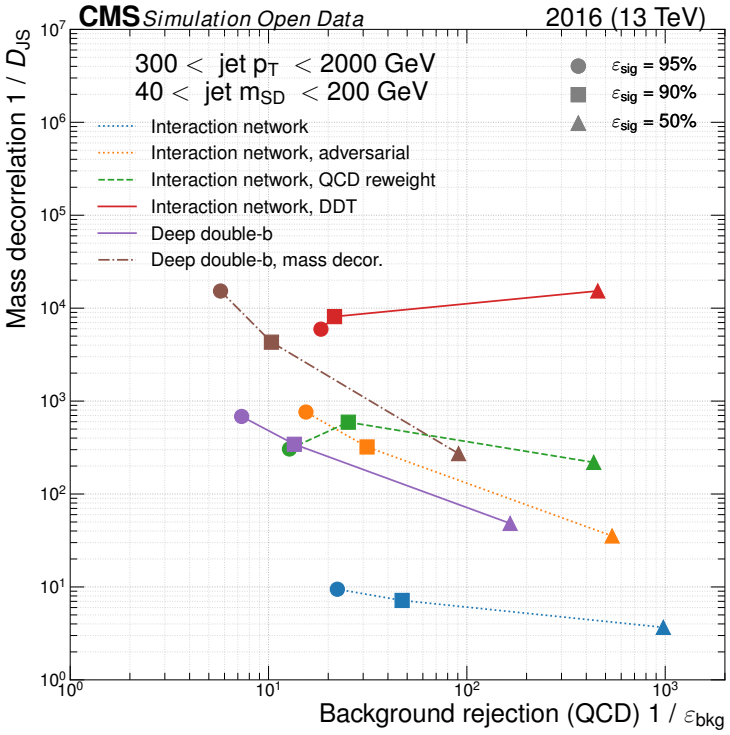


Figure 6: Mass decorrelation metric as a function of background rejection for the IN and DDB taggers in different training configurations. The decorrelation is quantified as the inverse of the Jensen-Shannon divergence between the background mass distribution passing and failing a given threshold cut on the classifier score. The background rejection is quantified as the inverse of the FPR, while the signal efficiency is equal to the TPR.

| Model | Accuracy | AUC | $1/\varepsilon_{bkg}$ @ $\varepsilon_{sig} = 30\%$ | $\varepsilon_{sig}$ @ $\varepsilon_{bkg} = 1\%$ | $1/D_{JS}$ @ $\varepsilon_{bkg} = 1\%$ |
|---|---|---|---|---|---|
| Interaction network | **95.5%** | **99.0%** | **4160.3** | **82.7%** | 5.2 |
| Deep double-b | 91.1% | 97.0% | 578.0 | 58.9% | **64.8** |
| Interaction network, adversarial | **94.6%** | **98.6%** | 2381.0 | **76.5%** | 124.6 |
| Interaction network, QCD reweight | 93.7% | 98.3% | 1864.9 | 73.2% | 22.6 |
| Interaction network, DDT | – | 98.5% | **8055.0** | 75.3% | **2635.7** |
| Deep double-b, mass decor. | 88.4% | 96.1% | 224.6 | 47.9% | 260.1 |

Table 1: Performance metrics of the different models, including accuracy, area under the ROC curve, background rejection at a true positive rate of 30%, and true positive rate and mass decorrelation metric $1/D_{JS}$ at a false positive rate of 1%.

## 8 Conclusions

We presented a novel jet-tagging technique using a graph representation of the jet's constituents and secondary vertices based on an interaction network to identify $H \to b\bar{b}$ jets 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 a simulation dataset released by the CMS collaboration in the CERN Open Data Portal. A significantly performance improvement is observed with respect to the corresponding Deep Neural Network currently used in the CMS collaboration (the DDB tagger). By design, our interaction network offers a more flexible representations

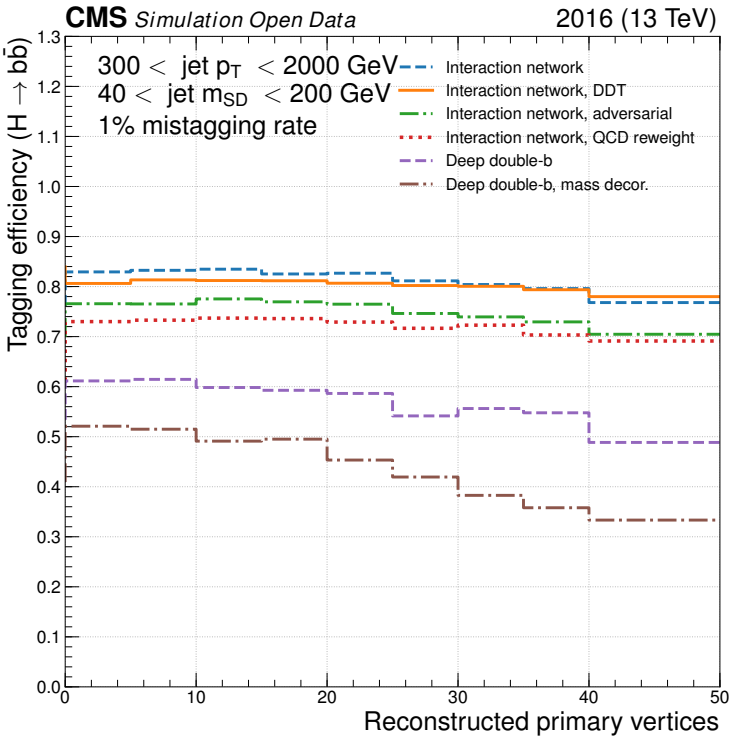


Figure 7: True positive rate of the IN tagger as a function of the number of reconstructed primary vertices for different FPR values.

of jet data and a robustness against the noise generated by pileup collisions. The algorithm implementation and its training code are available at Ref. [85].

Together with the best trained model, 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 this algorithm suitable for physics analyses relying on the jet mass as a discrimination variable. As expected, the three decorrelation procedures result in a reduction of the $H \to b\bar{b}$ identification performance. Nevertheless, the three decorrelated models outperform the best DDB referenced model.

Once applied to a full data analysis, our tagging algorithm could contribute a substantial improvement to the experimental precision. Our 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 n$^o$ 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.

## References

[1] 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*, 1709.04464.

[2] L. Asquith et al., *Jet Substructure at the Large Hadron Collider : Experimental Review*, 1803.06991.

[3] 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*, Tech. Rep. CERN-EP-2019-107. CMS-B2G-18-002-003, CERN, Geneva, Jun, 2019.

[4] CMS collaboration, *Identification of heavy-flavour jets with the CMS detector in pp collisions at 13 TeV*, *JINST* **13** (2018) P05011 [1712.07158].

[5] 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*, Jul, 2018.

[6] CMS collaboration, *Boosted jet identification using particle candidates and deep neural networks*, Nov, 2017.

[7] ATLAS collaboration, "Identification of boosted Higgs bosons decaying into $b$-quark pairs with the ATLAS detector at 13 TeV." 2019.

[8] 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** (2008) 242001 [0802.2470].

[9] K. Becker, F. Caola, A. Massironi, B. Mistlberger, P. Monni, X. Chen et al., *Recommended predictions for the boosted-Higgs cross section*, Tech. Rep. LHCHXSWG-2019-002, CERN, Geneva, Mar, 2019.

[10] CMS collaboration, *Inclusive search for a highly boosted Higgs boson decaying to a bottom quark-antiquark pair*, *Phys. Rev. Lett.* **120** (2018) 071802 [1709.05543].

[11] ATLAS collaboration, T. A. 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*, 2018.

[12] P. W. Battaglia, R. Pascanu, M. Lai, D. Rezende and K. Kavukcuoglu, *Interaction Networks for Learning about Objects, Relations and Physics*, *ArXiv e-prints* (2016) [1612.00222].

[13] E. A. Moreno, O. Cerri, J. M. Duarte, H. B. Newman, T. Q. Nguyen, A. Periwal et al., *JEDI-net: a jet identification algorithm based on interaction networks*, 2019.

[14] "CERN Open Data Portal." http://opendata.cern.ch.

[15] CMS collaboration, *Jet performance in CMS*, *PoS* **EPS-HEP2013** (2013) 433.

[16] D. Guest, K. Cranmer and D. Whiteson, *Deep Learning and its Application to LHC Physics*, *Ann. Rev. Nucl. Part. Sci.* **68** (2018) 161 [1806.11484].

[17] Y. LeCun, Y. Bengio et al., *Convolutional networks for images, speech, and time series*, *The handbook of brain theory and neural networks* **3361** (1995) 1995.

[18] S. Lawrence, C. L. Giles, A. C. Tsoi and A. D. Back, *Face recognition: A convolutional neural-network approach*, *IEEE transactions on neural networks* **8** (1997) 98.

[19] A. Krizhevsky, I. Sutskever and G. E. Hinton, *Imagenet classification with deep convolutional neural networks*, in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

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

[21] A. Graves, A.-r. Mohamed and G. Hinton, *Speech recognition with deep recurrent neural networks*, in *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pp. 6645–6649, IEEE, 2013.

[22] S. Hochreiter and J. Schmidhuber, *Long short-term memory*, *Neural computation* **9** (1997) 1735.

[23] J. Chung, C. Gulcehre, K. Cho and Y. Bengio, *Empirical evaluation of gated recurrent neural networks on sequence modeling*, *arXiv preprint arXiv:1412.3555* (2014) .

[24] L. de Oliveira, M. Kagan, L. Mackey, B. Nachman and A. Schwartzman, *Jet-images – deep learning edition*, *JHEP* **07** (2016) 069 [1511.05190].

[25] 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** (2016) 112002 [1607.08633].

[26] S. Macaluso and D. Shih, *Pulling out all the tops with computer vision and deep learning*, *JHEP* **10** (2018) 121 [1803.00107].

[27] G. Kasieczka, T. Plehn, M. Russell and T. Schell, *Deep-learning top taggers or the end of QCD?*, *JHEP* **05** (2017) 006 [1701.08784].

[28] P. T. Komiske, E. M. Metodiev and M. D. Schwartz, *Deep learning in color: towards automated quark/gluon jet discrimination*, *JHEP* **01** (2017) 110 [1612.01551].

[29] A. Schwartzman, M. Kagan, L. Mackey, B. Nachman and L. De Oliveira, *Image Processing, Computer Vision, and Deep Learning: new approaches to the analysis and physics interpretation of LHC events*, *J. Phys. Conf. Ser.* **762** (2016) 012035.

[30] K. Datta and A. J. Larkoski, *Novel jet observables from machine learning*, *JHEP* **03** (2018) 086 [1710.01305].

[31] A. Butter, G. Kasieczka, T. Plehn and M. Russell, *Deep-learned top tagging with a Lorentz layer*, *SciPost Phys.* **5** (2018) 028 [1707.08966].

[32] P. T. Komiske, E. M. Metodiev and J. Thaler, *Energy flow polynomials: A complete linear basis for jet substructure*, *JHEP* **04** (2018) 013 [1712.07124].

[33] G. Louppe, K. Cho, C. Becot and K. Cranmer, *QCD-aware recursive neural networks for jet physics*, 1702.00748.

[34] S. Egan, W. Fedorko, A. Lister, J. Pearkes and C. Gay, *Long Short-Term Memory (LSTM) networks with jet constituents for boosted top tagging at the LHC*, 1711.09059.

[35] T. Cheng, *Recursive neural networks in quark/gluon tagging*, *Comput. Softw. Big Sci.* **2** (2018) 3 [1711.02633].

[36] A. Butter et al., *The Machine Learning Landscape of Top Taggers*, 1902.09914.

[37] T. Heimel, G. Kasieczka, T. Plehn and J. M. Thompson, *QCD or What?*, *SciPost Phys.* **6** (2019) 030 [1808.08979].

[38] M. Farina, Y. Nakai and D. Shih, *Searching for New Physics with Deep Autoencoders*, 1808.08992.

[39] B. M. Dillon, D. A. Faroughy and J. F. Kamenik, *Uncovering latent jet substructure*, *Phys. Rev.* **D100** (2019) 056002 [1904.04200].

[40] I. Henrion, J. Brehmer, J. Bruna, K. Cho, K. Cranmer, G. Louppe et al., *Neural message passing for jet physics*, .

[41] H. Qu and L. Gouskos, *ParticleNet: Jet Tagging via Particle Clouds*, 1902.08570.

[42] M. Abdughani, J. Ren, L. Wu and J. M. Yang, *Probing stop with graph neural network at the LHC*, 1807.09088.

[43] N. Choma et al., *Graph neural networks for icecube signal classification*, *CoRR* **abs/1809.06166** (2018) .

[44] S. Farrell et al., *Novel deep learning methods for track reconstruction*, in *4th International Workshop Connecting The Dots 2018 (CTD2018) Seattle, Washington, USA, March 20-22, 2018*, 2018, 1810.06111.

[45] J. Arjona Martínez, O. Cerri, M. Pierini, M. Spiropulu and J.-R. Vlimant, *Pileup mitigation at the Large Hadron Collider with graph neural networks*, *Eur. Phys. J. Plus* **134** (2019) 333 [1810.07988].

[46] 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.* **C79** (2019) 608 [1902.07987].

[47] M. Niepert, M. Ahmed and K. Kutzkov, *Learning convolutional neural networks for graphs*, in *International conference on machine learning*, pp. 2014–2023, 2016.

[48] T. N. Kipf and M. Welling, *Semi-supervised classification with graph convolutional networks*, *arXiv preprint arXiv:1609.02907* (2016) .

[49] C. R. Qi, H. Su, K. Mo and L. J. Guibas, *Pointnet: Deep learning on point sets for 3d classification and segmentation*, *CoRR* **abs/1612.00593** (2016) [1612.00593].

[50] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein and J. M. Solomon, *Dynamic graph CNN for learning on point clouds*, *CoRR* **abs/1801.07829** (2018) [1801.07829].

[51] A. Grover, A. Zweig and S. Ermon, *Graphite: Iterative generative modeling of graphs*, *arXiv preprint arXiv:1803.10459* (2018) .

[52] J. You, R. Ying, X. Ren, W. L. Hamilton and J. Leskovec, *Graphrnn: Generating realistic graphs with deep auto-regressive models*, *arXiv preprint arXiv:1802.08773* (2018) .

[53] J. Bruna, W. Zaremba, A. Szlam and Y. LeCun, *Spectral networks and locally connected networks on graphs*, *arXiv preprint arXiv:1312.6203* (2013) .

[54] D. Zheng, V. Luo, J. Wu and J. B. Tenenbaum, *Unsupervised learning of latent physical properties using perception-prediction networks*, *arXiv preprint arXiv:1807.09244* (2018) .

[55] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski et al., *Relational inductive biases, deep learning, and graph networks*, *arXiv preprint arXiv:1806.01261* (2018) .

[56] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, -, *Proceedings of the IEEE.* **11** (1998) .

[57] 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 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 770–778, 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016, 2016, DOI.

[58] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam and P. Vandergheynst, *Geometric deep learning: Going beyond euclidean data*, *IEEE Signal Processing Magazine* **34** (2017) 18.

[59] Y. Li, D. Tarlow, M. Brockschmidt and R. Zemel, *Gated graph sequence neural networks*, *arXiv e-prints* (2015) arXiv:1511.05493.

[60] L. Randall and R. Sundrum, *Large mass hierarchy from a small extra dimension*, *Phys. Rev. Lett.* **83** (1999) 3370 [hep-ph/9905221].

[61] T. Sjöstrand, S. Ask, J. R. Christiansen, R. Corke, N. Desai, P. Ilten et al., *An introduction to PYTHIA 8.2*, *Comput. Phys. Commun.* **191** (2015) 159 [1410.3012].

[62] CMS collaboration, *Event generator tunes obtained from underlying event and multiparton scattering measurements*, *Eur. Phys. J. C* **76** (2016) 155 [1512.00815].

[63] NNPDF collaboration, *Parton distributions with LHC data*, *Nucl. Phys. B* **867** (2013) 244 [1207.1303].

[64] CMS collaboration, J. Duarte, "Sample with jet, track and secondary vertex properties for hbb tagging ml studies HiggsToBBNTuple_HiggsToBB_QCD_RunII_13TeV_MC." 10.7483/OPENDATA.CMS.JGJX.MS7Q.

[65] CMS collaboration, *Particle-flow reconstruction and global event description with the cms detector*, *JINST* **12** (2017) P10003 [1706.04965].

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

[67] M. Cacciari, G. P. Salam and G. Soyez, *FastJet user manual*, *Eur. Phys. J. C* **72** (2012) 1896 [1111.6097].

[68] M. Dasgupta, A. Fregoso, S. Marzani and G. P. Salam, *Towards an understanding of jet substructure*, *JHEP* **09** (2013) 029 [1307.0007].

[69] A. J. Larkoski, S. Marzani, G. Soyez and J. Thaler, *Soft Drop*, *JHEP* **05** (2014) 146 [1402.2657].

[70] N. Ketkar, *Introduction to pytorch*, in *Deep Learning with Python*, pp. 195–208, Springer, (2017).

[71] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, *CoRR* **abs/1412.6980** (2014) [1412.6980].

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

[73] H. Ajakan, P. Germain, H. Larochelle, F. Laviolette and M. Marchand, *Domain-Adversarial Neural Networks*, *arXiv e-prints* (2014) arXiv:1412.4446 [1412.4446].

[74] G. Louppe, M. Kagan and K. Cranmer, *Learning to Pivot with Adversarial Networks*, 2016.

[75] C. Shimmin, P. Sadowski, P. Baldi, E. Weik, D. Whiteson, E. Goul et al., *Decorrelated Jet Substructure Tagging using Adversarial Neural Networks*, *Phys. Rev.* **D96** (2017) 074034 [1703.03507].

[76] J. Dolen, P. Harris, S. Marzani, S. Rappoccio and N. Tran, *Thinking outside the ROCs: Designing Decorrelated Taggers (DDT) for jet substructure*, *JHEP* **05** (2016) 156 [1603.00027].

[77] J. H. Friedman, *Stochastic gradient boosting*, *Computational statistics & data analysis* **38** (2002) 367.

[78] J. H. Friedman, *Greedy function approximation: a gradient boosting machine*, *Annals of statistics* (2001) 1189.

[79] S. Ioffe and C. Szegedy, *Batch normalization: Accelerating deep network training by reducing internal covariate shift*, *CoRR* **abs/1502.03167** (2015) [1502.03167].

[80] 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)*, pp. 2608–2611, Aug, 2015, DOI.

[81] A. F. Agarap, *Deep learning using rectified linear units (relu)*, 2018.

[82] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, *Dropout: A simple way to prevent neural networks from overfitting*, *Journal of Machine Learning Research* **15** (2014) 1929.

[83] F. Chollet et al., "Keras." https://keras.io, 2015.

[84] ATLAS collaboration, *Performance of mass-decorrelated jet substructure observables for hadronic two-body decay tagging in ATLAS*, Tech. Rep. ATL-PHYS-PUB-2018-014, CERN, Geneva, Jul, 2018.

[85] E. Moreno, J. Duarte and A. Periwal, "IN." v1.2 https://github.com/eric-moreno/IN, Sept., 2019. 10.5281/zenodo.3462611.

[86] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro et al., *Tensorflow: Large-scale machine learning on heterogeneous distributed systems*, 2015.

[87] CMS collaboration, *CMS Physics: Technical Design Report Volume 1: Detector Performance and Software*, .

[88] Open Neural Network Exchange collaboration, *ONNX*, 2017.

## A Model implemented in TensorFlow

In order to integrate the interaction network algorithm into experimental workflows, it is often necessary to provide the algorithm was converted to TensorFlow [86] For example, The CMS experimental software framework CMSSW [87] currently only supports TensorFlow models. To perform this conversion, we utilize ONNX [88] to convert PyTorch model into both an ONNX model and a TensorFlow model, available at Ref. [85].

## B Data set features

The charged particle features used by the interaction network and deep deep-double-b taggers are listed in Tab. 2. The SV features used by both taggers are listed in Tab. 3, and the high-level features used by only the DDB tagger are shown in Tab. 4.

| Variable | Description |
|---|---|
| track_ptrel | $p_{\mathrm{T}}$ of the charged particle divided by the $p_{\mathrm{T}}$ 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 | $\Delta R$ between the charged particle and the AK8 jet axis |
| track_drminsv | $\Delta R$ between the associated SVs and the charged particle |
| track_drsubjet1 | $\Delta R$ between the charged particle and the first soft drop subjet |
| track_drsubjet2 | $\Delta R$ 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 paramater 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 paramater significance of the track |
| track_normchi2 | Normalized $\chi^2$ 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 ($p_{\mathrm{T}}$, $p_{\mathrm{T}}$) |
| track_detadeta | Track covariance matrix entry ($\eta$, $\eta$) |
| track_dphidphi | Track covariance matrix entry ($\phi$, $\phi$) |
| 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 ($d_\phi$, $d_z$) |
| track_dlambdadz | Track covariance matrix entry ($\lambda$, $d_z$) |
| **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 paramater of the track |
| **trackBTag_Sip2dSig** | Transverse (2D) signed impact paramater significance of the track |
| **trackBTag_Sip3dVal** | 3D signed impact paramater of the track |
| **trackBTag_Sip3dSig** | 3D signed impact paramater significance of the track |
| **trackBTag_JetDistVal** | Minimum track approach distance to the AK8 jet axis |

Table 2: Charged particle features. The interaction network uses all of the features, while DDB algorithm uses the subset of features indicated in bold.

| Variable | Description |
|---|---|
| sv_ptrel | $p_{\mathrm{T}}$ of the SV divided by the $p_{\mathrm{T}}$ 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 | $\Delta R$ between the SV and the AK8 jet axis |
| sv_pt | $p_{\mathrm{T}}$ of the SV |
| sv_mass | Mass of the SV |
| sv_ntracks | Number of tracks associated with the SV |
| sv_normchi2 | Normalized $\chi^2$ 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 3: Secondary vertex features. The interaction network uses all of the features, while DDB algorithm uses the subset of features indicated in bold.

| 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 | $\Delta R$ 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 |
| 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. [4] |

Table 4: High-level features used by the DDB algorithm.