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 ($p_T$). 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 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-$p_T$ 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-$p_T$ jet. This feature is particularly important for tagging a Higgs boson decaying to a bottom quark-antiquark pair ($H \rightarrow bb$) 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-$p_T$ $H \rightarrow bb$ decays [3][7]. This is an important task in particle physics because measurements of high-$p_T$ $H \rightarrow bb$ 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 $t\bar{t}H$ process [10][11].

In this work, we propose to identify $H \rightarrow bb$ 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 $H \rightarrow b\bar{b}$ 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 ($H \rightarrow b\bar{b}$)
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 $H \rightarrow b\bar{b}$ tagging algorithm in CMS, the deep
double-b (DDB) tagger [4].

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 [40–43], 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 \cite{kipf2016semi} and graph generative models \cite{brock2018large}, 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 \cite{rony2018graph}, (2) encode physics knowledge via graph construction \cite{pmlr-v97-luo19a}, and (3) introduce relational inductive bias into data-driven learning systems \cite{kipf2016semi}. Convolutional neural networks are powerful classifiers that work extremely well for images \cite{lecun2015deep}, 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 \cite{velivckovic2017graph}, that are invariant to the underlying grid structure, emerge as a more optimal choice for such data.

3 Data samples


Samples of $H \rightarrow b\bar{b}$ jets are available from simulated events containing Randall-Sundrum gravitons \cite{Randall:1999ee} decaying to two Higgs bosons, which subsequently decay to $b\bar{b}$ pairs. The event generation was done with MADGRAPH5_aMC@NLO 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 \cite{Sjostrand:2007gs}, using the CMS underlying event tune CUETP8M1 \cite{Khachatryan:2015pea} and the NNPDF 2.3 \cite{Ball:2014uwa} 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 \cite{cmsopendata}. In particular, particle candidates are reconstructed using the particle-flow (PF) algorithm \cite{Cacciari:2011ma}. 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 \cite{Cacciari:2008gp} 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 \cite{collins2016soft} is applied, with angular exponent $\beta = 0$, soft cutoff threshold $z_{cut} < 0.1$, and characteristic radius $R_0 = 0.8$ \cite{collins2015single}. The soft-drop mass ($m_{SD}$) is then computed from the four-momenta of the remaining constituents.

A signal $H \rightarrow b\bar{b}$ 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 $H \rightarrow b\bar{b}$ candidates.

The data set is reduced by requiring the AK8 jets to have $300 < p_T < 2400$ GeV, $|\eta| < 2.4$, and $40 < m_{SD} < 200$ GeV. 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 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 \cite{buican2017data} 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. \cite{buican2017data}. 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.
The data flow of our IN model is pictorially represented in Fig. 3. The input processing starts by creating the particle-particle interaction matrix $B_{pp}$ and the $(P + S) \times N_{vp}$ particle-vertex interaction matrices.

For the graph $G_p$, 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 $i^{th}$ particle receives the $j^{th}$ edge and is 0 otherwise. Similarly, the element $(R_S)_{ij}$ is set to 1 when the $i^{th}$ particle sends the $j^{th}$ 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:

$$R_R = \begin{pmatrix}
(p-p)_1 & (p-p)_2 & (p-p)_3 & (p-p)_4 & (p-p)_5 & (p-p)_6 \\
1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1
\end{pmatrix}$$

(1)

$$R_S = \begin{pmatrix}
(p-p)_1 & (p-p)_2 & (p-p)_3 & (p-p)_4 & (p-p)_5 & (p-p)_6 \\
0 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 0
\end{pmatrix}$$

(2)

$$R_K = \begin{pmatrix}
(p-v)_1 & (p-v)_2 & (p-v)_3 & (p-v)_4 & (p-v)_5 & (p-v)_6 \\
1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1
\end{pmatrix}$$

(3)

$$R_V = \begin{pmatrix}
(p-v)_1 & (p-v)_2 & (p-v)_3 & (p-v)_4 & (p-v)_5 & (p-v)_6 \\
1 & 0 & 1 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 & 0 & 1
\end{pmatrix}$$

(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 matrices.
particle-vertex interaction matrix \( B_{vp} \) defined as:

\[
B_{pp} = \begin{pmatrix} X \cdot R_R \\ X \cdot R_S \end{pmatrix},
\]

\[
B_{vp} = \begin{pmatrix} X \cdot R_K \\ Y \cdot R_V \end{pmatrix},
\]

where \( \cdot \) indicates the ordinary matrix product. Each column of \( B_{pp} \) consists of the 2\( P \) 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 \( f_{pp}^R \), one builds an internal representation of the particle-particle interaction with a function \( f_{pp}^R : \mathbb{R}^{2P} \rightarrow \mathbb{R}^{D_E} \), 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 \( f_{vp}^R : \mathbb{R}^{P + S} \rightarrow \mathbb{R}^{D_E} \).

We then propagate the particle-particle interactions back to the particles receiving them, by building \( E_{pp} = E_{pp} R^\top_R \) with dimension \( D_E \times N_p \). We also build \( E_{vp} = E_{vp} R^\top_V \) 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 \( (E_{pp}) \) and particle-vertex \( (E_{vp}) \) interactions:

\[
C = \begin{pmatrix} X \\ E_{pp} \\ E_{vp} \end{pmatrix}.
\]  

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 \( f_O : \mathbb{R}^{P + 2D_E} \rightarrow \mathbb{R}^{D_O} \), which computes the elements of the \( O \) matrix. The elements of the \( O \) matrix are computed by a function \( f_O : \mathbb{R}^{P + 2D_E} \rightarrow \mathbb{R}^{D_O} \), which returns the post-interaction representation for each of the input nodes. As is done for \( f_{pp}^R \) and \( f_{vp}^R \), \( f_O \) is applied to each column of \( C \).

We stress the fact that the by-column processing applied by the \( f_{pp}^R \), \( f_{vp}^R \), and \( f_O \) functions and the sum across interactions by defining the \( E_{pp} \) and \( E_{vp} \) 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 \( \bar{O} \) with length \( D_O \) for the jet as a whole. This is passed to a function \( \phi_C : \mathbb{R}^{D_O} \rightarrow \mathbb{R}^N \), 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 \( H \rightarrow b\bar{b} \) 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. The model is implemented with each of \( f_{pp}^R \) and \( f_{vp}^R \) 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...
We present results for the latter three approaches. As a baseline, we minimize the categorical cross-entropy loss for the classification task 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. One could also reweight or remove background events such that the background $m_{SD}$ distribution is indistinguishable from the signal $m_{SD}$ 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 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. One could also reweight or remove background events such that the background $m_{SD}$ distribution is indistinguishable from the signal $m_{SD}$ 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 $\overline{O}$ 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 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

$$L = L_C - \lambda L_{\text{adversary}}.$$  \hspace{1cm} (8)

Subsequently, the adversary is trained by minimizing $L_{\text{adversary}}$ using only the background QCD samples. To balance tagging performance and $m_{SD}$ 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 $N_{\text{bkg}}$ and $N_{\text{sig}}$, respectively, the event is weighted by $w_{\text{bin}} = N_{\text{bin}}^{\text{sig}} / N_{\text{bin}}^{\text{bkg}}$.

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 $m_{SD}$. 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 considering the distribution of the network score vs. the jet $p_T$ and $m_{SD}$ for the training dataset. A quantile regression was used to find the threshold on the network score as a function of $p_T$ and $m_{SD}$ distribution that would correspond to a fixed quantile (the chosen 1-FPR value). By construction, this procedure results in near-perfect mass-decorrelation.

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

- $\alpha$-quantile = $1 - \text{FPR}$,
- 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 $H \rightarrow b\bar{b}$ 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 which 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 → bb and corresponding to 2017 data-taking conditions. The training was performed with Keras [84] 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,

\[ D_{KL}(P \parallel Q) = \sum_i P_i \log \frac{P_i}{Q_i}, \] (9)

is used between two \( m_{SD} \) 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 \( P_{sig}(m_{SD}) \) and true background events \( P_B(m_{SD}) \), separately. The total loss function is then

\[ L = L_C + \lambda D_{KL}(P_{sig \text{ weighted}}(m_{SD}) \parallel P_{bkg \text{ weighted}}(m_{SD})) \] (10)

\[ + \lambda D_{KL}(P_{sig \text{ weighted}}(m_{SD}) \parallel P_{bkg \text{ weighted}}(m_{SD})) \] (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 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:

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

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. 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.

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%.

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

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 $H \rightarrow b\bar{b}$ 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.
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 $H \rightarrow b\bar{b}$ 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° 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


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


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


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


[64] CMS collaboration, J. Duarte, “Sample with jet, track and secondary vertex properties for hbb tagging ml studies HiggsToBBTuple_HiggsToBB_QCD_RunII_13TeV_MC” 10.7483/OPENDATA.CMS.JGJX.MS7Q


[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]

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].
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>track_ptrel</td>
<td>$p_T$ of the charged particle divided by the $p_T$ of the AK8 jet</td>
</tr>
<tr>
<td>track_erel</td>
<td>Energy of the charged particle divided by the energy of the AK8 jet</td>
</tr>
<tr>
<td>track_phirel</td>
<td>$\Delta\phi$ between the charged particle and the AK8 jet axis</td>
</tr>
<tr>
<td>track_etarel</td>
<td>$\Delta\eta$ between the charged particle and the AK8 jet axis</td>
</tr>
<tr>
<td>track_deltasR</td>
<td>$\Delta R$ between the charged particle and the AK8 jet axis</td>
</tr>
<tr>
<td>track_dminsv</td>
<td>$\Delta R$ between the associated SVs and the charged particle</td>
</tr>
<tr>
<td>track_drsubjett1</td>
<td>$\Delta R$ between the charged particle and the first soft drop subjet</td>
</tr>
<tr>
<td>track_drsubjett2</td>
<td>$\Delta R$ between the charged particle and the second soft drop subjet</td>
</tr>
<tr>
<td>track_dz</td>
<td>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</td>
</tr>
<tr>
<td>track_dxyz</td>
<td>Transverse (2D) impact parameter of the track</td>
</tr>
<tr>
<td>track_dxyzsig</td>
<td>Transverse (2D) impact parameter significance of the track</td>
</tr>
<tr>
<td>track_normchi2</td>
<td>Normalized $\chi^2$ of the track</td>
</tr>
<tr>
<td>track_quality</td>
<td>Track quality: undefQuality=-1, loose=0, tight=1, highPurity=2, confirmed=3, looseSetWithPV=5, highPuritySetWithPV=6, discarded=7, qualitySize=8</td>
</tr>
<tr>
<td>track_dptdpt</td>
<td>Track covariance matrix entry ($p_T$, $p_T$)</td>
</tr>
<tr>
<td>track_detaeta</td>
<td>Track covariance matrix entry ($\eta$, $\eta$)</td>
</tr>
<tr>
<td>track_dphiphi</td>
<td>Track covariance matrix entry ($\phi$, $\phi$)</td>
</tr>
<tr>
<td>track_dxydxy</td>
<td>Track covariance matrix entry ($d_{xy}$, $d_{xy}$)</td>
</tr>
<tr>
<td>track_dddz</td>
<td>Track covariance matrix entry ($d_\phi$, $d_\phi$)</td>
</tr>
<tr>
<td>track_dphidz</td>
<td>Track covariance matrix entry ($d_\phi$, $d_z$)</td>
</tr>
<tr>
<td>track_dlambdadz</td>
<td>Track covariance matrix entry ($\lambda$, $d_z$)</td>
</tr>
<tr>
<td>trackBTag_EtaRel</td>
<td>$\Delta \eta$ between the track and the AK8 jet axis, normalized to the track momentum</td>
</tr>
<tr>
<td>trackBTag_PParRatio</td>
<td>Component of track momentum perpendicular to the AK8 jet axis, normalized to the track momentum</td>
</tr>
<tr>
<td>trackBTag_PParRatio</td>
<td>Component of track momentum parallel to the AK8 jet axis, normalized to the track momentum</td>
</tr>
<tr>
<td>trackBTag_Sip2dVal</td>
<td>Transverse (2D) signed impact parameter of the track</td>
</tr>
<tr>
<td>trackBTag_Sip2dSig</td>
<td>Transverse (2D) signed impact parameter significance of the track</td>
</tr>
<tr>
<td>trackBTag_Sip3dVal</td>
<td>3D signed impact parameter of the track</td>
</tr>
<tr>
<td>trackBTag_Sip3dSig</td>
<td>3D signed impact parameter significance of the track</td>
</tr>
<tr>
<td>trackBTag_JetDistVal</td>
<td>Minimum track approach distance to the AK8 jet axis</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sv_ptrel</td>
<td>$p_T$ of the SV divided by the $p_T$ of the AK8 jet</td>
</tr>
<tr>
<td>sv_erel</td>
<td>Energy of the SV divided by the energy of the AK8 jet</td>
</tr>
<tr>
<td>sv_phirel</td>
<td>$\Delta\phi$ between the SV and the AK8 jet axis</td>
</tr>
<tr>
<td>sv_etarel</td>
<td>$\Delta\eta$ between the SV and the AK8 jet axis</td>
</tr>
<tr>
<td>sv_deltasR</td>
<td>$\Delta R$ between the SV and the AK8 jet axis</td>
</tr>
<tr>
<td>sv_pt</td>
<td>$p_T$ of the SV</td>
</tr>
<tr>
<td>sv_mass</td>
<td>Mass of the SV</td>
</tr>
<tr>
<td>sv_ntracks</td>
<td>Number of tracks associated with the SV</td>
</tr>
<tr>
<td>sv_normchi2</td>
<td>Normalized $\chi^2$ of the SV fit</td>
</tr>
<tr>
<td>sv_costhetasvpv</td>
<td>$\cos \theta$ between the SV and the PV</td>
</tr>
<tr>
<td>sv_dxy</td>
<td>Transverse (2D) flight distance of the SV</td>
</tr>
<tr>
<td>sv_dxyzsig</td>
<td>Transverse (2D) flight distance significance of the SV</td>
</tr>
<tr>
<td>sv_d3d</td>
<td>3D flight distance of the SV</td>
</tr>
<tr>
<td>sv_d3dsig</td>
<td>3D flight distance significance of the SV</td>
</tr>
</tbody>
</table>

Table 3: Secondary vertex features. The interaction network uses all of the features, while DDB algorithm uses the subset of features indicated in bold.
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fj_jetNTracks</td>
<td>Number of tracks associated with the AK8 jet</td>
</tr>
<tr>
<td>fj_nSV</td>
<td>Number of SVs associated with the AK8 jet ($\Delta R &lt; 0.7$)</td>
</tr>
<tr>
<td>fj_tau0_trackEtaRel_0</td>
<td>Smallest track $\Delta \eta$ relative to the jet axis,</td>
</tr>
<tr>
<td></td>
<td>associated to the first N-subjettiness axis</td>
</tr>
<tr>
<td>fj_tau0_trackEtaRel_1</td>
<td>Second smallest track $\Delta \eta$ relative to the jet axis,</td>
</tr>
<tr>
<td></td>
<td>associated to the first N-subjettiness axis</td>
</tr>
<tr>
<td>fj_tau0_trackEtaRel_2</td>
<td>Third smallest track $\Delta \eta$ relative to the jet axis,</td>
</tr>
<tr>
<td></td>
<td>associated to the first N-subjettiness axis</td>
</tr>
<tr>
<td>fj_taul_trackEtaRel_0</td>
<td>First smallest track $\Delta \eta$ relative to the jet axis,</td>
</tr>
<tr>
<td></td>
<td>associated to the second N-subjettiness axis</td>
</tr>
<tr>
<td>fj_taul_trackEtaRel_1</td>
<td>Second smallest track $\Delta \eta$ relative to the jet axis,</td>
</tr>
<tr>
<td></td>
<td>associated to the second N-subjettiness axis</td>
</tr>
<tr>
<td>fj_taul_trackEtaRel_2</td>
<td>Third smallest track $\Delta \eta$ relative to the jet axis,</td>
</tr>
<tr>
<td></td>
<td>associated to the second N-subjettiness axis</td>
</tr>
<tr>
<td>fj_tau_flightDistance2dSig_0</td>
<td>Transverse (2D) flight distance significance between the PV and the SV</td>
</tr>
<tr>
<td></td>
<td>with the smallest uncertainty on the 3D flight distance</td>
</tr>
<tr>
<td></td>
<td>associated to the first N-subjettiness axis</td>
</tr>
<tr>
<td>fj_tau_flightDistance2dSig_1</td>
<td>Transverse (2D) flight distance significance between the PV and the SV</td>
</tr>
<tr>
<td></td>
<td>with the smallest uncertainty on the 3D flight distance</td>
</tr>
<tr>
<td></td>
<td>associated to the second N-subjettiness axis</td>
</tr>
<tr>
<td>fj_tau_vertexDeltaR_0</td>
<td>$\Delta R$ between the first N-subjettiness axis and SV direction</td>
</tr>
<tr>
<td>fj_tau_vertexEnergyRatio_0</td>
<td>SV energy ratio for the first N-subjettiness axis, defined as the total</td>
</tr>
<tr>
<td></td>
<td>energy of all SVs associated with the first N-subjettiness axis divided by</td>
</tr>
<tr>
<td></td>
<td>the total energy of all the tracks associated with the AK8 jet that</td>
</tr>
<tr>
<td></td>
<td>are consistent with the PV</td>
</tr>
<tr>
<td>fj_tau_vertexEnergyRatio_1</td>
<td>SV energy ratio for the second N-subjettiness axis</td>
</tr>
<tr>
<td>fj_tau_vertexMass_0</td>
<td>SV mass for the first N-subjettiness axis, defined as the combined</td>
</tr>
<tr>
<td></td>
<td>invariant mass of all tracks from SVs associated with the first N-subjettiness</td>
</tr>
<tr>
<td></td>
<td>axis</td>
</tr>
<tr>
<td>fj_tau_vertexMass_1</td>
<td>SV mass for the second N-subjettiness axis</td>
</tr>
<tr>
<td>fj_trackSip2dSigAboveBottom_0</td>
<td>Track 2D signed impact parameter significance of the first track lifting</td>
</tr>
<tr>
<td></td>
<td>the combined invariant mass of the tracks above the b hadron threshold mass</td>
</tr>
<tr>
<td></td>
<td>(5.2 GeV)</td>
</tr>
<tr>
<td>fj_trackSip2dSigAboveBottom_1</td>
<td>Track 2D signed impact parameter significance of the second track lifting</td>
</tr>
<tr>
<td></td>
<td>the combined invariant mass of the tracks above the b hadron threshold mass</td>
</tr>
<tr>
<td></td>
<td>(5.2 GeV)</td>
</tr>
<tr>
<td>fj_trackSip2dSigAboveCharm_0</td>
<td>Track 2D signed impact parameter significance of the first track lifting</td>
</tr>
<tr>
<td></td>
<td>the combined invariant mass of the tracks above the c hadron threshold mass</td>
</tr>
<tr>
<td></td>
<td>(1.5 GeV)</td>
</tr>
<tr>
<td>fj_trackSipdSig_0</td>
<td>Largest track 3D signed impact parameter significance</td>
</tr>
<tr>
<td>fj_trackSipdSig_1</td>
<td>Second largest track 3D signed impact parameter significance</td>
</tr>
<tr>
<td>fj_trackSipdSig_2</td>
<td>Third largest track 3D signed impact parameter significance</td>
</tr>
<tr>
<td>fj_trackSipdSig_3</td>
<td>Fourth largest track 3D signed impact parameter significance</td>
</tr>
<tr>
<td>fj_trackSipdSig_0_0</td>
<td>Largest track 3D signed impact parameter significance</td>
</tr>
<tr>
<td></td>
<td>associated to the first N-subjettiness axis</td>
</tr>
<tr>
<td>fj_trackSipdSig_0_1</td>
<td>Largest track 3D signed impact parameter significance</td>
</tr>
<tr>
<td>fj_trackSipdSig_1_0</td>
<td>Largest track 3D signed impact parameter significance</td>
</tr>
<tr>
<td>fj_trackSipdSig_1_1</td>
<td>Second largest track 3D signed impact parameter significance</td>
</tr>
<tr>
<td>fj_z_ratio</td>
<td>$z$ ratio variable as defined in Ref. [4]</td>
</tr>
</tbody>
</table>

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