Prepared for submission to JHEP

Learning About Machine Learning With Di-Higgs Production at the LHC

B. Tannenwald[a] **C. Neu,**[a] **A. Li,**[a] **G. Buehlmann,**[a] **A. Cuddeback,**[a] **L. Hatfield,**[a] **R. Parvatam,**[a] **C. Thompson**[a]

[a] *University of Virginia, 248 McCormick Road, Charlottesville, VA, USA*

E-mail: benjamin.tannenwald@cern.sh

Abstract: Many domains of high energy physics analysis are starting to explore machine learning techniques. Powerful methods can be used to identify and measure rare processes from previously insurmountable backgrounds. One of the most profound Standard Model signatures still to be discovered at the LHC is the pair production of Higgs bosons through the Higgs self-coupling. The small cross section of this process makes detection very difficult even for the decay channel with the largest branching fraction ($hh \to b\bar{b}b\bar{b}$). This paper benchmarks a variety of approaches (boosted decision trees, various neural network architectures, semi-supervised algorithms) against one another to catalog the various techniques available to high energy physicists as the era of the HL-LHC approaches.

Contents

1 Introduction

The use of machine learning (ML) techniques in high energy particle physics has rapidly expanded since its first use at the Tevatron [1, 2]. The proliferation of methods and applications has touched nearly every segment of analysis and reconstruction [3] and will be vital in understanding the full dataset of the LHC and data from future colliders.

Common approaches involve using linear techniques like decision trees and non-linear approaches like neural networks. These techniques are then used to reconstruct objects like leptons and jets, to tag objects like b-quarks or boosted decays, and classify different processes. Many models rely on kinematic input features physicists have traditionally used; other architectures rely on emergent features produced in a more abstract phase-space. Regardless of the approach, the rise of so many different approaches raises interesting questions about what types of information are best to feed to our networks. The information that is best for physicists to learn from might not be optimal for sophisticated computing algorithms.

The goal of this paper is to explore a wide range of current ML techniques that can be used to identify di-Higgs production at the HL-LHC. Observing di-Higgs production is necessary to measure the self-coupling of the Higgs boson and fully understand the nature

of electroweak symmetry breaking. The difficulty in measuring Higgs pair production lies in the tiny cross-section of even the largest branching fraction ($hh \to b\bar{b}b\bar{b}$) and the relative abundance of similarly reconstructed QCD events. Section 2 deals with the physics relevant for di-Higgs production and the QCD background. Sections 3 and 4 summarize the various ML methods tested and Section 5 compares the results from the various approaches.

2 Di-Higgs Physics

2.1 Higgs Pair Production

The Higgs boson is an essential part of the Standard Model (SM) of particle physics and is a product of the mechanism responsible for electroweak symmetry breaking. Along with the interaction of the Higgs with the other particles of the Standard Model, the SM predicts the interaction of the Higgs boson with itself at tree-level (self-interaction). This mechanism contributes to non-resonant Higgs boson pair production together with quark-loop contributions via Yukawa-type interactions. Figure 1 shows a schematic diagram of non-resonant Higgs boson pair production. Since the production cross section for Higgs boson pair production is extremely small within the SM [4],

$$\sigma_{hh}\ (14\ \text{TeV}) = 45.1\ \text{fb},$$

any significant deviation would indicate the presence of new physics.

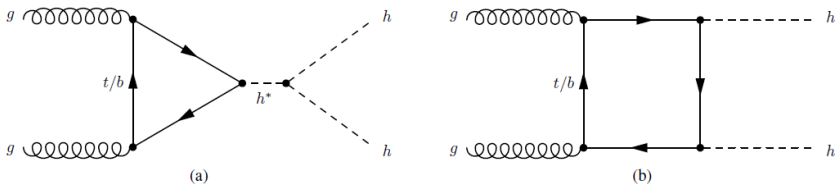


Figure 1. Leading order Feynman diagrams for non-resonant production of Higgs boson pairs in the Standard Model through (a) the Higgs boson self-coupling and (b) the Higgs-fermion Yukawa interaction.

Many extensions of the SM predict the existence of additional scalar bosons which may have mass larger than twice the Higgs mass and can decay into a Higgs boson pair. Searching for resonances in the hh mass spectrum can help us discover or limit exotic models which predict the presence of such particles. More importantly, measuring the SM di-Higgs cross-section (or placing limits on its magnitude) allow us to probe the self-coupling of the Higgs field and better understand the mechanism behind electroweak symmetry breaking.

The following work is focused on techniques for distinguishing non-resonant (SM-like) Higgs boson pair production where both Higgs bosons decay via $h \to b\bar{b}$. The choice of using the $4b$ decay mode provides the largest possible amount of signal events but requires powerful background reduction techniques due to the large production cross-section of fully hadronic QCD processes. All results are quoted for simulated events produced by pp collisions with a center-of-mass energy of 14 TeV and scaled to the full design luminosity

of the HL-LHC (a total integrated luminosity of 3000 fb^{-1}). Simulated samples were produced using ROOT v6.12/04 [5] and Madgraph v2.7.0 [6]. Events were then showered using Pythia v8.2.44 [7] and reconstructed with Delphes v3.0 [8] using the v2 approximation of the upgraded Phase-II CMS detector.

Both the signal and background samples were generated with minimal pileup addition sampled from a Poisson distribution with an expectation value of zero additional vertices. An additional generator-level cut requiring total hadronic energy greater than 300 GeV was applied when generating background QCD events. All code used to set up the generation environment and produce events is publicly available [9]. A summary of the sample generation details is shown in Table 1.

Name	Process	σ_{eff} [fb]	N_{events}
Di-Higgs	$pp \to hh,\ h \to b\bar{b}$	12.4	$1{\cdot}10^6$
QCD	$pp \to b\bar{b}b\bar{b}$	441866.0	$4{\cdot}10^6$

Table 1. Madgraph processes, effective cross-sections, and number of generated events for the signal and background samples used in this paper. The effective cross-sections differ slightly from the total theoretical cross-sections due to branching fractions and generation-level cuts on hadronic activity.

2.2 Event Reconstruction

The first step in reconstructing the $4b$ system is to reconstruct and select b-jet candidates. Jets are clustered using an anti-k_T algorithm with a radius of R=0.4. To be selected for use in event reconstruction, a jet must have $p_{\text{T}} > 20$ GeV and an absolute value of $|\ \eta\ | < 2.5$. Delphes uses an internal b-tagging efficiency parameterization to predict whether jets are tagged, and an event is only fully reconstructed if at least 4 jets are b-tagged (unless otherwise specified). The properties and kinematics of selected jets are shown in Fig 2. This strict requirement of having at least 4 b-tags in an event helps to reduce contributions from QCD and reduce the combinatoric ambiguity in event reconstruction.

Once events with at least 4 b-tags are selected, there is a choice about how to reconstruct the di-Higgs system. Several reconstruction methods were tested for pairing b-jets to find an optimal algorithm for correctly pairing Higgs boson constituents. Two algorithms were selected for use in the following sections: the first iterates through all selected jets in an event and returns the two pairs with closest di-jet masses to one another. The second returns the two jet pairs that minimize the difference between the individual candidate pairs and a Higgs boson mass of 125 GeV. Unless otherwise specified, the method that selects di-jets with masses closest to each other is used when training algorithms that require reconstructed events. Fig 3 shows a selection of distributions describing the di-Higgs system using this reconstruction algorithm.

Reconstructed variables include the masses and momentum of the two- and four-body Higgs candidates as well as the angular separations between the two Higgs candidates and their constituent jets. Additional event-level variables like the number of selected jets, the number of b-tagged jets, and the missing transverse energy in the event were also

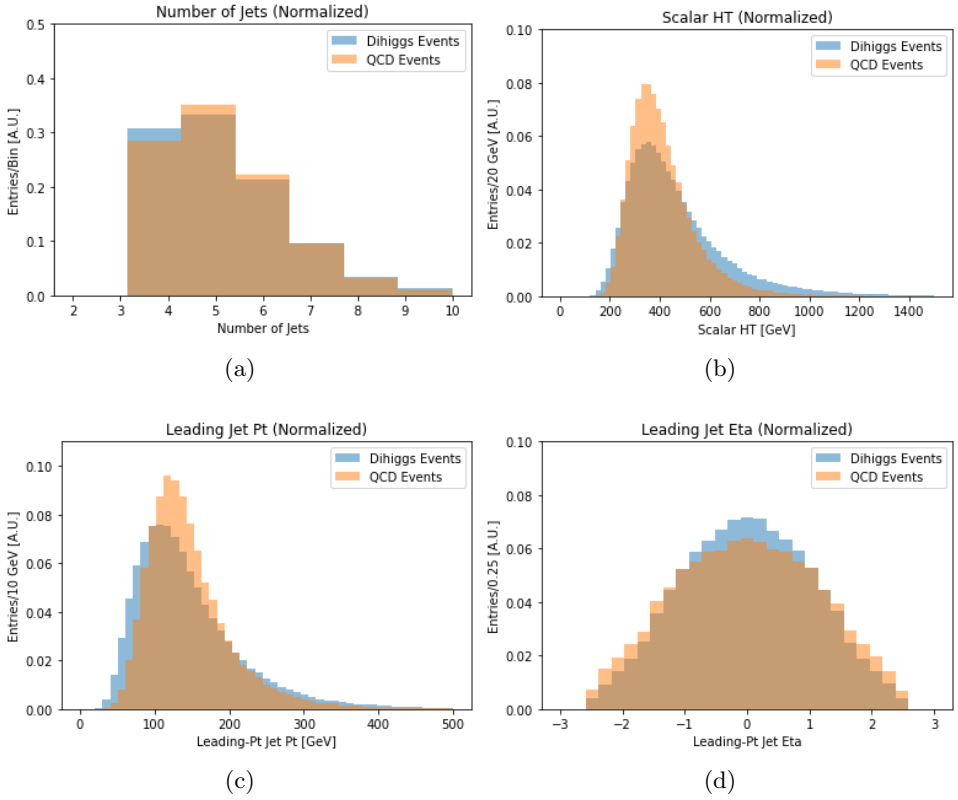


Figure 2. Sample of event information and leading jet kinematics from QCD and di-Higgs simulation. Distributions are normalized to the same area to compare shapes.

considered as inputs to various algorithms. All possible variables were evaluated using the Kolmogorov-Smirnov (KS) test for individual separation power between signal and background. Variables were sorted in descending order of KS separability. Each algorithm is trained on a subset that balances minimizing the number of variables without sacrificing performance.

3 Supervised Learning

Searches for specific signatures or interactions in collider data can be thought of as a classification problem - some known signal process must be identified and separated from some known and well-modeled set of background processes. Any iterative algorithm can then improve its ability to properly identify signal from background by comparing its predicted classifications to the true known classifications and adjusting its internal parameters. This type of approach is known as supervised machine learning, and it is particularly relevant when training models to distinguish between different known processes.

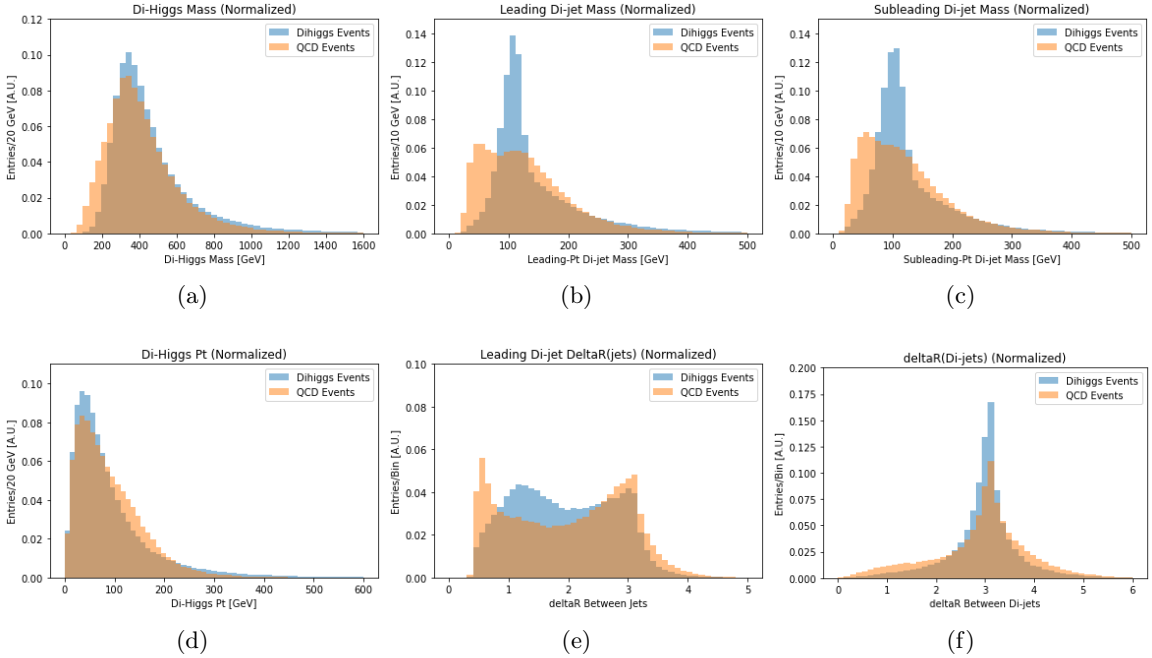


Figure 3. Sample of reconstructed kinematics of the di-Higgs system in QCD and di-Higgs simulation. Distributions are normalized to the same area to compare shapes.

3.1 Boosted Decision Tree

Boosted Decision Trees (BDTs) have a long history in high energy physics from enabling the first observation of single top production at the Tevatron [1, 2] to helping in the discovery of the Higgs boson at the LHC [10, 11]. A decision tree functions by making a series of sequential cuts (or decisions) that each maximize the separation between signal and background events for a single variable. Each cut produces a branch in the tree which contains independent populations. The depth of the tree sets the number of decisions a tree will make, and a well-designed tree will have end-nodes that efficiently separate the constituent classes. Any series of cuts for identifying events will inevitably misclassify some events, and there are many strategies for improving the results. A boosted decision tree attempts to improve the classification by creating a new set of data from the improperly classified events and training a new decision tree on these inputs. Each step of re-training with misclassified events is called a *boost*, and the total prediction for an event is the weighted sum of predictions from the original tree plus the predictions from the boosted trees where each sequential boost receives a smaller weight in the sum.

The BDT trained for di-Higgs detection was built using the xgboost package [12]. The top seventeen reconstructed and event-level variables ranked by KS separability (discussed in Section 2.2) were used in training. The hyperparameters describing the boosted decision tree were optimized for maximum $S/\sqrt{B}$. The optimal hyperparameters were found to be: multiplicative boost factor of 0.1, maximum tree depth of 9, gamma (minimum loss reduction needed for further partition) of 1.1, and an L2 regularization term of 8.28.

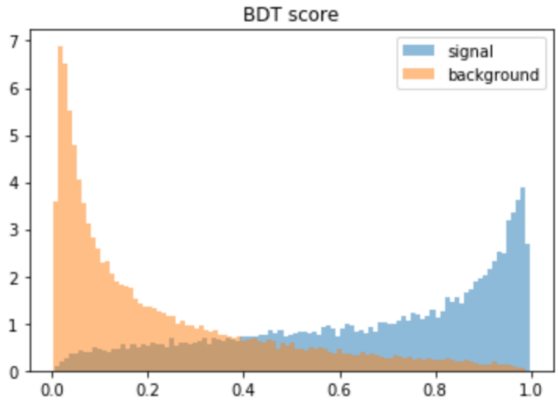


Figure 4. Signal predictions of the trained BDT for independent signal and background samples not used for training.

The predictions from the optimized BDT are shown in Figure 4. A maximum significance of 1.84±0.09 was obtained, yielding 986 signal events and $2.8{\cdot}10^5$ background events.

3.2 Random Forest

Random Forest algorithms share a similar tree structure with BDTs, but they leverage ensembles of independent decision trees as opposed to iteratively improving the predictions of a single tree using misclassified events. Each tree in a random forest is 'grown' using a random sampling of input variables and training events. The randomness of the sampling ensures each tree yields a unique but correlated prediction compared to the other trees in the forest. The class prediction of the forest is the majority vote of the constituent trees. Tuning the hyperparameters of a random forest requires optimizing the number of trees in the forest, the variable sub-sampling used to produce each tree, and the depth of the constituent trees.

The random forest trained for di-Higgs classification uses the reconstruction algorithm that selects di-jet pairs consistent with a Higgs mass hypothesis, and the top seventeen reconstructed and event-level variables were used as input. The random forest was implemented using xgboost [12]. An optimal forest was obtained by individually varying each hyperparameter over a reasonable range and selecting the best performing model. The optimal set of hyperparameters consisted of training with 300 constituent trees, a maximum tree depth of 20, column sub-sampling rate of 0.8, and an L1 regularization term of 1.175. The best significance for the random forest approach was found to be $S/\sqrt{B} = 2.44\pm0.19$ when requiring a prediction score > 0.80. Prediction results are shown in Figure 5.

3.3 Feed Forward Neural Network

Fully connected or feed-forward neural networks (NNs) also have a long history in high energy physics. The earliest usage of neural networks in particle physics were as part of the

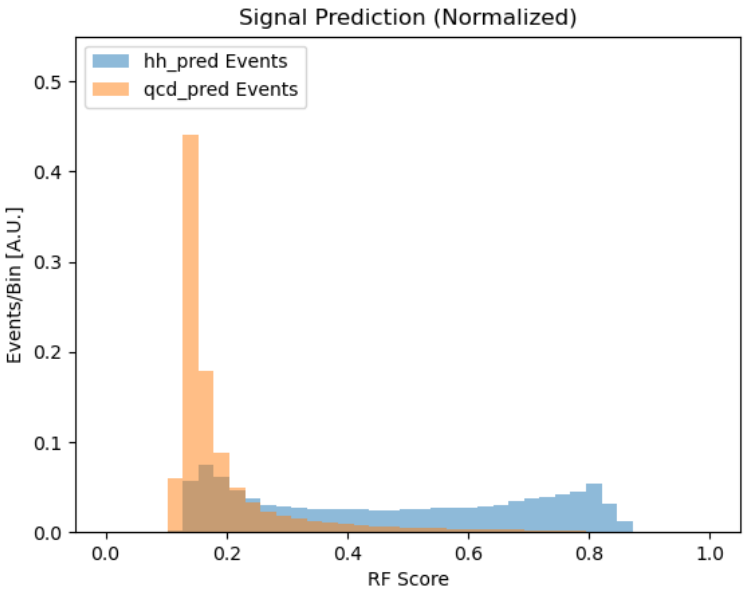


Figure 5. Output score on the testing dataset with the fully trained random forest classifier.

program of top physics measurements at the Tevatron[1, 2]. The fundamental element of any neural network is called a *layer*. Multiple layers are stacked together to connect input variables with a predicted outcome which is then evaluated against a known target value. A fully-connected network can have multiple internal (or hidden) layers between the input and output layers, and each hidden layer is composed of a series of trainable activation functions and weights that allow the network to identify and iteratively combine important features of the input space. A function (called the loss function) is chosen to quantify the difference between the model prediction and target values. The loss calculated after a single training iteration is used to adjust the internal network weights in the next training iteration through a process called backpropagation. The model is fully trained once the improvement in the loss between iterations falls beneath some user-defined threshold.

The NN trained for di-Higgs detection was built using the Keras [13] and Tensorflow [14] packages. The top twenty-two most separable reconstructed and event-level variables were used as the input variables for the NN. The complete network structure consists of the input layer, two hidden layers, and a single-node output layer. The first hidden layer contains 175 nodes with an L2 kernel regularizer ($\lambda = 10^{-4}$). The second hidden layer contains 90 nodes with no kernel regularizer. A batch normalization layer and a dropout (0.2) function are placed in between the two hidden layers to prevent over-fitting. Both hidden layers use a rectified linear (ReLU) activation function, while the output layer uses a sigmoid activation function. Several models were trained by individually tuning each hyperparameter over a reasonable range in order to produce a final optimized model. A schematic flowchart of the network structure is shown in Figure 6.

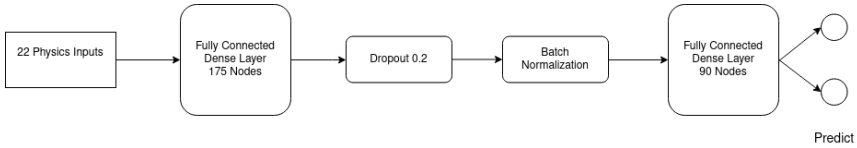


Figure 6. Structure of the feed-forward neural network. The input variables are fed through two fully connected dense layers to classify events. One dropout layer and one batch normalization layer help mitigate over-fitting during training.

The NN was trained for 25 epochs before the minimal loss-improvement threshold was met, and the results are shown in Figure 7. The trained model obtained a maximum $S/\sqrt{B} = 2.40 \pm 0.08$ when considering events with a signal prediction score > 0.94. This phase-space has a signal yield of 1659.9 events and a background yield of 477215.3 events.

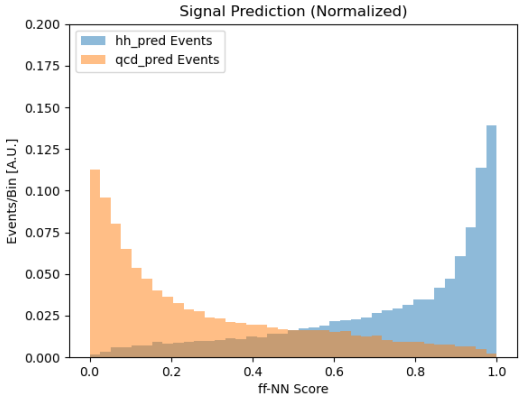


Figure 7. Final predictions of the feed-forward network for signal and background samples.

3.4 Convolutional Neural Network

Convolutional Neural Networks (CNNs) are neural networks that predict the content of an input image by using assumptions about the local relationships between neighboring pixels. For this analysis, content prediction is simplified to a general classification of whether the image comes from a di-Higgs or QCD event. The fundamental elements of any convolutional network are convolutional layers and pooling layers. Convolutional layers use filters that perform linear combinations of neighboring pixels within the filter size, and pooling layers aggregate information by grouping neighboring pixels using either their maximum or average values. After some number of these layers, the output is flattened into a one-dimensional vector, and this flattened vector is pushed through a set of feed-forward layers in order to make a final output prediction.

Many previous papers have explored the use of convolutional networks trained on low-level quantities (e.g. tracks and calorimeter deposits) for the purposes of object identification [15] at colliders. This paper extends the application to event-level identification. Using

low-level quantities removes the need to reconstruct higher-level objects like jets or jet pairs; only the detector-level measurements are required for image creation. The performance of four convolutional networks were studied in the context of di-Higgs identification. The first network used a 3-layer image composed of energy/momentum weighted tracks, electromagnetic calorimeter deposits, and hadronic calorimeter deposits. The second network uses the same three layers but appends additional global event-level information to the flattened vector after image processing and before the fully connected layers. Figures 8 and 9 depict both network structures. The third and fourth networks follow the same pattern as the previous two but with the addition of two image layers corresponding to longitudinal and transverse impact parameter-weighted track information.

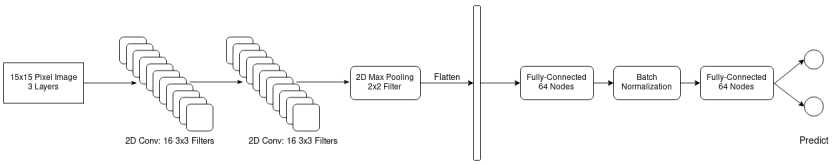


Figure 8. Structure of the nominal convolutional neural network. The input images are fed through two convolutional layers and a single max-pooling layer before being flattened into a one-dimensional vector. The flattened vector is then fed through one fully connected layer, a batch normalization layer, and a final fully connected layer before a final prediction is made.

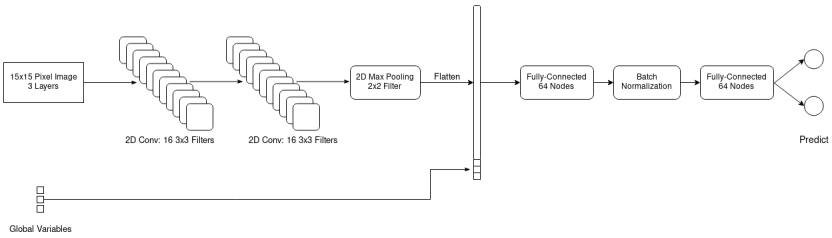
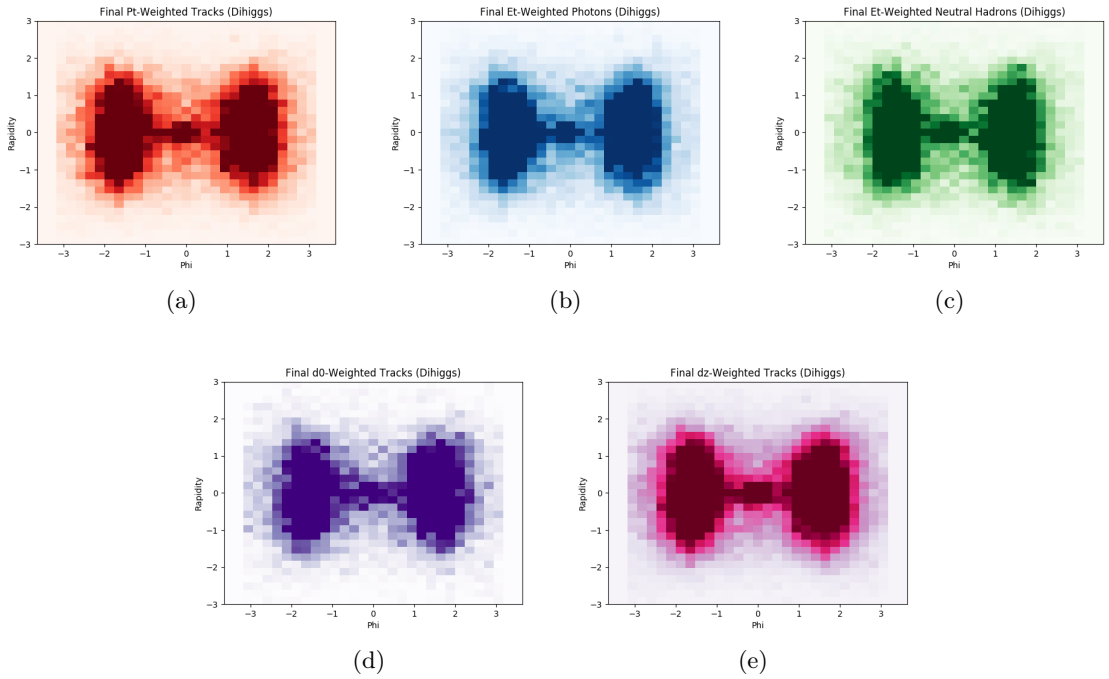


Figure 9. Structure of the hybrid convolutional neural network. The input images are fed through two convolutional layers and a single max-pooling layer before being flattened into a one-dimensional vector. Scaled user-specified variables (e.g. H_T) are then concatenated with the flattened image vector. The concatenated vector is then fed through one fully connected layer, a batch normalization layer, and a final fully connected layer before a final prediction is made.

In order to produce coherent images, the center of mass and the center of momentum for each event are calculated. All constituents are then boosted longitudinally into the center of mass of the event and rotated in phi to the center of momentum. After this pre-processing, each image layer corresponds to a 31x31 pixel grid centered on the total activity in the event. Figure 10 shows an average QCD image and Figure 11 shows an average di-Higgs image. While the average image layers for each sample closely resemble one another, they do contain different information, and variations are visible.

Importantly, clear differences are observed between the average QCD images and the average di-Higgs images. Each half of the di-Higgs image (split across $\phi = 0$) is arranged in a roughly circular, isotropic shape due to the spin-0 nature of the Higgs. The QCD images appear balanced because of the effect of the pre-processing, but no similar circular structure is produced. Additionally, the variance of pixel intensities in di-Higgs images is much smaller than the variance in QCD images due to the more balanced kinematics of Higgs pair production compared to QCD processes.

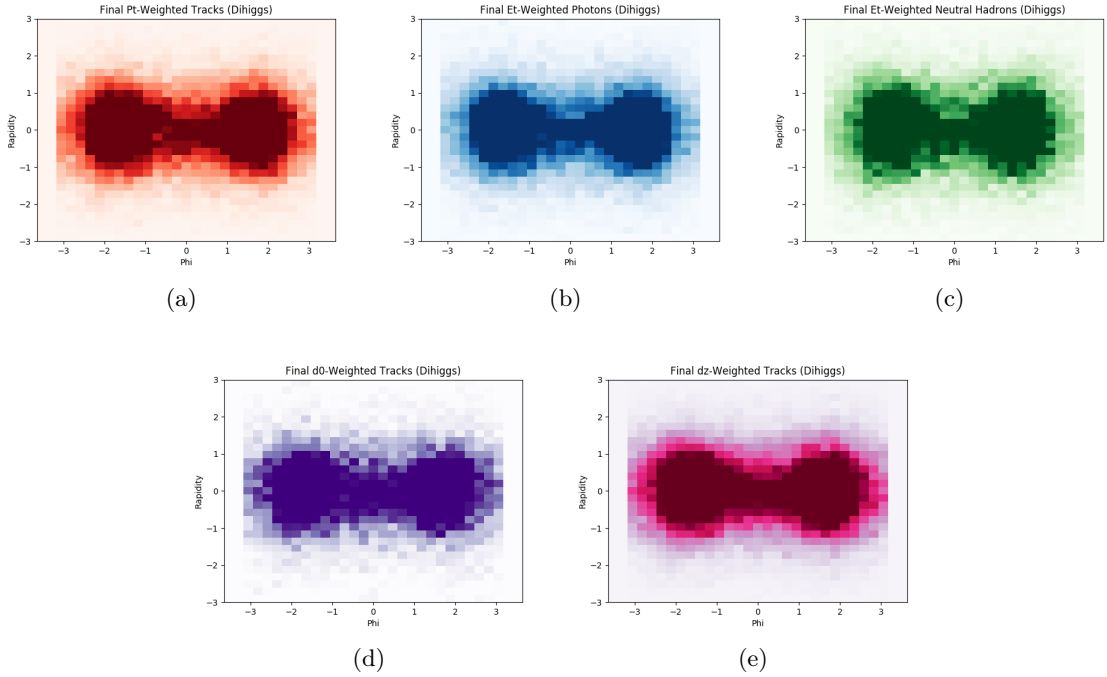


(a) (b) (c)

(d) (e)

Figure 10. Average QCD image showing (a) p_{T}-weighted tracks, (b) E_{T}-weighted ECAL deposits, (c) E_{T}-weighted HCAL deposits, (d) transverse impact parameter-weighted tracks, (e) longitudinal impact parameter-weighted tracks.

As shown in Figures 8 and 9, the CNN network structure uses two sequential 2D convolutional layers each with 16 3x3 filters, one max-pooling layer with a 2x2 window, a flattening of the outputs, two 64-node fully connected hidden layers, and one output layer for making the final prediction. As previously described, two of the networks append additional high level variables (scalar sum of transverse hadronic energy, number of jets, and number of b-tags) after the flattening and before the image information is fed through the fully connected layers. The optimal significance for each network is shown in Table 2. The best results were obtained using the 5-color network with additional high-level inputs, and the final predictions for this configuration are shown in Figure 12. A best significance of 2.86 ± 0.03 was found for a prediction cut > 0.94 with a signal yield of $1.0 \cdot 10^4$ events and a background yield of $1.3 \cdot 10^7$ events.



(a) (b) (c)

(d) (e)

Figure 11. Average di-Higgs image showing (a) p_{T}-weighted tracks, (b) E_{T}-weighted ECAL deposits, (c) E_{T}-weighted HCAL deposits, (d) transverse impact parameter-weighted tracks, (e) longitudinal impact parameter-weighted tracks.

Method	Best $S/\sqrt{B}$	AUC
Tracks+HCAL+ECAL	1.77 ± 0.01	0.818
Tracks+HCAL+ECAL + high-level	2.12 ± 0.01	0.846
Tracks+HCAL+ECAL+D0+DZ	2.45 ± 0.02	0.863
Tracks+HCAL+ECAL+D0+DZ + high-level	2.86 ± 0.03	0.882

Table 2. Normalized to full HL-LHC dataset of 3000 fb^{-1}

3.5 Energy Flow Network

Energy Flow Networks (EFN) and Particle Flow Networks (PFN) are neural networks that also operate with basic jet constituent information as input rather than reconstructed jets and multi-jet composites [16]. The EFN structure takes only the rapidity, y, and azimuthal angle, ϕ, of jet constituents as input, while the PFN takes the rapidity, azimuthal angle, and transverse momentum, p_T, of jet constituents as input. Both the EFN and PFN are two-component networks, and their internal structures are shown in Figure 13. The implementations of the EFN and PFN used for di-Higgs classification use 200 nodes for each hidden layer in network (a), 256 nodes for the latent space dimension, and 300 nodes for each hidden layer in network (b).

The EFN/PFN networks were trained using four separate categories split by number of

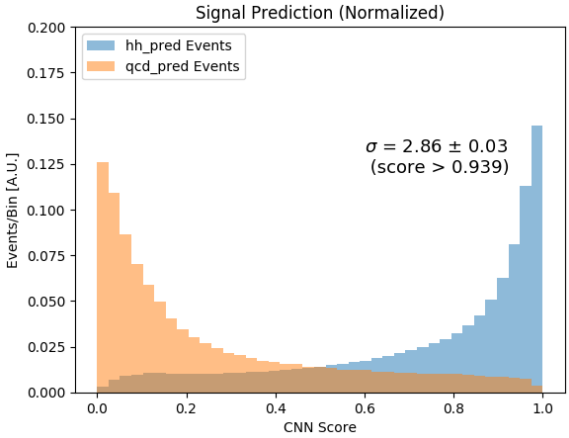


Figure 12. Signal prediction for the 5-color convolutional network with additional high-level inputs. The total area of the signal and background predictions are normalized to unity for easier shape comparison.

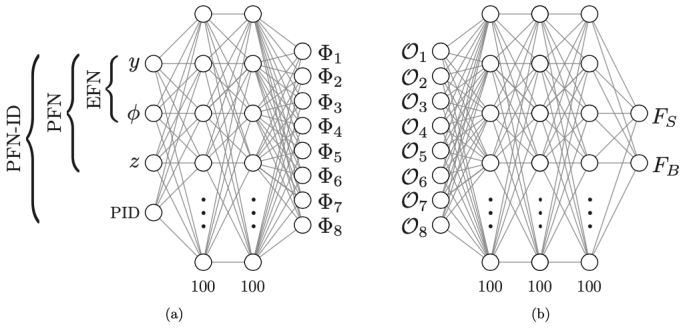


Figure 13. Network (a) takes jet constituents information as input and outputs latent space Φ for each jet constituents. Network (b) takes $\mathcal{O}$, which is the linear combination of Φ, as input and outputs final result.

jets and number of b-tags to test the network's dependence on higher-level jet information. Independent networks were trained using: all events, only events with $\geq$4 jets, only events with $\geq$4 jets and $=$2 b-tags, and only events with $\geq$4 jets and $\geq$4 b-tags. In each configuration, the number of signal and background events were adjusted to maintain an equal proportion of each population in the training sample. L2 regularization and dropout layers were added to minimize over-fitting. The results obtained from each EFN configuration are shown in Table 3. The results of each PFN configuration are shown in Table 4.

Both networks performed best when trained over all events without any cuts on the number of jets or b-tags. The EFN obtained a highest significance of 1.41$\pm$0.01, and the PFN obtained a highest significance of 1.62$\pm$0.01.

Category	0PU		
	Best $S/\sqrt{B}$	$\mathbf{N}_{\text{Signal}}$	$\mathbf{N}_{\text{Background}}$
All Events	1.407 ± 0.006	$1.89 \cdot 10^4$	$1.80 \cdot 10^8$
4Jets	1.363 ± 0.006	$1.63 \cdot 10^4$	$1.43 \cdot 10^8$
4Jets 2BTags	1.343 ± 0.006	$1.33 \cdot 10^4$	$9.95 \cdot 10^7$
4Jets 4BTags	0.867 ± 0.008	3468.65	$1.60 \cdot 10^7$

Table 3. EFN results. Normalized to full HL-LHC dataset of 3000 fb^{-1}

Category	0PU		
	Best $S/\sqrt{B}$	$\mathbf{N}_{\text{Signal}}$	$\mathbf{N}_{\text{Background}}$
All Events	1.618 ± 0.008	$1.79 \cdot 10^4$	$1.21 \cdot 10^8$
4Jets	1.580 ± 0.008	$1.32 \cdot 10^4$	$7.00 \cdot 10^7$
4Jets 2BTags	1.574 ± 0.009	$1.32 \cdot 10^4$	$4.85 \cdot 10^7$
4Jets 4BTags	0.903 ± 0.009	3297.34	$1.33 \cdot 10^7$

Table 4. PFN results. Normalized to full HL-LHC dataset of 3000 fb^{-1}

4 Semi-Supervised Learning

While it makes sense to treat searches for new physics or rare signatures as a supervised classification problem, an alternative approach is to let an algorithm learn intrinsic features from an unlabeled dataset and then evaluate whether this self-learned information can be used to separate signal from background processes. The process of learning features from unlabeled datasets is called unsupervised learning. Unsupervised machine learning can be transformed into semi-supervised learning when the results of unsupervised learning are evaluated using known classification information.

4.1 k-Means Clustering

K-means clustering is an unsupervised learning algorithm that finds natural unlabeled groupings in the phase-space of the inputs. The k-means approach creates clusters by defining cluster centroids and associating events to the closest nearby centroid. The centroid positions are iteratively improved by minimizing the ensemble distance of all events to their associated clusters. Combining unsupervised clustering with the supervised structure of a BDT converts the unsupervised approach into a semi-supervised algorithm whose performance can be compared to other supervised methods.

The number of clusters to fit is a user-defined hyperparameter, and three different clusterings (15, 20, 40) were tested. Two unsupervised transformations of the input variables were tested to try to improve the performance of the nominal BDT. The first approach was to pass all reconstructed kinematic inputs through a k-means clustering stage before training the BDT. The second transformation involved performing a principal component analysis (PCA) decomposition on the nominal kinematic inputs before passing through the clustering step and the BDT. PCA is a technique for finding an orthogonal basis of the input data that minimizes the variance along each new axis. No transformation was found

to improve the performance of the nominal configuration, and the results are shown in Table 5.

Method	$S/\sqrt{B}$	N_{sig}	N_{bkg}
Nominal BDT	1.84 ± 0.09	986.3	$2.9{\cdot}10^5$
15 Clusters	1.29 ± 0.02	2100.2	$2.7{\cdot}10^6$
15 Clusters + PCA	1.25 ± 0.02	2189.5	$3.1{\cdot}10^6$
20 Clusters	1.30 ± 0.02	2260.6	$3.0{\cdot}10^6$
20 Clusters + PCA	1.27 ± 0.03	21756.4	$1.9{\cdot}10^6$
40 Clusters	1.44 ± 0.03	1704.6	$1.4{\cdot}10^6$
40 Clusters + PCA	1.34 ± 0.02	2144.5	$2.0{\cdot}10^6$

Table 5. Significance and yields showing BDT performance when using the nominal kinematic inputs, clustered kinematic inputs, and clustered inputs from a PCA decomposition. All yields are normalized to full HL-LHC dataset of 3000 fb^{-1}.

4.2 Autoencoder

An autoencoder (AE) is an unsupervised machine learning architecture used for detecting anomalies that differ significantly from the data used to train the network. The structure of the AE compresses the input information into a lower-dimensional representation called the latent space. This compression 'encodes' the most important features of the training data into the latent space, while the second half of the network 'decodes' the latent space back into a representation approaching the original inputs. This construction fundamentally changes the meaning of the loss calculation; rather than computing the loss between a prediction and a target, the AE loss is a measure of how well the network reproduces the original inputs after encoding and decoding. Inputs that differ significantly from the data used to train the AE will not be properly reconstructed, and anomalies can be identified by selecting events with large losses. Training with Monte Carlo simulations allows for a semi-supervised cross-check on AE performance since the classes of training and testing samples are known in advance.

Because AE anomaly detection relies on a well-modeled understanding of background processes, the network was trained using only QCD events. Additional models were trained by substituting the pure-QCD training sample with training samples consisting of mixtures of QCD and di-Higgs events to test the stability of the method against signal contamination. No significant deterioration was observed for reasonable levels of contamination. The AE used for di-Higgs detection was built using the Keras package [13] and consists of an input layer, a single hidden layer, and an output layer. Eleven reconstructed variables were selected for use in the AE.

The output layer is a mirror of the input layer and therefore has 11 nodes. A PCA analysis (shown in Figure 14) was used to determine that a latent space of three nodes was optimal. The hidden layer and output layer use ReLU and sigmoid activation functions respectively. An L2 regularization term was added to the hidden layer to avoid over-fitting.

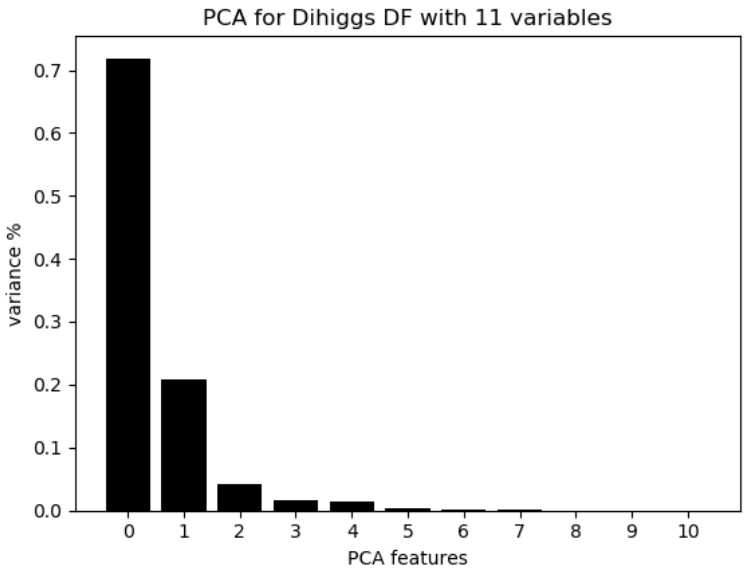


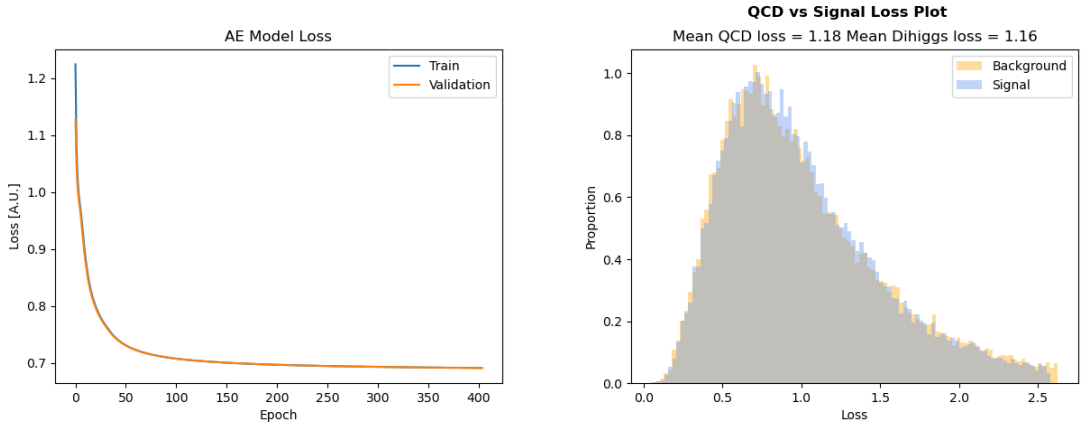
Figure 14. PCA performed on the selected eleven kinematic inputs. The x-axis indicates the number of PCA features, and the y-axis indicates the variance. Choosing the optimal size for the latent space requires identifying the point of diminishing variance return.

Because training an autoencoder is an unsupervised and unlabeled process, there is no prediction of whether a given event is signal or background. Since di-Higgs events should be relatively anomalous compared to the QCD training set, signal events should have a relatively larger average loss value. Cutting on the loss function allows for a significance to be calculated for comparison with other fully supervised methods.

Training for several hundred epochs leads the model to converge, and it reaches an asymptotic loss value near 0.7 (see Fig.15). Requiring the loss to be larger that 0.05 yields a best $S/\sqrt{B}$ of 0.81±0.01. This significance value may be somewhat misleading since the signal and background loss distributions have little separation. The highest significance result effectively is a cut that keeps nearly all events. This suggests that the kinematic inputs used in training do not significantly differ between signal and background processes after the latent space compression.

5 Results

The methods covered in this paper are by no means an exhaustive review of the ML landscape available to high energy physics. Still, a wide range of techniques and philosophies are covered. Table 6 provides a summary of the methods described in the previous sections. The results of a traditional 1-D sequential cut technique is shown for comparison though the details have not been discussed in the previous sections. Clear gains in sensitivity compared to this baseline are apparent for many of the ML models tested in this review.



(a) (b)

Figure 15. (Left) The loss of the AE during QCD training/reconstruction converged after 700 epochs. (Right) The loss distribution generated by the AE when being tested on QCD and di-Higgs event data separately.

Method	0PU		
	Best $S/\sqrt{B}$	$\mathbf{N}_{\text{Signal}}$	$\mathbf{N}_{\text{Background}}$
Autoencoder	0.81 ± 0.01	5840.8	$5.2 \cdot 10^7$
1D-Rectangular Cuts	0.82 ± 0.02	3621.0	$1.97 \cdot 10^7$
k-Means Clustering	1.44 ± 0.02	1703.6	$1.4 \cdot 10^6$
Particle Flow Network	1.62 ± 0.01	$1.8 \cdot 10^4$	$1.2 \cdot 10^8$
Boosted Decision Tree	1.84 ± 0.09	986.3	$2.8 \cdot 10^5$
Feed-Forward NN	2.40 ± 0.08	1659.9	$4.8 \cdot 10^5$
Random Forest	2.44 ± 0.19	544.7	$5.0 \cdot 10^4$
Convolutional NN	2.85 ± 0.02	$1.0 \cdot 10^4$	$1.3 \cdot 10^7$

Table 6. Comparison of method significance and signal/background yields normalized to full HL-LHC dataset of 3000 fb^{-1}.

An important caveat to keep in mind is that all results discussed here were determined in conditions with zero pileup. In higher pileup environments like those expected at the HL-LHC, reconstruction algorithms see serious reductions in correct combinatoric matching. This effect will certainly degrade the expected performance of techniques that rely on explicit event reconstruction. Methods that do not rely on event reconstruction (CNN, PFN) might be more robust to these effects, and this should be studied in further work. The unsupervised AE technique performed the worst among all the methods tested, but this is likely a reflection of the fact that model-specific methods often outperform model-unspecific methods when evaluated on the model used in training.

6 Conclusions

Measuring the rate of di-Higgs production will be a problem facing the high energy physics community through the end of the HL-LHC era. The techniques explored in this paper show the power of machine learning techniques in identifying di-Higgs signals amid overwhelming QCD backgrounds. The significance obtained for many of these methods is impressive given the simplicity of the evaluation metric. This bodes well for both current and future measurements of Higgs pair production at the LHC.

References

[1] D0 collaboration, *Evidence for Production of Single Top Quarks and First Direct Measurement of |Vtb|*, *Phys. Rev. Lett.* **98** (2007) 181802 [hep-ex/0612052].

[2] CDF collaboration, *Measurement of the Single Top Quark Production Cross Section at CDF*, *Phys. Rev. Lett.* **101** (2008) 252001 [0809.2581].

[3] K.A. et al., *Machine Learning in High Energy Physics Community White Paper*, 2018.

[4] LHC Higgs Cross Section Working Group collaboration, *Handbook of LHC Higgs Cross Sections: 4. Deciphering the Nature of the Higgs Sector*, 1610.07922.

[5] R. Brun and F. Rademakers, *ROOT: An Object Oriented Data Analysis Framework*, *Nucl. Instrum. Meth. A* **389** (1997) 81.

[6] J. Alwall, R. Frederix, S. Frixione, V. Hirschi, F. Maltoni, O. Mattelaer et al., *The Automated Computation of Tree-Level and Next-to-Leading Order Differential Cross Sections, and their Matching to Parton Shower Simulations*, *JHEP* **07** (2014) 079 [1405.0301].

[7] T. Sjöstrand, S. Ask, J.R. Christiansen, R. Corke, N. Desai, P. Ilten et al., *An Introduction to PYTHIA 8.2*, *Computer Physics Communications* **191** (2015) 159–177.

[8] J. de Favereau, C. Delaere, P. Demin, A. Giammanco, V. Lemaître, A. Mertens et al., *DELPHES 3: A Modular Framework for Fast Simulation of a Generic Collider Experiment*, *Journal of High Energy Physics* **2014** (2014) .

[9] B. Tannenwald, A. Li, A. Cuddeback, R. Parvatam and C. Thompson, "dihiggsMLProject." https://github.com/neu-physics/dihiggsMLProject, 2020.

[10] G. Aad, T. Abajyan, B. Abbott, J. Abdallah, S. Abdel Khalek, A. Abdelalim et al., *Observation of a New Particle in the Search for the Standard Model Higgs boson with the ATLAS detector at the LHC*, *Physics Letters B* **716** (2012) 1–29.

[11] S. Chatrchyan, V. Khachatryan, A. Sirunyan, A. Tumasyan, W. Adam, E. Aguilo et al., *Observation of a New Boson at a Mass of 125 GeV with the CMS experiment at the LHC*, *Physics Letters B* **716** (2012) 30–61.

[12] T. Chen and C. Guestrin, *XGBoost: A Scalable Tree Boosting System*, *CoRR* **abs/1603.02754** (2016) [1603.02754].

[13] F. Chollet, "Keras." https://github.com/fchollet/keras, 2015.

[14] M.A. et al., *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems*, *CoRR* **abs/1603.04467** (2016) [1603.04467].

[15] J. Alison, S. An, P. Bryant, B. Burkle, S. Gleyzer, M. Narain et al., *End-to-end Particle and Event Identification at the Large Hadron Collider with CMS Open Data*, in *Meeting of the Division of Particles and Fields of the American Physical Society*, 10, 2019 [1910.07029].

[16] P.T. Komiske, E.M. Metodiev and J. Thaler, *Energy Flow Networks: Deep Sets for Particle Jets*, *JHEP* **01** (2019) 121 [1810.05165].