Prepared for submission to JHEP

Learning About Machine Learning With Dihiggs 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: Machine learning techniques have started being explored in many new domains of high energy physics analysis. 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, neural networks with straightforward and novel architectures, semi-supervised algorithms) against one another in an attempt to catalogue the various avaiable to high energy physicists as the era of the HL-LHC approaches.

Contents

1 Introduction

Talk about machine learning (ML). It's really cool. Also most of it is just ideas that people came up with in the 90s and we only now have sufficient computing architecture. Yes Chris used a simple neural network in his thesis at the Tevatron, but honestly, who cares about history these days?

Deep networks have a lot of really cool potential and are pretty easy to implement. The usual feed-forward approach isn't too complicated, but it can have lots of hyperparameters to worry about! You can also do lots of cool stuff that involve emergent information like Lorentz Boost Networks or maybe even energyflow stuff. These approaches raise interesting questions about what sorts of information are best to feed to our networks; things that are best for physicists to learn from might not be optimal for machines!

You can even think of fun semi-supervised approaches like clustering and maybe some stuff with auto-encoders. To be totally honest, I'm not really sure if the auto-encoder stuff is relevant for this paper, but it's really interesting and is maybe cool for unexpected BSM signatures? Could be a cool side-study.

All of this is really interesting, but it's important to ground it in reality. High energy physicists care about stuff like dihiggs production because it's lit. The problem is that it's

really rare. And the most common branching fraction ($hh \to b\bar{b}b\bar{b}$) is swamped by orders more magnitude similar-appearing events from QCD. The goal of this paper is to show how effective different machine learning techniques are at overcoming this issue and leading to potential Higgs self-coupling measurements at the LHC.

Section 2 deals with the physics relevant for dihiggs production and the QCD background. Sections 3 and 4 summarize the various ML methods we played with, and Section 5 compares the results from the various approaches.

2 Dihiggs Physics

2.1 Double Higgs Production

The Higgs boson is an essential part of the SM 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 Standard Model,

$$\sigma_{hh}\ (13\ \text{TeV}) = 33\ \text{fb},$$

any significant enhancement would indicate the presence of new physics.

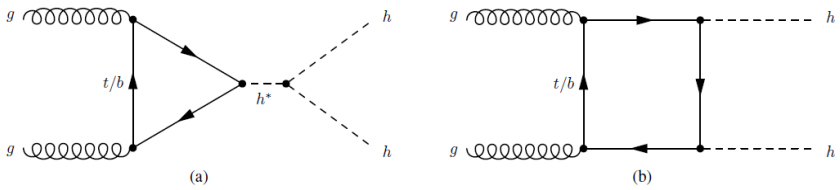


Figure 1. Leading order Feynman diagrams for non-resonant production of Higgs boson pairs in the Standard Model through (a) the Higgs boson self-coupling and (b) the Higgs-fermion Yukawa interaction.

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 dihiggs cross-section (or placing limits on its magnitude) allow us to probe the self-coulpling 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 (an integrated luminosity of 3000 fb^{-1}). Simulated samples are produced with Madgraph [1] and recontructed in Delphes [2] using an approximation of the upgraded Phase-II CMS detector.

2.2 Event Reconstruction

We tried a lot of different methods to build dijet pairs for hh reconstruction. This inherently involves making some choices, and no method is 100% efficient. Sometimes one or another is better for a given method. What's cool is that some methods don't require reconstrucion at all.

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 $p_{\mathrm{T}} > 20$ 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 reconstructed only 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 recontruction.

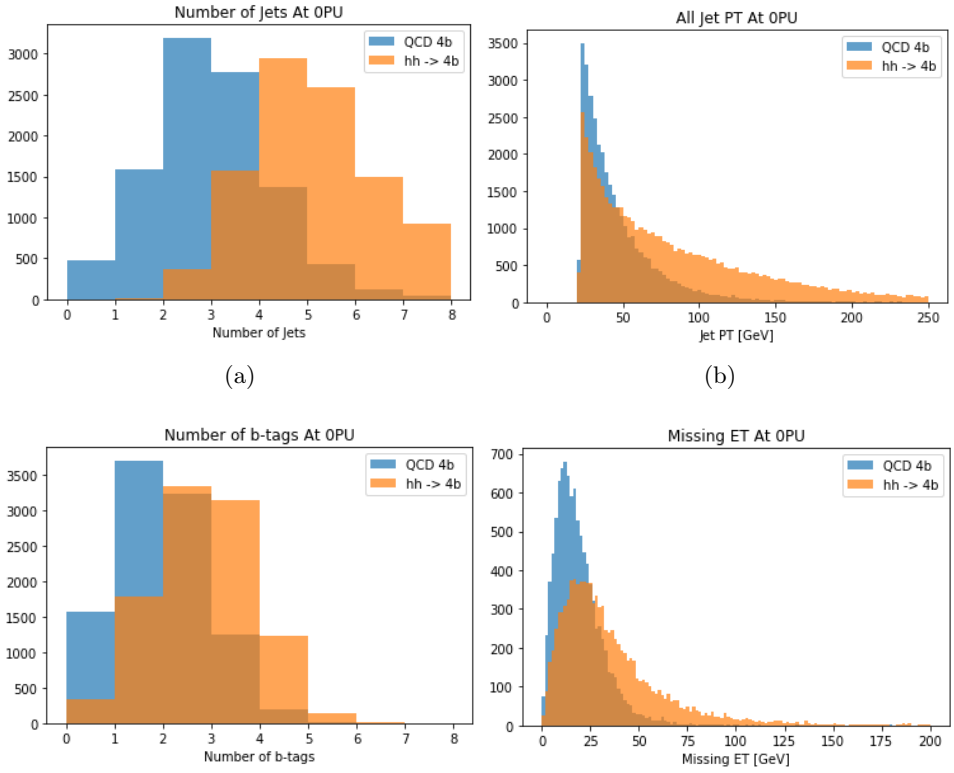
Once events with at least 4 b-tags are selected, there is a choice about how to reconstructed the dihiggs system. Several reconstruction methods were testedfor pairing b-jets to find an optimal algorithm for correctly pairing Higgs boson consituents, and two algorithms were selected for use in the following sections. The first iterates through all selected jets in an event and returns the two pairs with closest dijet masses ('equalDijetMass'), and the second returns the two jet pairs that minimize the difference between the candidate pairs and a higgs boson mass of 125 GeV ('closestDijetMassToHiggs'). Fig 2 shows a selection of distributions describing the reconstructed dihiggs system for each algorithm. Unless otherwise specified, the 'equalDijetMass' algorithm is used for machine learning techniques that require reconstructed events.

3 Supervised Learning

Searches for specific signatures or interactions in collider data can in general 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 predictions to the true known classifications and adjusting its internal parameters. This type of approach is known as supervised machine learning, and it is particulary relevant for measuring dihiggs decays.

3.1 Boosted Decision Trees

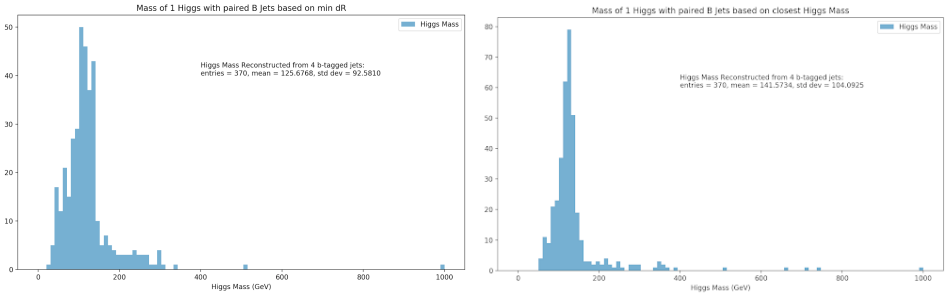
Boosted Decision Trees (BDTs) have a long history in high energy physics and have been used since Ancient Age to identify a variety of signatures [CITATIONS NEEDED]. A deci-



(a) (b)

(c) (d)

Figure 2. Event and jet properties and kinematics of events from QCD and dihiggs simulation. [THESE ARE PLACEHOLDER PLOTS.]



(a) (b)

Figure 3. Leading dijet mass using different reconstruction algorithms. Left plot shows mass when minimizing dijet angular separation, right plot shows mass when minimizing different between dijet mass and a Higgs mass of 125 GeV ('closestDijetMassToHiggs'). [PLACEHOLDER PLOTS]

sion tree functions by making a seriies of cuts (or decisions) that maximize the separation between signal and background events in a single dimension. Each cut produces a branch in the tree containing independent populations. The depth of three sets the number of de-

cisions a tree will make, and a well-designed tree will have end-nodes that provide relatively pure classification of the inputs. Any series of cuts for identifying events will inevitably mis-classify 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 mis-classified events is called a *boost*, and the total prediction for an eventis the weighted sum of predictions from the orignal tree and the boosted trees where each sequential boost recieves a smaller weight in the sum.

The BDT trained for dihiggs detection was built using the xgboost package [3] and uses events reconstructed using the 'equalDijetMass' reconstruction algorithm. Reconstructed variables include the masses and momentum of the four- and two-body higgs candidates as well as the angular separations between the two Higgs candidates and their consituent 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 considered. All possible varibles were evaluted for individual separation power between signal and background, and the top seventeen variables were used in training. The hyperparameters describing the boosted decision tree were optimized for maximum $S/\sqrt{B}$. The optimal hyperparamters were found to be as follows: 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.

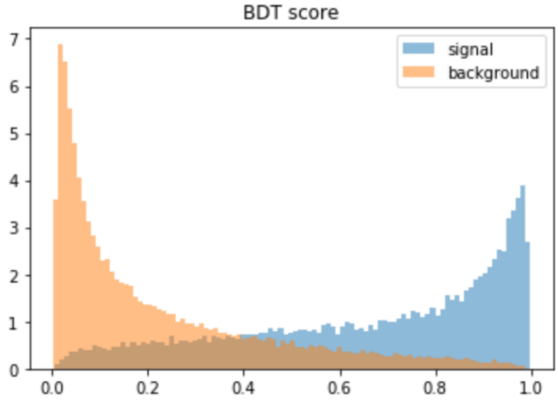


Figure 4. Signal predictions of the trained BDT for signal and background samples independent from the training sets.

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.

3.1.1 PCA and Clustering Extensions

Two additional transformations of the input variables were tested to try to improve the BDT performance. The first transformation was to pass the nominal kinematic inputs through a k-means clustering stage before training the BDT. K-means clustering is an

unsupervised learning algorithm that finds unlabelled groupings in the phase-space defined by the input variables. The number of clusters to fit is a user-defined hyperparameter, and three different clusterings (15, 20, 40) were tried. The second transformation involved performing a principal component analysis (PCA) decomposition on the nominal kinematic inputs before passing through the clustering step and finally the BDT. PCA is a technique for finding an orthogonal basis of 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 1.

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 1. Significance and yields showing BDT performance for 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}.

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 mis-classified 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 sub-sampling used to produce each tree, as well as more familiar decision tree parameters like child depth.

The random forest trained for dihiggs classification uses the 'closestDijetMassToHiggs' reconstruction algorithm and samples from the top seventeen reconstructed kinematic and event-level variables. The input variables were selected as having the highest separability between dihiggs and QCD processes. The random forest was implemented using the 'XGBRFClassifier' functionality from xgboost [3]. Detail the hyperparameters. Predictions were made on data fully independent from the training data, and the results are shown in Figure 5

The best significance for the random forest approach is found to be $S/\sqrt{B} = 2.17{\pm}0.22$ when requiring a prediction score > 0.81.

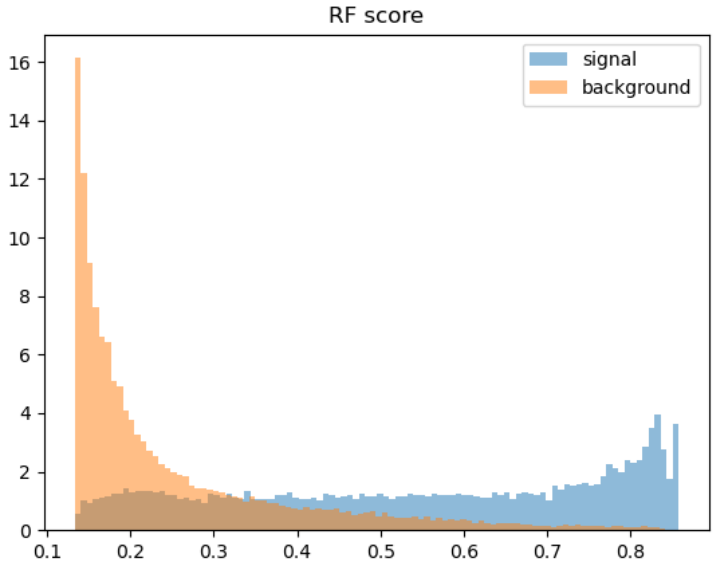


Figure 5. Output score on the testing dataset with the fully trained random forest classifier.

3.3 Feed Forward Neural Network

Fully connected or feed-forward neural networks (ffNN or NN) have a long history in high energy physics. One of the earliest applications of this type of approach was in a search for top quark production using the CDF experiment at the Tevatron. The fundamental element of a feed-forward neural network is called a 'layer', and these layers connect input variables to a predicted outcome which is 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 vector of relevant physics-level information (e.g. mass of two highest p_{T} jets, angle between measured objects, etc) is constructed for each event, and these vectors are then 'fed forward' through multiple layers in order to predict whether the event comes from a signal dihiggs process or a background QCD process. A function is chosen to quantify the difference between the model prediction and target values, and the loss is used to adjust the network weights in the next training iteration in a process called backpropagation. The training stops and the model is fully trained once the improvement in the loss between iterations falls beneath some user-defined threshold.

Using the KS test method, the 22 best variables for the 'equalDijetMass' reconstruction algorithm were chosen to be the input variables for the feed forward neural network (FFNN).

The NN used for dihiggs classification is built using the Keras [4] and Tensorflow [5] pacakges. Input events are reconstructed using the 'closestDijetToHiggsMass' reconstruction algorithm, and the top twenty-two variables identified as most-separable are used as

the input vector. The complete NN structure consists the an input layer, two hidden layers, and an 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. A schematic flowchart of the network structure is shown in Figure 6

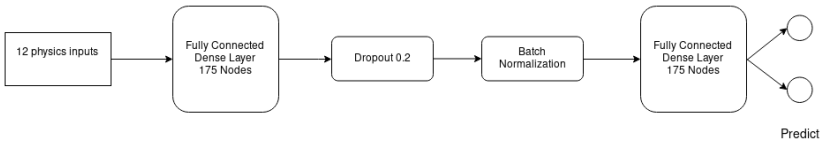
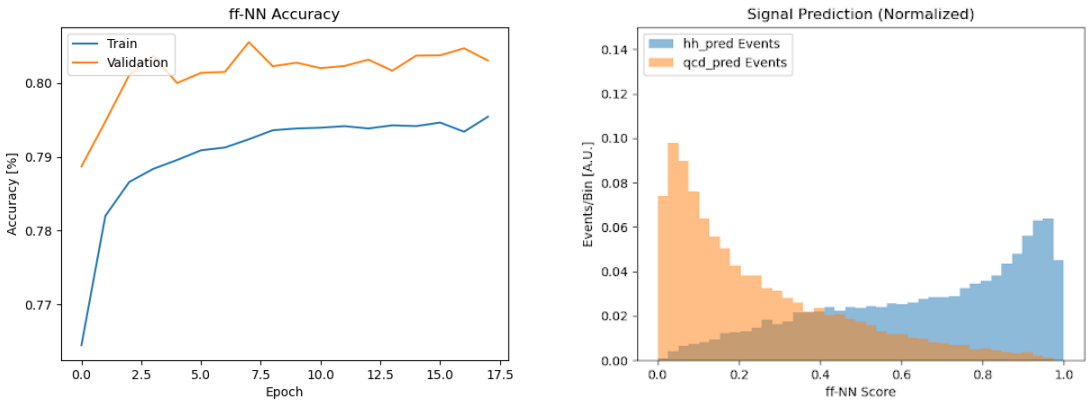


Figure 6. Structure of the feed-forward neural network. The input variables are fed through two fully connected dense layers to classify events. One dropout layer and one batch normalization layer help mitigate over-fitting during training.

The hyperparameters of the feed-forward NN (N_{nodes} in each hidden layer, learning rate, regularization, etc) were optimized by something more rigorous than trial and error. Describe the process semi in-depth and talk about what effect different parameters had on the overall performance. The model 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 $S/\sqrt{B} = 1.9 \pm 0.09$ when considering events with a signal prediction score > 0.89. This phase-space has a signal yield of 1303.73 events and a background yield of 467273.3 events.



(a) (b)

Figure 7. Accuracy (left) for the testing and training datasets and the final predictions (right) of the feed-forward network.

3.4 Convolutional Neural Network

Convolutional Neural Networks (CNNs) are neural networks that use assumptions about the local relationships between neigboring pixels in order to predict the content of an input image. For this analysis, content prediction is simplified to a general classification of whether the image comes from a dihiggs or QCD event. The fundamental elements of any convolutional network are convolutional layers and pooling layers. Convolutional layers use filters that perform non-linear combinations of neighboring pixels within the filter size, and pooling layers aggregate information by grouping neighboring pixels by either their maximum or average values. In a normal image processing context, each of these layers takes a two-dimensional shape as input and outputs a modified two-dimensional shape. 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 training convolutional networks using low-level measured quantities (e.g. tracks and calorimeter deposits) in high energy physics in the context of object identification [CITATION NEEDED]. This paper extends the application to event-level identification. Using such low-level quantities removes the need for reconstructing the event since no jets need to be selected or combinatorics performed to identify Higgs candidates. The performance of four convolutional networks is studied here in the context of dihiggs identification. The first network uses 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 information to the flattened vector after image processing and before the fully connected layers. Figures 8 and 9 depict both network strucutres. The third and fourth networks follow the same pattern as the previous two with the addition of two additional image layers corresponding to impact parameter-weighted track information (both longitudinal and transverse).

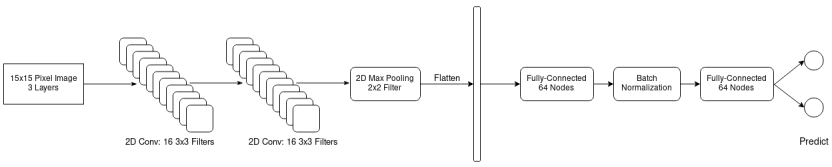


Figure 8. Structure of the nominal convolutional neural network. The input images are fed through two convolutional layers and a single max-pooling layer before being flattened into a one-dimensional vector. The flattened vector is then fed through one fully connected layer, a batch normalization layer,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

In order to produce coherent images, the center of mass and the center of momentum for each event are calculated. All consituents are then boosted longitudinally into the center of mass of the event and rotated in phi to the center of momentum. At the end, each image layer corresponds to a 31x31 pixel grid centered on the total activity in the event.

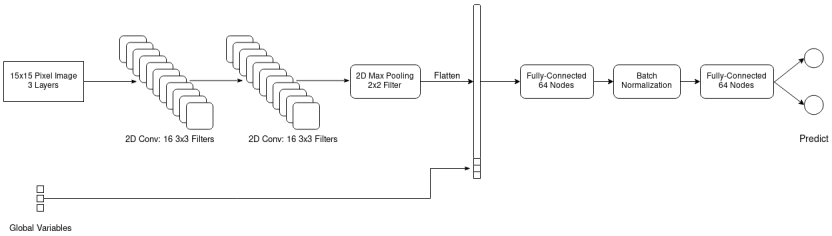


Figure 9. Structure of the hybrid convolutional neural network. The input images are fed through two convolutional layers and a single max-pooling layer before being flattened into a one-dimensional vector. Scaled user-specified variables (e.g. H_T) are then concatenated with the flattened image vector.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

Figure 10 shows an average QCD image and Figure 11 shows an average signal image. While the layers in each figure closely resemble one another, they do contain different information, and variations are visible. More importantly, the average QCD image differs significantly from the average dihiggs image.

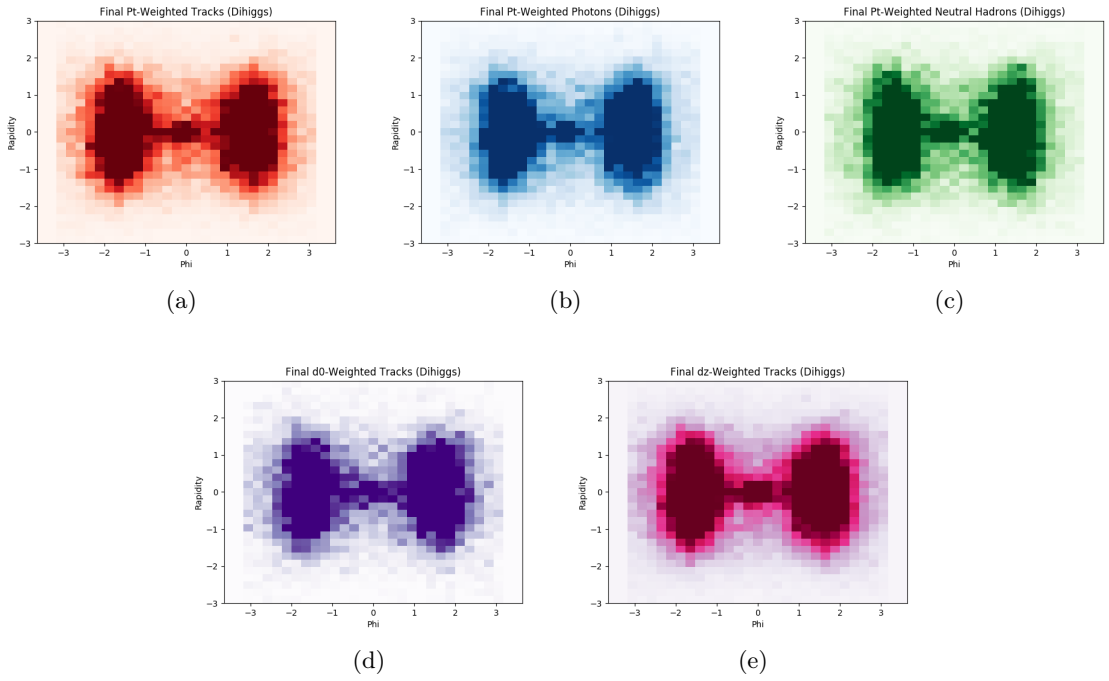


Figure 10. Average QCD image showing (a) pt-weighted tracks, (b) Et-weighted ECAL deposits, (c) Et-weighted HCAL deposits, (d) transverse impact parameter-weighted tracks, (e) longitudinal impact parameter-weighted tracks.

As shown in Figures 8 and 9, the CNN network structure uses two sequential 2D

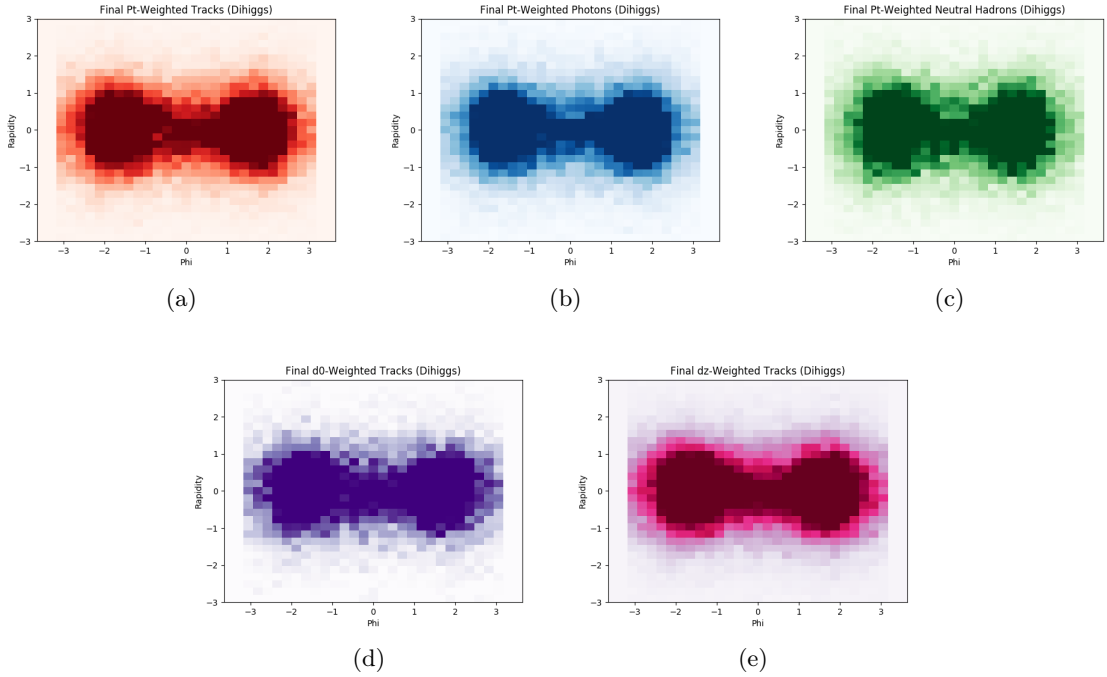
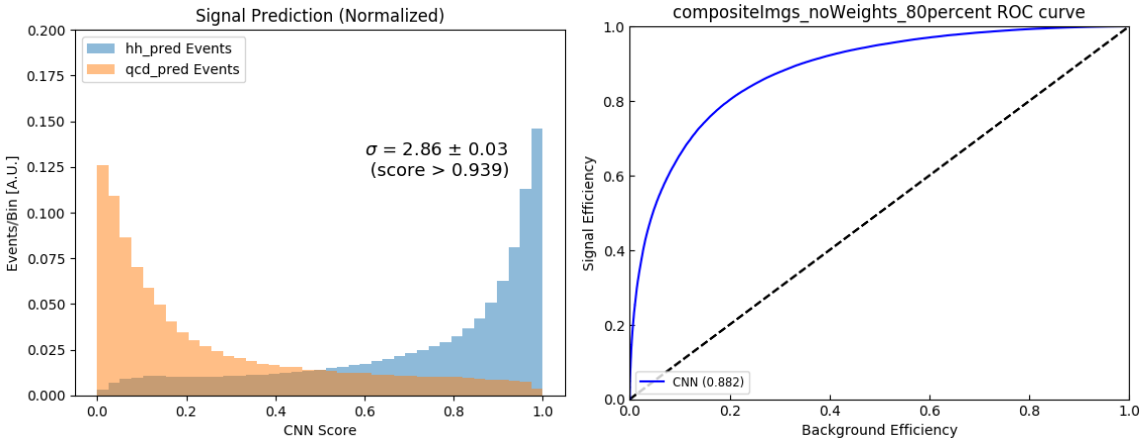


Figure 11. Average dihiggs image showing (a) pt-weighted tracks, (b) Et-weighted ECAL deposits, (c) Et-weighted HCAL deposits, (d) transverse impact parameter-weighted tracks, (e) longitudinal impact parameter-weighted tracks.

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 network structure 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. Optimal significances for each network are shown in Table 2 and the output signal predictions and AUC for the 5-color network with high-level inputs are shown in Figure 12.

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}



(a) (b)

Figure 12. (Right) signal prediction for the 5-color convolutional network with additional high-level inputs, (Left) AUC for the 5-color convolutional network with additional high-leve inputs.

3.5 Energy Flow Network

Energy Flow Networks (EFN) and Particle Flow Networks (PFN) are algorithms that take basic jet constituents information as input rather than reconstructed jets and multi-jet composites, e.g. Higgs candidates. The EFN structure takes only the rapidity y and azimuthal angle ϕ of jet constituents as input, while the PFN takes the rapidity y, azimuthal angle ϕ and transverse momentum p_T of jet constituents as input. Using the constituents as input, no high level reconstruction is necessary when identifying events similar to the CNN approach. 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 dihiggs classification use 200 nodes for each hidden layer in network (a), 256 for latent space dimension and 300 nodes for each hidden layer in network (b).

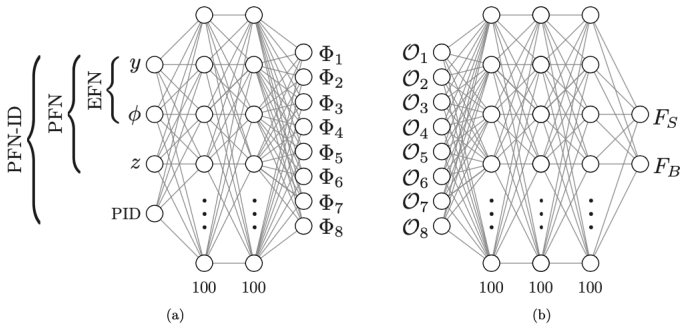


Figure 13. Network (a) takes jet constituents information as input and outputs latent space Φ for each jet constituents. Network (b) takes $\mathcal{O}$, which is the linear combination of Φ, as input and outputs final result

The EFN/PFN networks were trained using four seperate categories split by number of jets and number of b-tags in order to test the network performance 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 was adjusted to maintain an equal proportion of each population in the training sample. L2 regularization and dropout layers were added to minimize overfitting. The results obtained with the EFN are shown in Table 3. The results of the PFN are shown in Table 4.

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}

The PFN provided a best significance of 1.618 when trained over all events without any cuts on the number of jets or b-tags.

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. This type of approach is called semi-supervised learning because an algorithm identifies patterns in the data in an un-supervised way and is then evaluated using known classification information.

4.1 Auto-encoders

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. In principle this compression encodes the most important features of the training data, and the second half of the network 'decodes' this latent representation back into a representatino approaching the original inputs. This construction fundamentally changes the meaning of the loss function; 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 selected events with large losses. Monte Carlo simulations enable a semi-supervised cross-check on AE performance, meaning AEs may have exciting implications for signal detection in particle physics.

Because the AE uses semi-supervised/unsupervised learning, it is trained solely on QCD background and then tested with a mixture of background and dihiggs signal. Tests were performed substituting a pure-QCD training sample with a training sample with various proportions of dihiggs signal. No significant deterioration was observed for reasonable levels of contamination. The AE used for dihiggs detection was built using the keras package [4] and consists of an input layer, a single hidden layer, and an output layer. Eleven reconstructed variables obtained using the 'closestDijetMass' algorithm were selected for use in the AE. The number of hidden layers and the number of hidden nodes per layer are hyperparameters that were optimized. A PCA analysis shown in Figure 14 was used to determine that a latent space of three nodes was optimal.

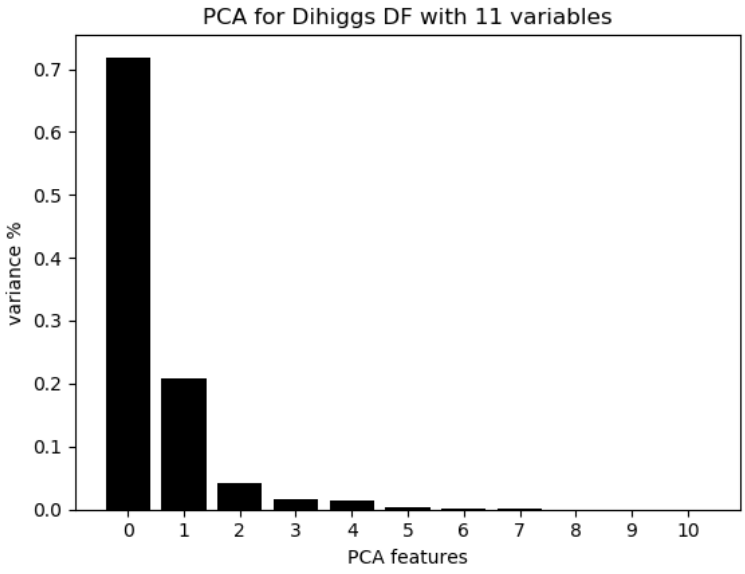
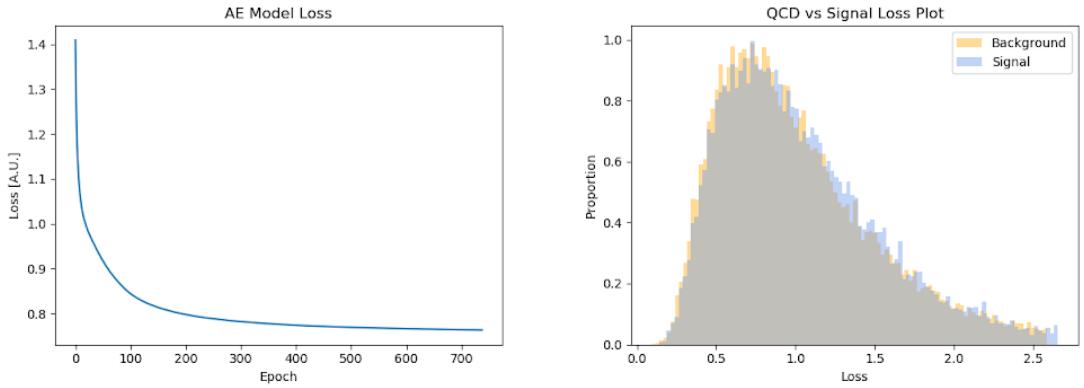


Figure 14. PCA performed on the selected eleven kinematic inputs. The x-axis indicates the number of PCA features, and the y-axis indicates the variance. Choosing the optimal size for the latent space requires identifying the point of diminishing variance return.

The output layer is a mirror of the input layer and therefore has 11 nodes. The hidden

layer and output layer use ReLU sigmoid activation respectively. An L2 regularization term was added to the hidden layer to avoid overfitting. Because training an auto-encoder is an unsupervised and unlabelled process, there is no prediction of whether a given event is signal or background. Instead of using a prediction to evaluate the efficacy of the AE approach, the loss output from a testing dataset is used as a proxy. Since dihiggs 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 supervised methods.



(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 dihiggs event data separately.

Training for several hundred epochs leads the model to converge, reaching a loss near 0.8 (see Fig.15). Requiring the loss to be larger that 0.1 yields a best $S/\sqrt{B}$ of 0.67±0.01. This is however somewhat misleading significance value may be misleading since the signal and background loss distributions have little separation and the highest significance result effectively is a cut that keeps nearly all events.

5 Results

The methods covered in this paper are by no means exhaustive of the ML landscape available to high energy physics, but a wide range of techniques and philosophies are covered. A weak observed trend is that the more complex the inputs are allowed to remain, the more information sophisticated algorithms can wring out of the data. This is less of a hard rule and more of a loose heuristic. Table 5 provides a summary of the methods described in the previous sections. The results of a traditional 1-D sequential cut techniques is shown for comparison though it has not been previously described.

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.

Method	0PU		
	Best $S/\sqrt{B}$	$\mathbf{N}_{\text{Signal}}$	$\mathbf{N}_{\text{Background}}$
Autoencoder	0.67 ± 0.01	5849.4	$7.7 \cdot 10^7$
1D-Rectangular Cuts	0.82 ± 0.XX	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	$2.8 \cdot 10^5$
Lorentz Boost Network	1.87 ± 0.08	1123.3	$3.6 \cdot 10^5$
Feed-Forward NN	1.90 ± 0.09	1303.7	$4.6 \cdot 10^5$
Random Forest	2.17 ± 0.22	-	-
Convolutional NN	2.85 ± 0.02	-	-

Table 5. Comaprison of method significance and signal/background yields normalized to full HL-LHC dataset of 3000 fb^{-1}.

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 a mismatch between problem and proposed solution rather than a statement on the use of unsupervised techniques in general.

6 Conclusions

Measuring the rate of dihiggs production will be a problem facing the high energy physics community until the end of the HL-LHC era. The techniques explored in this paper show the power of machine learning techniques to identify dihiggs signals from amongst overwhelming QCD signals. The significance obtained for many of these methods is impressive given the simplicity of the evaluation metric and bodes well for current and future measurements of double higgs prodcution at the LHC.

References

[1] J. Alwall, R. Frederix, S. Frixione, V. Hirschi, F. Maltoni, O. Mattelaer, H. S. Shao, T. Stelzer, P. Torrielli, and M. Zaro, "The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations," *JHEP*, vol. 07, p. 079, 2014.

[2] J. de Favereau, C. Delaere, P. Demin, A. Giammanco, V. Lemaître, A. Mertens, and M. Selvaggi, "Delphes 3: a modular framework for fast simulation of a generic collider experiment," *Journal of High Energy Physics*, vol. 2014, Feb 2014.

[3] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," *CoRR*, vol. abs/1603.02754, 2016.

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

[5] M. A. et al., "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *CoRR*, vol. abs/1603.04467, 2016.