

Florida State University Libraries

2017

Exploration of Deep Learning Methods for Vector Boson Fusion Event Discrimination

Alejandro Mario Sanchez



THE FLORIDA STATE UNIVERSITY
COLLEGE OF ARTS & SCIENCES

EXPLORATION OF DEEP LEARNING METHODS FOR VECTOR BOSON FUSION
EVENT DISCRIMINATION

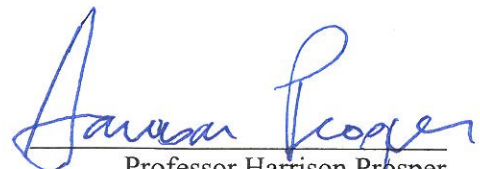
BY


ALEJANDRO M. SANCHEZ


A Thesis submitted to the
Department of Physics
in partial fulfillment of the requirements for graduation with
Honors in the Major

Degree Awarded:
Spring, 2017

The members of the Defense Committee approve the thesis of Alejandro M. Sanchez defended on April 26, 2017.


Professor Harrison Prosper
Thesis Director


Professor Anke Meyer-Baese
Outside Committee Member


Professor Todd Adams
Committee Member

Exploration of Deep Learning Methods for Vector Boson Fusion Event Discrimination

Alejandro M. Sanchez

Thesis Director: Harrison B. Prosper

Department of Physics, Florida State University

Abstract

In the past, the discrimination of Higgs boson events from background events has been done with the introduction of clever variables to make discrimination easier for machine learning methods. By using a complex neural network with two hidden layers and a stochastic optimizer algorithm called *Adam*, I can discriminate Higgs Boson events from non-Higgs events with a receiver operating characteristic (ROC) score of 0.961 and a validation accuracy score of 94.4% using the raw data alone. I also attempted to discriminate those Higgs events that were produced by the process of Vector Boson Fusion (VBF), and achieved a ROC score of 0.858 and a validation score of 79.7%. Although not as high a score as the Higgs/non-Higgs discrimination, the correct choice of raw data variables outperformed the historical clever variables used for this discrimination task. These results confirm something that it intuitively obvious, namely, that all the information about events is contained in the raw data and can be extracted directly with a sufficiently smart machine learning algorithm. A lot of time currently goes into designing clever variables for these purposes, but, in the future, computers will be able to discriminate with no need of these clever variables, allowing researchers to spend less time designing complicated, clever, variables and more time inventing better ways to analyze data for new physics.

1 Introduction

High energy physics (HEP) is the study of the fundamental constituents of matter and the forces between them. The basic research method in this field involves colliding particles at high energies in large particle accelerators and studying the debris from those collisions, called events. For instance, the Large Hadron Collider (LHC) at CERN currently collides protons at energies of 13 TeV with various independent collaborations of scientists using the facility to do their research via large complex experiments. The Compact Muon Solenoid (CMS) is one of these experiments and comprises a large detector used to carefully track the debris that comes out of the resultant proton-proton collisions. The CMS collaboration uses the data from these events to test predictions of the Standard Model (SM) as well as look for new physics beyond the SM such as searching for Supersymmetry (SUSY) and Dark Matter.

The Standard Model of particle physics is a theory that effectively describes the most fundamental level of the Universe. By combining special relativity and quantum mechanics, it provides a description for the behavior of subatomic particles, the fundamental constituents of matter, and the interactions between them. It has successfully predicted the existence of several particles, such as the top quark, the W and Z Bosons, and the Higgs boson. The Higgs boson was the last of these to be discovered, in 2012, and with its discovery the SM is now considered complete. The Higgs boson is an important find because it explains where the fundamental particles get their mass from.

1.1 The Higgs Boson

Without the Higgs boson, the SM would require all particles to be massless to maintain its symmetry. The Higgs mechanism was originally proposed to account for the masses of the W and Z bosons, which

are known to have mass from previous experiments. However, the Higgs mechanism also provided an explanation for the mass of the other massive fundamental particles, the charged leptons and quarks. The Higgs mechanism works similarly to a person trying to run while having their legs submerged in water. Massless particles, such as photon and gluons, do not interact with the Higgs field, so they move as fast as they can: the speed of light. However, particles with mass are those that do interact with the Higgs field mediated by the Higgs boson, and in this case the field drags on the particles and slows them down, not letting them to go at the speed of light and giving them the property of mass [1].

What exactly is measured and analyzed in most HEP experiments is the cross section of these events. The cross section is the effective target area in which two particles must meet for them to scatter a certain way. In HEP, the cross section is proportional to the probability of an event happening, so it is an indirect way of measuring the likelihood of an event. A lot of information can be derived from cross sections and comparing them to one another. For example, in quantum chromodynamics (QCD), the experimental evidence for the existence of only three colors, the charges of the strong nuclear force, was given by the ratio of two cross sections agreeing with the prediction of that ratio for only three colors [2].

Now that the Higgs boson has been discovered, the next step is to find out what properties it has and how those compare to predictions. If the results we get agree to what the SM predicts, then we know that we are on the right track to understanding the behavior of these processes. If they don't, then that means that there is new physics waiting to be uncovered. As the data from CMS is delved into deeper, the cross sections that we want to measure get smaller, making them harder to measure with precision. Although difficult, there is still important information to be collected from all processes, as any deviation from the SM could provide some new insight. The most difficult part of measuring the cross sections is sorting the events properly, making sure the correct events are picked out and counted. With the overwhelming amount of data taken at CERN, we must rely on computers to discriminate these events.

1.2 Vector Boson Fusion

One of my goals was to study one of the production modes for the Higgs boson, namely, Vector Boson Fusion (VBF), and measure its cross section. VBF is the process of two quarks each emitting a vector boson, W or Z boson, which then fuse together to create a Higgs boson [3]. The two quarks then scatter, producing an avalanche of hadrons, composite particles made of quarks and gluons, which are then measured by the detector. This avalanche of hadrons caused by high energy quarks is known as a jet. The Higgs boson itself is a short-lived particle, which has several decay modes, but I will only be studying events, so-called channels, with a final state of a Higgs decaying into two Z bosons, which then decay into two leptons. So, the data that makes it to the detector are the four leptons and the two jets.

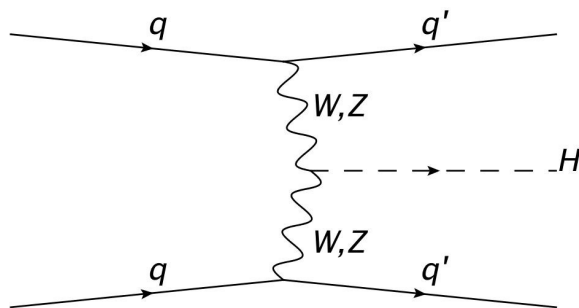


Fig. 1: Vector Boson Fusion Feynman Diagram. Two quarks scatter by exchanging vector bosons which then fuse to form a Higgs boson. Two final state quarks then produce jets which make it to the detector.

What makes VBF so interesting to study is that it is an uncommon production mode for the Higgs boson. The VBF channel accounts for roughly 10% of the total number of Higgs events. The rest of

Higgs boson production events consist of other production channels, such as gluon-gluon Fusion (ggF).

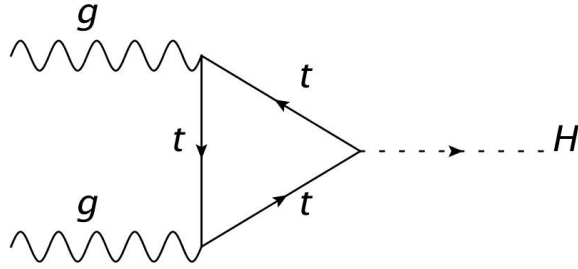


Fig. 2: Gluon-gluon Fusion Feynman Diagram. Although this process itself does not produce jets, other processes during the event could create resultant jets which then make it to the detector.

Higgs production itself is a rare event with a very low cross section, so discriminating VBF events is pushing the limits of our current ability to measure small cross sections. My goal is to show that the use of raw information together with state-of-the-art machine learning is better able to pick out these events and thereby measure their cross section. Getting a good measurement of the cross section of VBF can also be of great scientific value in frontier research [4] [5].

1.3 Neural Networks

Machine learning techniques, such as neural networks (NN), have proven to be an effective tool in high energy physics since the 1990s. Neural networks work by taking a set of inputs and going through a multilayer process in which the input set is passed to several nodes. Each node calculates a value which is a linear combination of the inputs passed to it plus or minus some constant, and then transformed by some activation function, such as a hyperbolic tangent. So, each node produces a value such as the following:

$$y = \tanh\left(b + \sum_{i=1}^N a_i x_i\right), \quad (1)$$

where x_i denotes the initial inputs, a_i is the factor each term is scaled by, known as a weight, and b is the constant term, known as the bias.

After passing the inputs through various independent nodes, each with its own a_i 's and b 's, you now have a new list of values derived from the old ones, which can now be used as inputs for another iteration, or layer. Neural networks have an input layer, the initial inputs; an outer layer, the final answer, and one or more hidden layers, the in-between layers that do much of the work. Here is an example of a mathematical representation of a NN with one hidden layer using a $\tanh x$ activation function (note: the output layer may or may not have an activation function) [6].

$$f = b + \sum_{j=1}^M v_j \tanh\left(a_j + \sum_{i=1}^N u_{ij} x_i\right). \quad (2)$$

Here, a_j and b are the biases, and u_{ij} and v_j are the weights. For the case of discrimination between signal and background, a network may be passed through an additional sigmoid activation function to restrict the output between 0 and 1.

$$y = \frac{1}{1 + e^{-f}}. \quad (3)$$

The training of these networks involves the tweaking of the biases and the weights based on a training data set that teaches the network what to look for. This is usually done by minimizing a loss function that indicates how far away the network output is from the targets (that is, the desired outputs)

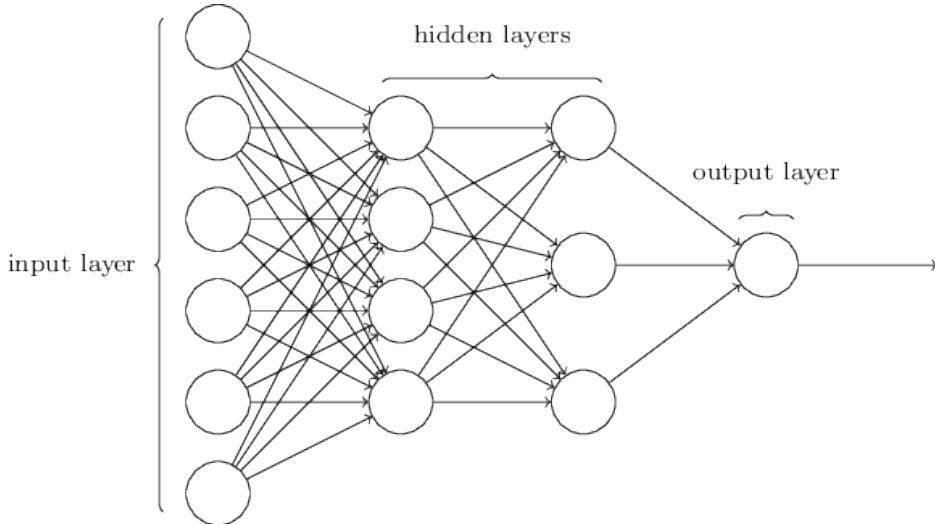


Fig. 3: Visual representation of a neural network (NN). An NN with multiple hidden layers is often called a multilayer perceptron (MLP) [7].

given by the training data. The most common and well understood form of training algorithm is called stochastic gradient descent (SGD). This algorithm works by calculating the gradient of the loss function with respect to the weights and biases and then moving these parameters in the direction opposite the gradient until a minimum is reached or some other criterion to tell the network to stop training. It is wise to stop training past a certain point, because one risks over-training, that is, over-fitting the training data and end up losing some of the features one is trying to capture.

Neural networks have made it possible to automate a large part of the event discrimination process, separating signal events (those that one is interested in) from background ones. Over the past two decades, NNs have increased in accuracy as our understanding of them and the processing power of computers have increased. However, as it stands now, the training of NNs remains a long, computationally demanding process. First, a lot of effort goes into generating simulations of how the events and the data should look, so one can effectively teach a computer to tell which event is which. Second, a lot of work is devoted to finding ways to get the training data in a form that makes training NNs feasible.

1.4 Clever Variables

Transforming data into clever variables that make it easier for a network to discriminate between events is a nontrivial problem that requires a lot of human time and effort. The discrimination of Higgs and non-Higgs events provides an example of clever variables that were derived to make training easier in machine learning methods.

The data in their simplest form are the 4-vectors of the particles that make it to the detector, describing the states of these particles. Since the observed particles are effectively massless compared with the energy scale of the LHC, these 4-vectors can be described with three numbers for each final state particle: p_T , the magnitude of the momentum in the transverse direction; η , the pseudorapidity, a simple rebranding of the polar angle θ such that

$$\eta \equiv -\ln \left[\tan \left(\frac{\theta}{2} \right) \right]; \quad (4)$$

and finally, ϕ , the azimuthal angle. For Higgs discrimination, only the states of the four leptons are needed, and with three numbers describing each state, one gets a total of 12 numbers to describe the system.

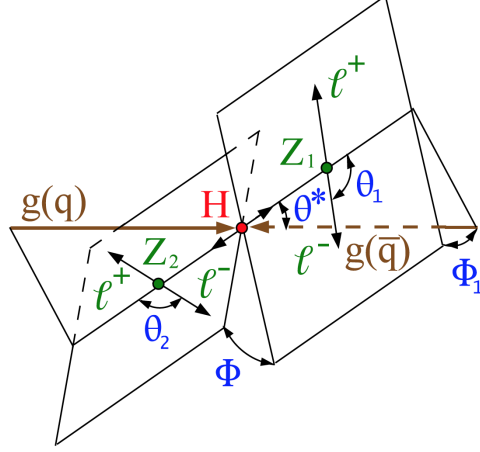


Fig. 4: Visualization of the clever variables commonly used in Higgs boson event discrimination in the channel in which the Higgs boson decays to two Z bosons, which subsequently decay to charged leptons. The seven clever variables are m_{Z_1} , m_{Z_2} , the masses of the two Z bosons, and the angular variables θ^* , θ_1 , θ_2 , Φ_1 , and Φ . See main text for explanation.

In the past, these 12 variables have been transformed into seven variables cleverly derived from the initial 12 to make training easier, as shown in Figure 4. The first two variables are the invariant masses, m_{Z_1} , m_{Z_2} , of the Z bosons Z_1 and Z_2 that decay from the Higgs boson. In the rest frame of the Higgs boson, these Z bosons will fly off in opposite directions along the same line. The angle between the axis formed by Z_1 and the beam axis is called θ^* . Now we move to the rest frame of Z_1 , which decays into a positive and a negative lepton. The initial direction of Z_1 and the axis of the decay products in the rest frame of Z_1 make a plane in space. The angle that this plane is rotated away from the beam axis (or alternatively about the Z_1 direction) is denoted by Φ_1 . The angle that the direction of the negative lepton makes with the direction of Z_1 is denoted by θ_1 . Similarly, for Z_2 , the angle between its direction and its negative lepton is θ_2 . Finally, the rotational angle between the decay planes is denoted by Φ .

A similar approach has been taken for the discrimination of VBF events. Instead of looking at the leptons in these events, researchers have looked at the resultant jets. The clever variables used for the jets in this case are the combined mass of the two jets, m_{jj} , and the angular separation between them, δ_{jj} .

Most NN research has been done with NNs containing one hidden layer due to computational limitations and the complex structure of networks with many layers of nodes. However, in 2010, machine learning researchers Cireřan *et al.* demonstrated that it is feasible to train deep NNs (networks with multiple hidden layers, also called multilayer perceptrons or MLPs) using a standard gradient descent algorithm that achieves higher accuracy than more sophisticated methods with less effort on the part of the user [8]. In their paper, they demonstrate the effectiveness of a deep network using the MNIST database, a large database of handwritten digits commonly used to test the accuracy of machine learning methods. By implementing deep networks, they achieved an accuracy of 0.35% error, only getting 35 of the 10,000 test data inputs incorrect. Moreover, of those 35, the MLP was correct on its second guess 30 times.

In previous NN research, record accuracies on MNIST were achieved by the transformations of the raw input variables into clever ones, such as looking for straight lines or curves in the digit data. What is promising about Cireřan *et al.* is that they achieved record accuracy using the data in its raw form, the grayscale value of the pixels representing the digit on a 28x28 image. Using deep networks, they showed that the training of networks using raw data is effective, with less effort on the part of the user. The goal of my project was to see if this is true also for VBF event discrimination.

1.5 Adam SGD Optimizer

Training deep neural networks is computationally demanding. Therefore, it is of interest to make use of the most effective training algorithms. I decided to use a recently published algorithm called *Adam*. *Adam*, a name derived from adaptive moment estimation, is an algorithm introduced by Kingma and Ba built on the foundation of optimizers for the traditional SGD algorithm [10]. *Adam* is based on adaptive estimates of the moments of the network. It is a simple and intuitive algorithm that is computationally efficient and works with little memory. In summary, it works by calculating the exponential moving averages of the first moment, the gradient, and the second moment, the squared gradient (squaring the gradient term by term), and moving with a step size proportional to their ratios. As the loss function to be minimized approaches its minimum, this ratio goes to zero and the training automatically becomes more fine-tuned.

Algorithm 1 *Adam* SGD Optimizer Algorithm as described by Kingma and Ba. **Passed:** $f(\theta)$, Function being optimized; θ_0 , Initial parameter vector. **Hyperparameters:** α , stepsize; β_1, β_2 , exponential decay rates for first and second moment vectors, respectively; ϵ , small number to avoid dividing by 0. **Good default settings:** $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$

```
 $m_0 \leftarrow 0$  (Initialize first moment vector)
 $v_0 \leftarrow 0$  (Initialize second moment vector)
 $t \leftarrow 0$  (Initialize timestep)
while  $\theta_t$  not converged do
   $t \leftarrow t + 1$ 
   $g_t \leftarrow \nabla_{\theta} f(\theta_{t-1})$  (Compute gradient with respect to parameters)
   $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$  (Update moving average first moment)
   $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$  (Update moving average second moment)
   $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$  (Compute bias-corrected first moment)
   $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$  (Compute bias-corrected second moment)
   $\theta_t \leftarrow \theta_{t-1} - \alpha \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$  (Update Parameters)
end while
```

The full proof behind the expressions for the moving averages, m_t and v_t , is found in the paper by Kingma and Ba. The gradient moments \hat{m} and \hat{v} are computed to account for the bias of initializing the moments to zero. What makes this algorithm so powerful is the fact that by using moving averages to compute how far to move in parameter space, it is at less risk of falling into a local minimum, a problem that becomes more prominent with increasing complexity of the network structure. Their algorithm has been shown to be especially useful in training large data sets.

For the purposes of my project, I used an instance of the *Adam* algorithm found in the `scikit-learn` 3rd party Python library for machine learning algorithms [9]. The algorithm excels at handling large data sets, such as the ones found in HEP. `scikit-learn` provides a multitude of pre-built and easy-to-use tools for machine learning. For this project, I used their class `MLPClassifier`, a pre-built class designed to fit a multilayer NN to a set of training data passed. This class also boosts computation time by estimating the gradient calculations through the use of mini-batches, where it selects a small subset of the data to perform the gradient calculations.

2 Methods

My goal was to see if, similarly to Cireřan *et al.* and their work with MNIST, it is feasible to train networks using raw data instead of cleverly derived variables by utilizing more complex network structures for the VBF process. The clever variables used in the past are nothing more than functions of the originals, so it stands to reason that all the information should be contained in the original variables. Instead of using simple NNs with one hidden layer, I trained with a network containing two hidden layers. Due

to increasing computational power over the years and the development of better training algorithms, the training of a complex MLP has become a reasonable task. I tested the power of an MLP by showing how it performs in collider event discrimination using the data in its raw form.

In principle, the discrimination of events should be a simpler task than training for handwriting recognition. First, the number of inputs is smaller by almost two orders of magnitude, consisting of 12 variables instead of the overwhelming $28 \times 28 = 784$ inputs used in MNIST. Second, event discrimination is binary; an event can be classified as signal or background. For MNIST, there were ten outputs to be calculated, the probabilities for a set of inputs to be any of the digits, 0 through 9. Third, although events of the same kind can manifest themselves in various ways, their behavior is described by laws of physics that are more constrained than the various styles of human handwriting.

2.1 Data Used

The training was done using synthetic data created by Monte Carlo simulations, used to mimic the data found in the real experiment, so we know what to look for once the real data are gathered. The total number of simulated events were split into two randomized subsets, 75% to be used for training and 25% to be used for validation. The original events have weights assigned to each of them to show how likely they are and therefore how much priority they should be assigned. Since the networks used treat all training data with equal weight, I ran experiments using both the weighted events (but assigning each of them equal weight), and the same set of events unweighted so they each carried a weight of 1. Unweighting events consists of picking out events at random from the weighted set, with the probability with which events are picked proportional to their weights. (This means that there is a chance that a weighted event shows up twice in an unweighted file. This sampling method, which preserves the probability distribution of the weighted events is called *sampling with replacement*.) Since events are picked at random, it is crucial to separate the training and validation events first before unweighting them. This way, the training data set is completely independent of the validation set, and an event found in one data set, weighted or unweighted, will not be found in the other. In order to test the effectiveness of this method in real application, the trained networks were fed a small set of real data that was obtained during the most recent runs by the CMS collaboration in 2016.

2.2 Network Settings

Using `sklearn.neural_network.MLPClassifier` and the *Adam* algorithm, various tests were performed to determine how accurate of a discrimination was possible by using raw data. All networks trained contained two hidden layers, the first with 100 nodes and the second with 50 nodes, and binary output containing the probability of background and the probability for signal. Although there are two outputs, their probabilities always add to 1, so I treated the signal probability as the one effective output of the networks. Therefore, the network structure for all networks used are $n_{inputs} \rightarrow 100 \rightarrow 50 \rightarrow 1$ where n_{inputs} denotes however many inputs are being taken in by the network.

The networks used the hyperbolic tangent as the activation function, as is standard in previous HEP discriminants. In trying out several settings, the variable `early_stopping` was set to either `True` or `False` throughout the course of the project. When set to `True`, the network initially sets aside 10% of the training data given to it at random, and uses it to validate the current iteration of its training. With this subset a test validation score is calculated, the fraction of guesses on the test validation subset it got correct. Once the test validation score has not improved by `tol` for two consecutive iterations, training is considered complete, and the training will stop. If `early_stopping` is set to `False` as is the default, the network will keep the training data intact and instead compare the value of the loss function to the same tolerance `tol`. In principle, one could let the network keep training and it would keep lowering the error function until it reaches 0 error on the training data, but after a point one risks over-fitting the network to the data and accuracy on the validation set begins to decrease. Both values of `early_stopping` were tested to check for the possibility of over-training.

2.2.1 Default Settings

Relevant default settings of `sklearn.neural_network.MLPClassifier` that were left untouched for the purposes of this project are:

- **solver**: *Adam* (Training algorithm)
- **alpha**: 0.0001 (Stepsize)
- **batch_size**: 200 (Size of mini-batches for training)
- **max_iter**: 200 (Maximum iterations of network before stopping)
- **random_state**: None (State or seed for random number generator)
- **shuffle**: True (Whether to shuffle training data in each iteration)
- **tol**: 1e-4 (Tolerance for training. When the loss function or test validation score has not improved by tol for two consecutive iterations, training will stop)
- **warm_start**: False (If True, reuse solution from previous fit as initialization)
- **validation_fraction**: 0.1 (If `early_stopping` is True, proportion of training data set aside as a test validation set)
- **beta_1**: 0.9 (Exponential decay rate for first moment vector in *Adam*, should be within [0,1))
- **beta_2**: 0.999 (Exponential decay rate for second moment vector in *Adam*, should be within [0,1))
- **epsilon**: 1e-8 (Value for numerical stability in *Adam*)

2.2.2 Modified Settings

The following parameters were set:

- **hidden_layer_sizes**: (100, 50) (ith element represents the number of neurons in the ith hidden layer)
- **activation**: 'tanh' (Activation function used in hidden layers)
- **verbose**: True (Print progress messages)
- **early_stopping**: Set to both True and False (Whether or not to train to the loss function (False) or to set aside a test validation set and train to that (True))

2.3 Testing Network Effectiveness

Two methods were used to quantify the effectiveness of the networks being trained. The first is a receiver operating characteristic curve (ROC curve), a graphical plot that intuitively demonstrates the performance of a binary classifier. It is a parametric curve with the parameter being the threshold for the classifier. For a specific threshold, the false positive rate (FPR) is plotted on the x-axis and the true positive rate (TPR) is plotted on the y-axis. The false positive rate is the fraction of the total background events found above this threshold, and the true positive rate is the fraction of signal events found in the same region. A numerical value for accuracy can be obtained from the area under the ROC curve.

A discriminant with no power (guesses at random whether an event is signal or background) would show up as a straight diagonal line in a ROC plot and a ROC score of 0.5, since it would randomly guess correctly half of the time. As discrimination becomes more powerful, this curve will move towards the upper left corner of the plot, with the points (0,0) and (1,1) fixed. At one extreme, no signal is above the threshold, but no background either. At the other extreme, all the signal is above a threshold, but so is all the background. ROC curves are also useful for selecting an appropriate threshold with a given discriminant. What's nice about ROC curves is that not only do they give a numerical value representative of discrimination accuracy, but they also give an intuitive visual representation of the

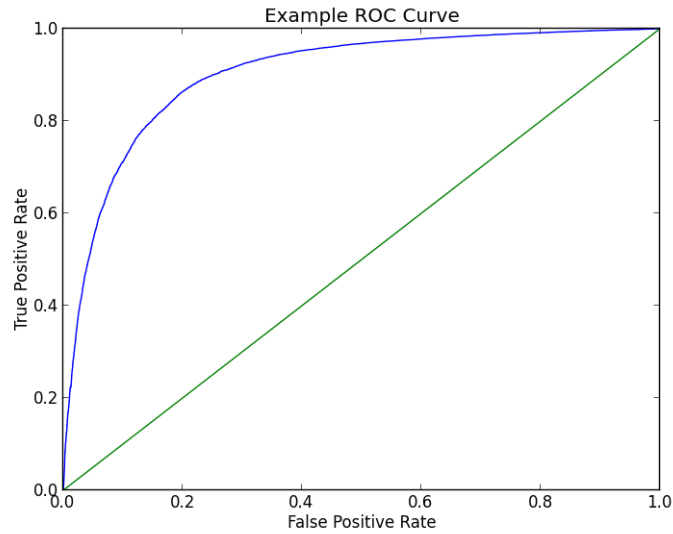


Fig. 5: Example of a ROC curve. It is a parametric function of the fraction of signal (y-axis) and the fraction of background (x-axis) above a certain threshold. The line $y = x$ demonstrates no discrimination power, and the larger the area under the ROC curve, the more accurate a classifier is considered.

same accuracy. As accuracy increases, the curve is stretched more towards the upper left of the plot, with perfect discrimination represented by an L-shaped plot going from (0,0) to (0,1) to (1,1).

After the training of the network, the 25% of the data set aside for validation was used to create these ROC curves, showing the discrimination power of the trained network on a set of points that is totally independent of the training. Once the network is trained, the function `MLPClassifier.predict_proba()` is used to determine the probability that an entry is a signal. This function returns two values for each entry, the probability for it to be a background entry, and the probability for it to be a signal. Since for binary discrimination, these values are dependent on one another, only the signal probability is considered. `scikit-learn` also provides its own functions to generate ROC curves and to calculate the area under them, `sklearn.metrics.roc_curve()` and `sklearn.metrics.auc()`, respectively. `roc_curve()` is passed two lists, one containing the target (0 for background, and 1 for signal) and the other containing the signal probabilities for the respective events. It returns the points for the false positive rate, the true positive rate, and the threshold. These are the x-values, y-values, and parametrization of the ROC curve, respectively. `auc()` is passed the FPR and TPR points, conveniently already given by `roc_curve()`, and calculates the area under the curve. It is also passed the parameter `reorder=True`, to avoid errors if two consecutive x-values are the same. This should only be set to `True` when dealing with ROC curves.

The second method used to test accuracy was the accuracy score of the validation data set. Simply, the validation data was passed into the network, and it would calculate the fraction of guesses the network got correct. Once again, there should be no bias in the validation score from the training. This value is obtained from the function `MLPClassifier.score()` which is passed the validation inputs and their respective targets.

Both `roc_curve()` and `MLPClassifier.score()` have the ability to account for entry weights as optional parameters, so all validation data is weighted, and the functions above are both passed a list with weights of the events.

2.4 Training

The training of the networks was performed using various versions of the training data to explore what provided the best accuracy with this network. The first two tests were performed on networks discrimi-

nating between Higgs and non-Higgs events. The variables being used for this were the 12 raw variables from these events as opposed to the seven clever ones used historically.

The first test that was performed was to see which signal-to-background ratio (SNR) would provide the most accurate results. The original training data contains an SNR that mimics the ratio that is expected in the real experiment. While this SNR may appear differently in nature, training a discriminant might benefit from using a different SNR. Three sets of the training data were created and trained. The first with an SNR of 1, equal numbers of signal and background. This set was made by reducing the number of background data in this set to match the number of signal data available. The second set contained a realistic SNR, however, it was reduced to match the number of data in the first set in order to ensure that any difference between the two was solely due to their respective SNRs. The third set was the control set, using all the data available with the original SNR given. Since the first two sets were created by truncating the original, the unweighted data sets were originally used for this test to avoid wild fluctuations from the weights of the events.

The second test performed was to test which value is best to use to determine when to stop training. The networks were run with equal settings and training data, changing only the parameter `early_stopping` to either `True`, for training to the test validation score, or `False`, for training to the loss function. Training to the loss function will allow the network to be trained for longer iterations, however, one runs into the risk of overtraining since the network is testing its validity using only the training data.

Using the best choice of SNR and tolerance function to train, the signal and background in the validation sets were separated and plotted in a histogram showing the distribution of signal probabilities for each of them. The CMS data were also plotted to see if the patterns in the histograms matched.

The third test performed tested if unweighting the training events is actually beneficial. For validation, it makes sense that weights are crucial to determine the realistic accuracy of the trained networks, however, the training function, `MLPClassifier.fit()` does not provide an option to account for the weights in the events given to it. So that leaves the user with two options: train with the weighted events that contain wildly varying degrees of importance, or unweight the events so they are all of equal importance and then proceed to do the training. Unweighting the events comes at a cost of reducing the size of the training data set by a small chunk, so it is initially uncertain if it is beneficial to unweight the events beforehand or not.

Once the most appropriate SNR and choice of weights was determined from Higgs to non-Higgs discrimination, the next step was to determine the best choice of variables for discriminating between VBF Higgs production and non-VBF events. For this test, only events from the Higgs boson data set were considered. Historically, the clever variables used for VBF discrimination are m_{jj} and δ_{jj} , two variables derived only from the resultant jets. So, it is uncertain which choice of raw variables will produce the best results for discrimination. Perhaps the 4-vectors of the leptons, the same 12 variables as before, contains enough information for discrimination, or the additional 4-vectors of the jets are required, bringing the total number of inputs to 18. Another possibility is that only the 4-vectors of the jets are needed, reducing the required inputs to 6, and the 12 lepton variables will only hinder the discriminant by adding unnecessary noise. These three choices of raw variables, as well as the clever variables for VBF, give four versions of the training data to feed into networks and see which one works best. In these data sets, the values for p_T , η , and ϕ assigned to a jet if one was not detected is -999.0. These networks were trained using both the full data set with these values left in, and also using a reduced set applying a selection rule requiring that all events in training and validation have two jets.

Finally, once a good choice of variables for VBF discrimination was produced, the two discriminants were combined to see what kind of resolution could be acquired from all of the data, which is how these discriminants would be used in the future to discriminate between real data. Using the best SNRs, weights, and choice of variables as determined in the previous tests, two discriminants were made and plotted in 2-dimensional histograms, with Higgs/non-Higgs boson signal probability on the x-axis,

and VBF Higgs/non-VBF Higgs signal probability on the y-axis. Since the first discriminant doesn't care about jets, it was computed using all the data, regardless of the number of jets. The more events used in training, the higher the accuracy, so it is not worth making an unnecessary cutoff for this one. For the second discriminant, and later the validation data, only events that had produced two jets were considered. The validation events were then separated into subsets of their true value and plotted to see the discrimination obtained from each of them. The data from CMS are also plotted as a histogram to see how they behave under these discriminants.

3 Results

3.1 Higgs vs. non-Higgs Discrimination Tests

The ROC curves made for all Higgs vs. non-Higgs discrimination tests include the results achieved from each of the various SNR sets. Those labeled as "SNR = 1" have equal numbers of signal and background. Those labeled as "Realistic SNR" have the same SNR as "All Data", but have the number of events reduced to match the total event count for "SNR = 1". Therefore, the first test is accounted for in the plots for the second and third test. This first test was conducted using the unweighted data in order to not worry about the significance of events being trained during each run. Each line shown in the graphs below corresponds to an independent network trained separately from the rest. The quantity "auc" represents the area under the ROC curve, and "score" represents the validation score, both from the 25% subset set aside. Each network configuration was run 10 times, although I will only be presenting a select few of these graphs, given that most graphs appear very similar. The graphs I've chosen to show are not outliers, and represent the patterns observed throughout all runs.

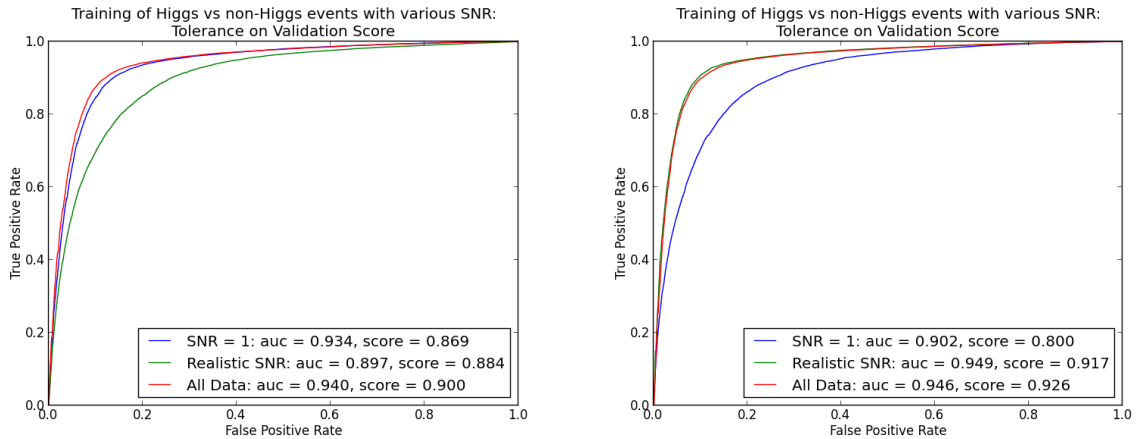


Fig. 6: Two different runs of the network training to the test validation scores. It is apparent that the accuracy of a particular set varies significantly from one run to the next. Note that the test validation score is calculated from a small subset of the training data and is not shown in the final plots, which are computed using a separate validation set.

Here are the results obtained from the training of a discriminant using either the test validation score or the loss function. For the networks being trained to the test validation score, the accuracy values obtained from the networks varied a significant amount from run to run. On the other hand, training using the loss function produced plots that were much more consistent between runs, and also consistently performed better than its test validation score counterparts.

Noting that the loss function showed better results during training, all future tests were done by placing a tolerance on the loss function, that is, `early_stopping` set to `False`. Then, I moved on to test what happens if I introduce weighted events into the training data.

The trainings performed with weighted data as opposed to unweighted data showed an accuracy

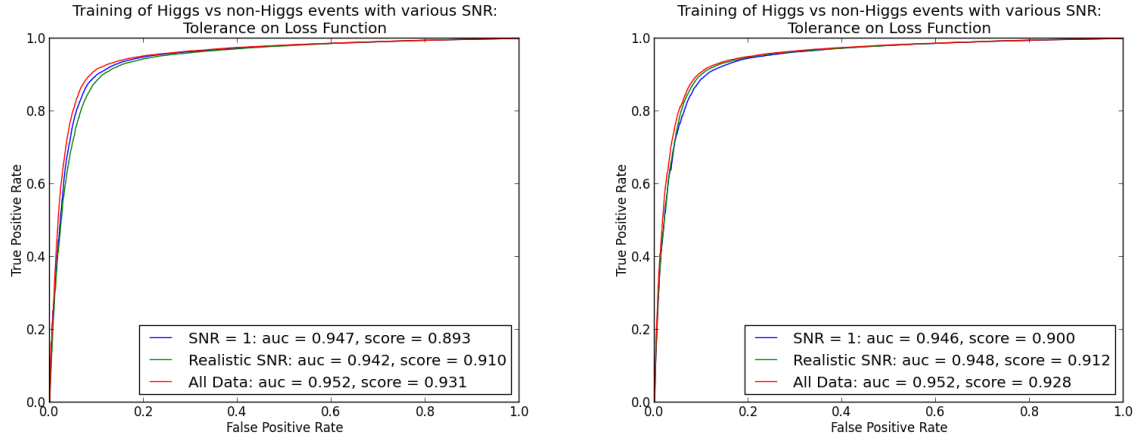


Fig. 7: Two different runs of the network training to the loss function. The trainings are more consistent from one run to the next, and also show higher scores than their validation score counterparts.

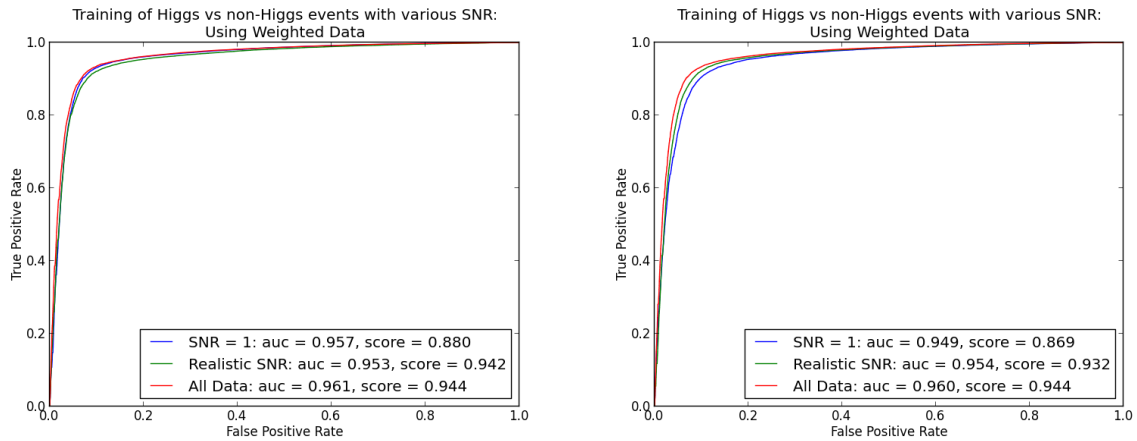


Fig. 8: Two runs of network training, now using the weighted data instead of unweighted data to train. Again, there is an accuracy increase from the previous test.

gain of roughly 1% for all training data sets used. Therefore, the tests moving forward were trained using weighted data instead of unweighted data. It is also apparent from all previous tests that the training data set containing all the data, the red curve, produced the most consistent accurate results. The blue and green curves performed with similar accuracies within statistical fluctuations from run to run. The tests did not provide enough difference between the two to declare one SNR better than the other, however, using all of the data available, effectively increasing the data set of the green curve by a factor, did show a visible gain in accuracy.

Using these results, I chose to train with all available data in its full weighted form and a training tolerance based on the loss function. I tested this by separating the validation data into subsets of its true target values, and plotting a histogram of the signal probability distributions. The background data completely overpowers the signal in a regular histogram, so in order to see the discrimination power of the signal, the two histograms were normalized and replotted, scaling down the background histogram and scaling up the signal histogram. Finally, a similar histogram is made with the data from CMS, consisting of 234 entries. It is uncertain which events are truly Higgs and which are non-Higgs, especially with such low statistics, but the initial behavior given from this small set shows some promise since it matches the behavior in the validation histogram.

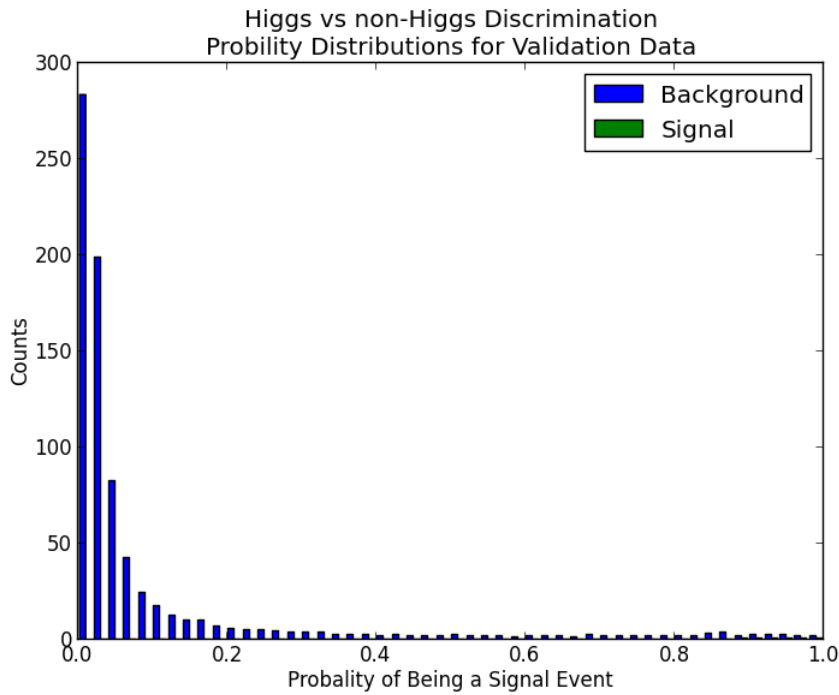


Fig. 9: Histogram of the signal probabilities for the validation data. While the trend for the background data is what is desired, it is difficult to say anything about the signal data from this plot.

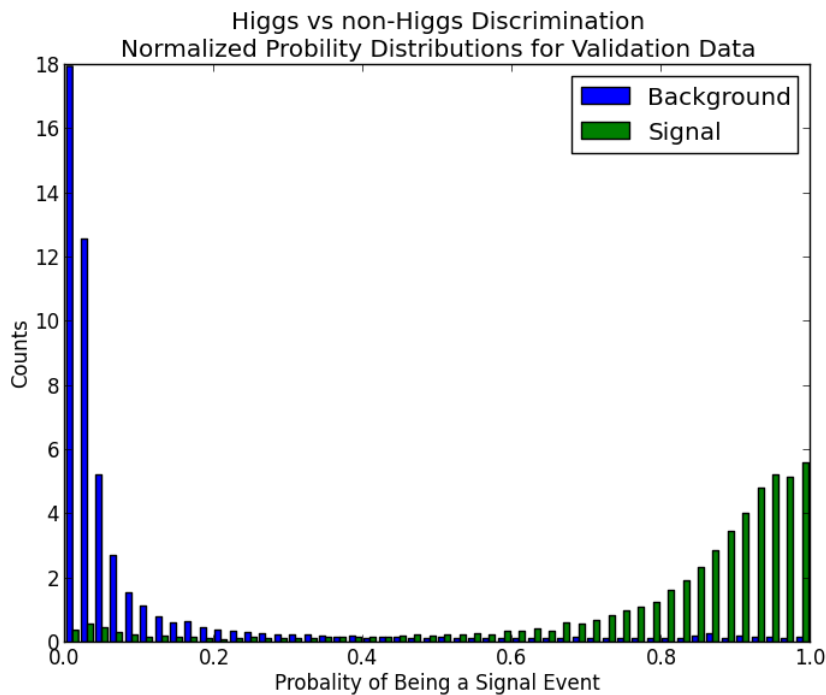


Fig. 10: Rescaling of the histogram above, to now show the discrimination power in both the signal and the background. As desired, most of the background is close to 0 signal probability, and most of the signal is close to 1. This is evidence that the ROC curves have not given false hope, and effective discrimination is actually being obtained.

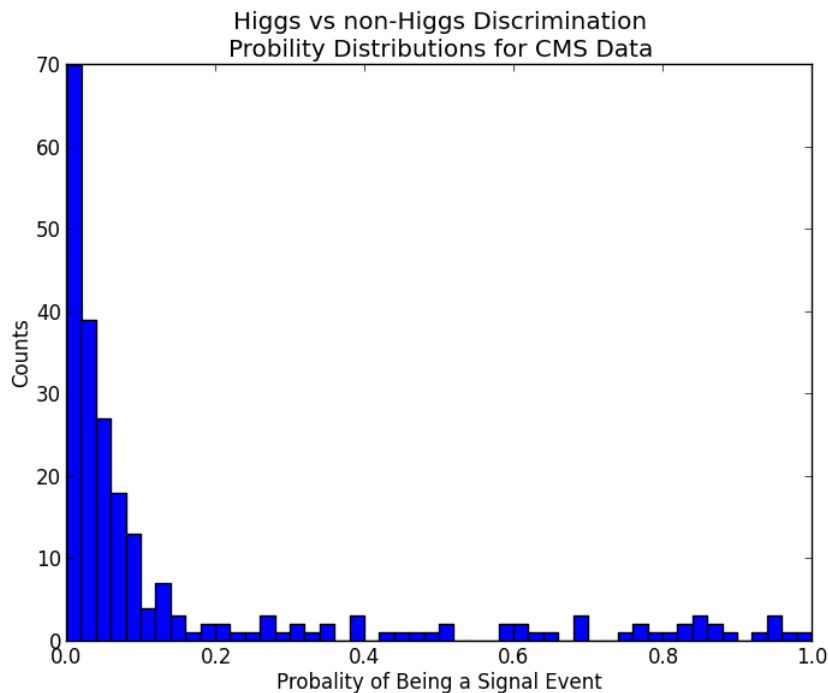


Fig. 11: Histogram of signal probabilities using 234 real events from CMS. Although statistics are too low to make a good selection of events, the initial trend from this small set is a promising indication that we are on the right track.

3.2 VBF Higgs vs. non-VBF Higgs Discrimination Tests

From the first round of tests, I decided that moving forward I would train networks with all available data regardless of SNRs, with a tolerance on the loss function instead of a test validation score, and with weighted data instead of unweighted. So for the next test, I tested which choice of raw variables is best for this discrimination. The data are set up so that when no jet is recorded, its values are all set to a very large and negative number, so I checked to see if the network would learn to discriminate events with no jets on its own. Then, I compared it to the trainings performed by requiring all events to have two jets.

There are 4 sets of input variables used: the 4-vectors (3 numbers each) for the leptons and the jets (18 inputs total), just the 4-vectors of the leptons as before (12 inputs), just the 4-vectors of the jets (6 inputs), and the historically used clever variables as a control (2 inputs, m_{jj} and δ_{jj}).

Using all the available data, regardless of how many jets were detected, the ROC curves showed that there was some discrimination being obtained, but large segments of the curve would move in a straight line at 45 degrees, indicating that for this region of the data, the network was performing no better than random guessing. Meanwhile, for the networks trained and validated once the two-jet selection rule had been applied, all ROC curves, with the exception of the network trained with just lepton 4-vectors, showed more deterministic discrimination throughout and more accurate results overall.

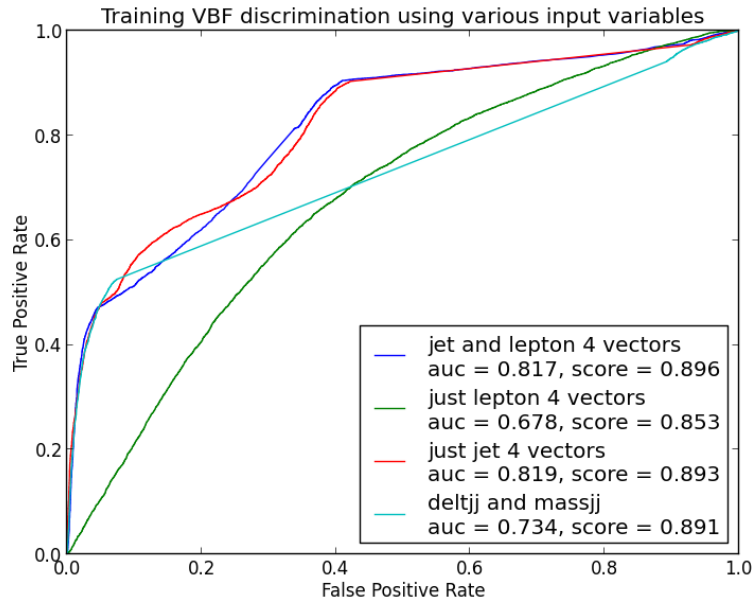


Fig. 12: Training VBF discrimination without applying two-jet selection rule. Regions along the middle of the curves appear to have very little discrimination power.

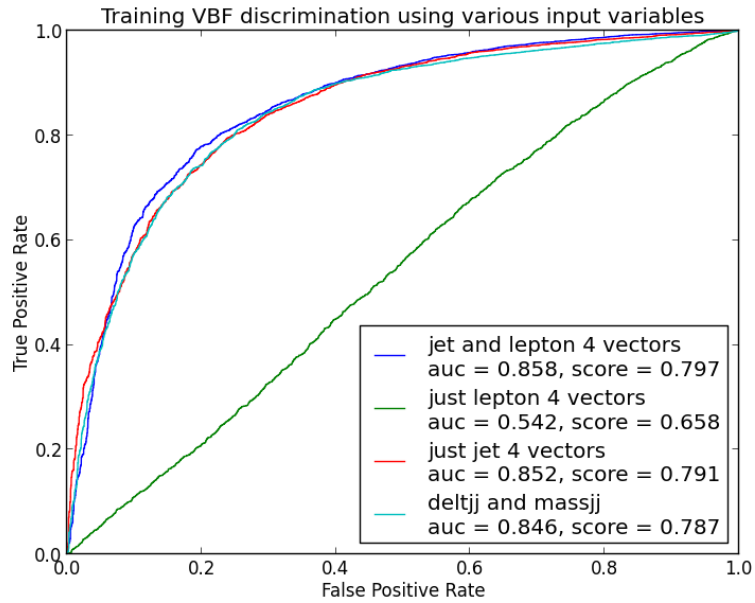


Fig. 13: Training VBF discrimination, now having applied discrimination rule. All training and validation events are required to have two jets. Training data set using all 4-vectors show best results, while the set using only the lepton 4-vectors only has a very small deviation from 0 discrimination power.

3.3 Combining Discriminations

Based on the results from the second round of tests, I decided to use the training data in the form of the 4-vectors from both the leptons and the jets, and requiring events to have two jets. Then, I checked to see how both discriminants would behave together. The first discriminant separates Higgs from non-Higgs data, and the second separates VBF events from non-VBF events. By plotting in a 2D histogram, it is possible to see how well the networks perform in separating the validation data into its various components. The first plot shows 2D histograms with Higgs signal probability on the x-axis, and VBF signal probability on the y-axis.

The first network was trained without applying the selection rule since it makes no difference for Higgs non-Higgs boson discrimination whether or not there are two jets. However, for the second network and the validation data set, the two-jet selection rule was applied. Therefore, all events plotted in the 2D histograms are those that include two jets.

The validation data was broken up into several 2D histograms to see the probability distributions for each category of event. Finally, the CMS data with the selection rule applied gave 36 data entries that were plotted to see how the real data compared to the validation data.

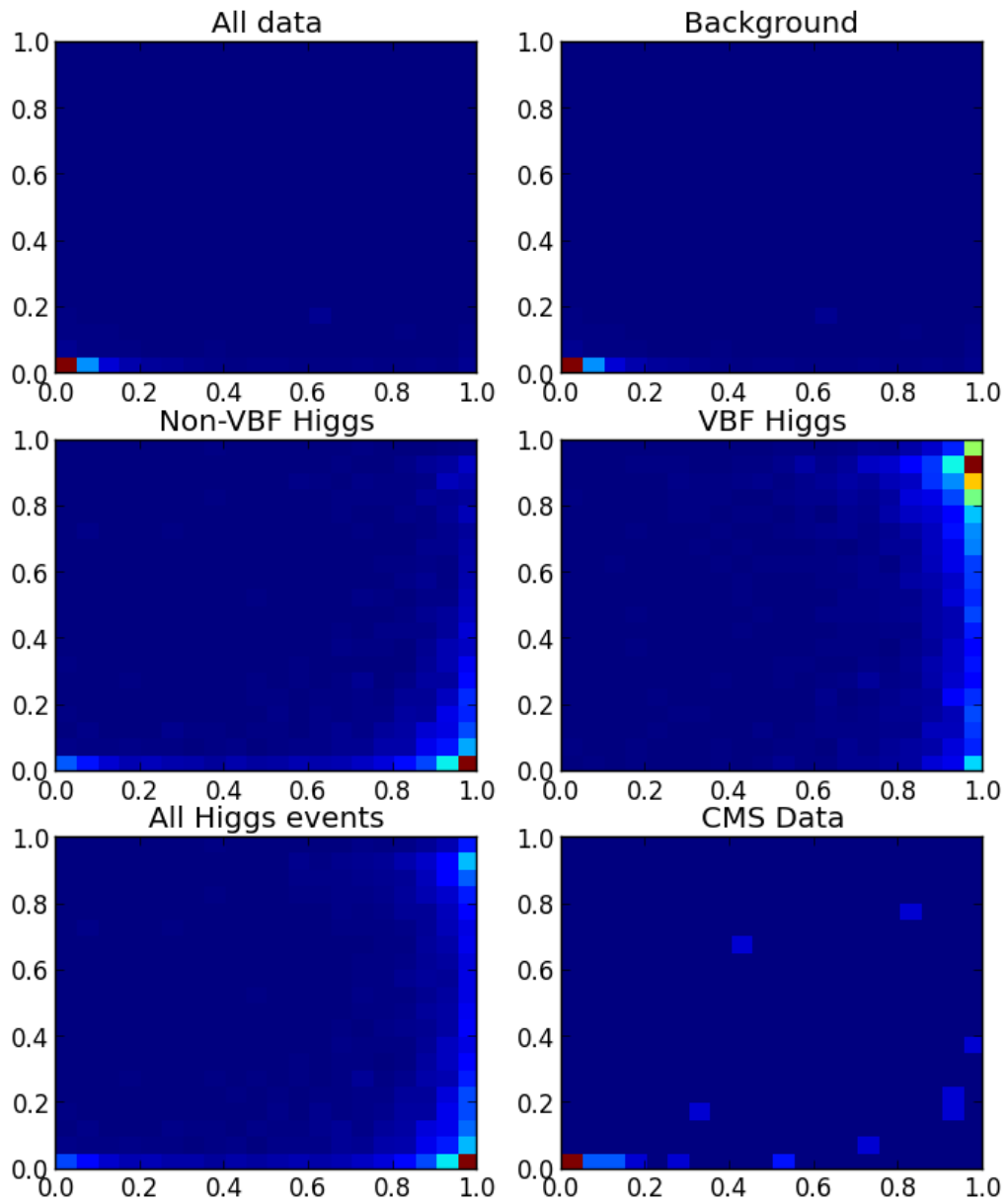


Fig. 14: 2D histograms showing the signal probability distributions for Higgs to non-Higgs on the x-axis and VBF to non-VBF on the y-axis. The data is separated into several bins based on its true values. A plot of the CMS data (with the selection rule applied) is also shown.

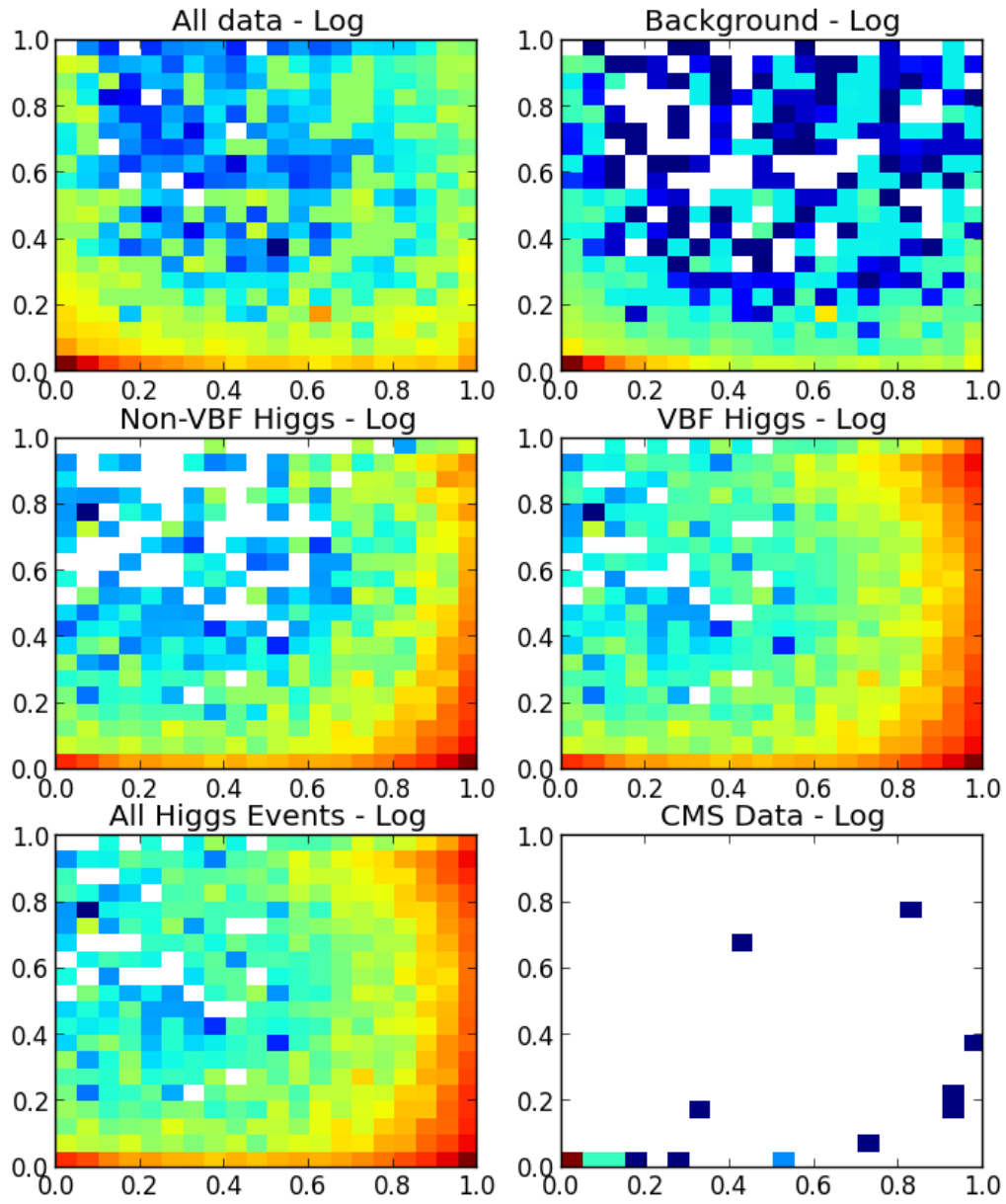


Fig. 15: Since most of the data, is focused on select regions in the histogram, it is useful to plot the histograms on a logarithmic scale to obtain a better sense of the distributions.

4 Discussion

4.1 Raw Data vs. Clever Variables

The first thing of note is that I have successfully shown that, indeed, the use of raw data to build discriminants using neural networks with multiple hidden layers is completely feasible. This was especially apparent in the VBF discrimination tests, where a network trained using 18 raw variables instead of the two clever ones produced a higher accuracy in discrimination. For Higgs vs. non-Higgs discrimination, the networks achieved a consistent ROC area of 0.96 and a validation score of 94.4%. It makes total sense that all the information that is in the clever variables historically used are also contained in the original variables from which they are derived. By using more complex machinery as I did, it is possible to extract all the necessary information without having to first transform the data in some clever way.

4.2 Signal-to-Noise Ratios

The tests regarding signal to noise ratios (SNR) showed that there is no discernible difference in using one SNR as opposed to another when it comes to training for these discriminants. This makes it possible to deal with problems in which the amount of signal is small in comparison to the background, as in VBF discrimination. Instead, it is better to use all training data available regardless of its SNR. This does not mean that having too much signal or too much background will not hinder the training. For the Higgs vs. non-Higgs event discrimination training, there were roughly 118,000 weighted signal events and 587,000 weighted background events, so although there is a larger number of background events, they are still of the same order of magnitude.

The output of the networks are probability density functions, so by Bayes' Theorem,

$$P(S|D) = \frac{P(D|S)P(S)}{P(D|S)P(S) + P(D|B)P(B)}, \quad (5)$$

where $P(S|D)$ is the output of the network, the probability of signal given a data set, $P(D|S)$ and $P(D|B)$ are the likelihoods of signal and background, respectively, and $P(S)$ and $P(B)$ are the prior probabilities of signal and background. We can define the SNR as

$$R \equiv P(S)/P(B), \quad (6)$$

and rewrite the prior equation as

$$P(S|D) = \frac{P(D|S)}{P(D|S) + P(D|B)/R}. \quad (7)$$

If we then define a discriminant of the data, d , for when $R = 1$,

$$d(D) \equiv \frac{P(D|S)}{P(D|S) + P(D|B)}, \quad (8)$$

we can rewrite the equation again, now as

$$P(S|D) = \frac{d(D)}{d(D) + (1 - d(D))/R}. \quad (9)$$

From this final expression, we can see that there is a one-to-one mapping between $P(S|D)$ and $d(D)$, and in turn, between any two network outputs, regardless of SNR. So in theory, a network can be effectively trained with any SNR.

4.3 Using Weighted Events in Training

Given the results I found with the SNR tests, it makes sense that using the weighted data to train, even if all weighted events are considered equally, is better to use than unweighting the data first. Unweighting the data to its effective count reduces the amount of data available to train. Instead, it proved more beneficial to have the network train using more data, even though ignoring the weights likely distorts the probability distributions. However, the distortion will be small if the spread of the weights is not too great.

4.4 Jet Selection Rule

One interesting failure was that unless the two-jet selection rule was applied for VBF events, the discrimination was not too strong. Events with no jets have wildly different values for the 4-vector variables, -999.0 for each of the 3 values. The network trains by recognizing differences in signal and background, and since VBF events are very often accompanied by the detection of two jets, this should have been an easy problem for the network to solve. It seems to me that what happened is that the SNR was too low and the network couldn't get a good enough resolution of the signal to effectively learn how to tell it apart from the overwhelming amount of background. Even with the two-jet selection rule applied, there was a significantly lower maximum accuracy achieved, so the addition of a large amount of noise and little to no extra signal only hindered the discriminant even further. Although Equation 9 tells us that there is no theoretical difference in training using different SNRs, a very skewed SNR is subject to practical computational limitations.

4.5 Two-Fold Discrimination

The results obtained from the 2D histograms showed promising results, with the peaks of each class of data in the region that it should be found in. For background and non-VBF Higgs events, the distributions are apparent in showing that the network is very effective in discriminating these kinds of events. Meanwhile, for VBF-Higgs events, the distribution appears more spread out, signifying that better discrimination is needed. One thing that is promising is that the majority of VBF events are still found on the right side of the histograms, meaning that they were classified correctly as Higgs events by the first discriminant.

4.6 Time Required to Run Networks

One thing that I have not mentioned before is the time the networks I used took to train. From anecdotal sources, I have heard that networks in the past used to discriminate events took on the order of a day or two to train. Due to the use of mini-batches to estimate the gradient during training, along with other methods to reduce computational effort, as well as the increase of computational power over the years, the time required to train a network has been reduced by orders of magnitude. The networks I trained took of the order of minutes, with the longest training time around 12 minutes. When I ran each training program 10 times to ensure the behavior was consistent, I wrote a simple script to do it automatically and save the graphs, and all 10 runs would finish by a little over an hour at most. The final discriminants trained to generate the 2D histograms took about 10 minutes in total to train.

5 Conclusion

For this project, I ran several tests which successfully demonstrated that the use of cleverly derived variables is not necessary to effectively train a discriminant, if one uses a complex enough network with the correct choice of inputs. For the case of Higgs to non-Higgs discrimination, my methods achieved an accuracy of 94.4% on an independent validation set and a ROC score of 0.961. Results for VBF Higgs discrimination were less optimistic, with a maximum validation score of 0.797 and a ROC score of 0.858. However, these scores were obtained using the original variables as opposed to the clever ones

historically used. The effective discrimination of VBF events still proves to be a problem, but results are promising and are a step in the right direction. This demonstrates that more complex network structures, a tool becoming more feasible with increasing computational power, can be used as a way to place more of the discrimination burden on the computer, allowing researchers to spend less time worrying about how to effectively discriminate data, and more time analyzing the data they have at hand. The code I used for this project is public and available on GitHub at <https://github.com/amsanchez96/vbf-train>.

References

- [1] Robbins, D. (2010). The Higgs Boson. Physics and Arts Summer Institute 2010
- [2] Martin, B.R. (2009). *Nuclear and Particle Physics: An Introduction, 2nd Edition*. United Kingdom: Wiley.
- [3] Reina, L. (2004). Higgs Boson Physics. Boulder CO: Tasi 2004
- [4] Flórez *et al.* (2017). Searching for new heavy neutral gauge bosons using vector boson fusion processes at the LHC. *Physics Letters B*, 767, 126–132.
- [5] Green, D. Vector Boson Fusion and Quartic Boson Couplings. US CMS Dept., Fermilab
- [6] Jain, S., Prosper, H.B. (2010). Bayesian Neural Networks.
- [7] Nielsen, M. (2017). *Neural Networks and Deep Learning*
<http://neuralnetworksanddeeplearning.com/chap1.html>
- [8] Cireřan *et al.* (2010). Deep Big Simple Neural Nets Excel on Handwritten Digit Recognition. *Neural Computation*, 22(12). <https://arxiv.org/abs/1003.0358>
- [9] <http://scikit-learn.org/stable/>
- [10] Kingma, D.P., Ba, J. (2014). Adam: A Method for Stochastic Optimization. CoRR, abs/1412.6980, <https://arxiv.org/abs/1412.6980>