Increasing the accuracy of the calculations to finding heavy Higgs Boson by using a machine learning algorithm.

Ibrahim Ahmed*
Queen Mary University of London
(Dated: April 22, 2021)

The Standard Model is the most accepted theory of the fundamental particles and how they interact with each other. However, the Standard Model has some shortcomings for example it cannot explain the matter and anti-matter imbalance. Therefore, we are searching for a model that goes beyond the Standard Model. One theory that goes beyond the standard model is the Higgs doublet Model, this model requires the existence of 5 Higgs Particles. We hope that our search for a 2nd heavier Higgs particle can validate these theories. To be able to increase the sensitivity of our results we use a machine learning algorithms trained with different data sets to better categorize active signals for particles from the background. We find that Ml algorithms are able to categorize better than classical cutoff methods.

CONTENTS

LIST OF FIGURES

* Correspondence email address: ap18020@qmul.ac.uk

I. INTRODUCTION

A. Finding the Higgs Boson

To find a particle like the Higgs Boson we need to first synthesise it. The Large Hadron Collider is used to collide 2 proton beams that are travelling with a lorentz factor of 6,930 and colliding into each other at individual energies of 6TeV giving a collision energy of 12 TeV.[3] The energy of the collision is conducive to creating a Higgs Boson particle as well as a plethora of other elementary particles. The particle that are created from the collisions are typically unstable and thus decay into lighter particles.[8][1] For us to take any useful information about the collision it must take place in a detector so that we can measure certain characteristics products of the decay. These characteristics are what allow us to determine the mother particle at the epicentre of the detector. The ATLAS Detector was used in this experiment to collect the characteristics of each particle. It is made up of multiple sub detectors constructed radially from the point of collision, such as Calorometers and trackers to collect all the data that we need. The calorimeter allows us to measure the momentum and energy of the final state particles and the tracker gives us the trajectory of the particles.[3][4] To find the Higgs Boson we are trying to look for final state particles that fit the theory of how the Higgs boson would decay, however there are a numerous amount of other particles that can be created from a proton to proton collision that are not the Higgs Boson particle.[9] We can easily spot the majority of these particles however some particle interactions can mimic the decay of a Higgs Boson Particle making the detectors detect the same final state particles, this can make seperating the Higgs Boson from these events challenging.

1. Importance of Higgs Boson

The standard model is our most complete theory in elementary particle physics and the Higgs field is an integral part of it. The Higgs mechanism requires the existence of the Higgs Boson that couples with elementary particles to give them mass in a gauge invariant way.[1] The Higgs field was theorised however no definitive proof was found until the discovery of the Higgs Boson. The Higgs Boson is an excitation in the Higgs field[1], therefore the discovery of the Higgs Boson is proof of the existence of the Higgs field. The Higgs Boson was found by utilising the ATLAS and ZMS detectors to analyse collisions. The rate of non elastic proton to proton collisions in the Atlas detector is 1 billion Hz.[3][4][9] Finding the Higgs Boson was important in solidifying the standard model as it was predicted by it. However the standard model is inherently flawed as it does not explain CP violation to the extent that we see in the universe. According to the standard model there should be equal amounts of matter and anti matter however there is no natural way of explaining the clear imbalance using just the standard model[2][8]. But, other models like the Higgs doublet model can be used to explain these phenomena with the addition of five Higgs Boson particles.[7] Of which, We are interested in finding the Heavy Higgs Boson.

2. Alternative Models

As hinted before there are models that can explain phenomena that breaks the standard model. With all models there are phenomena that are impossible to explain and some that require an addition to the model to explain them such as the neutrino flavour change phenomena.[2] Our aim is to discover the second Higgs Boson to legitimise some of the other models that require multiple Higgs particles to exist, such as the Higgs doublet model which requires the existence of 5 Higgs particles.[7] The Higgs Doublet model can explain some phenomena such as CP violation but will still leave quite a few other phenomena unexplained. A problem with the Higgs Doublet model is that there is no prediction for the mass of the heavy Higgs Boson so we have to use a series of models with different assumptions for the Heavy Higgs mass.

3. Finding the Heavy Higgs

To find a second heavier Higgs Boson we utilise a similar methodology to the original, however we have to use higher energy protons to collide to find the heavier Higgs as it requires more energy to exist. One of the main problems in our data analysis is the background processes have a larger volume compared to the volume of our signal. To mitigate these problems we have used a machine learning to be able to limit the results we use in the calculation to choose a specific area that has a higher signal to background process ratio.[9]

B. Dataset

The particles that hit the ATLAS detector are all the final state particles from the mother particle made in the collision that occurs. Of these particles there is only a few specific final state particles with a particular energy that can be the products of the second Higgs Boson particles decay. However certain decays mimic the ones we look for. So, the events that are recorded in the ATLAS detector are of all varieties.

As you can see from figure 1 there are some processes

Figure 1: background processes vs signal processes

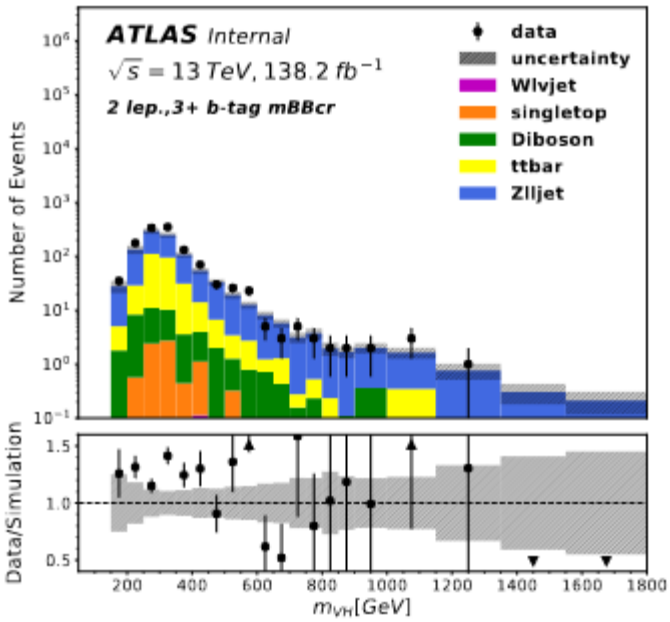


Figure 2: Feynman diagram for all Top decays

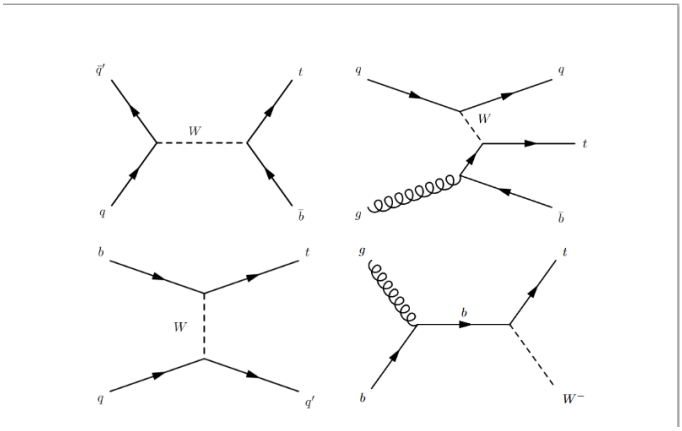


Figure 3: Feynman diagram for top anti top annihilation

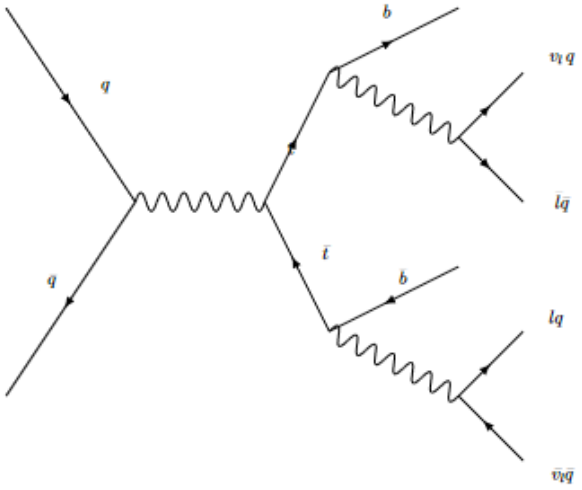
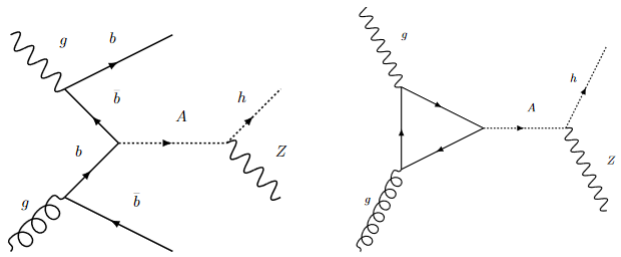


Figure 4: Feynman diagram for theorised heavier Higgs Boson decay



that have a larger contribution toward the overall background signal. This is due to the similarities between the background process and the Higgs Boson decay's final state particles.

Here are a few examples of the background processes that can occur in the inelastic proton proton collision[9]:

As you can see, in figure 3, the final state particles contain a lepton and b-jet. Both of these particles are final state particles of the Higgs decay which is the reason why this process can be mistaken for Higgs decay. This is an example of a background process that cannot be muted a lot and would need extra analysis to categorise between Higgs decay.

In figure 2 there are no leptons or b-mesons as final state particles therefore this process has less likelihood to be mistaken as Higgs decay.[1][9] This is an example of a background process that will be muted without the need of any extra algorithms.

These two processes constitute the largest and lowest contribution to our background processes. The reason why the decay of the top quark constitutes to the background more is because they cannot be muted through other means as they are too similar to the theorised decay of the Higgs. The decay of the Higgs looks like:

As seen in Figure 2,3 and 4 the top anti-top annihilation has similar final state particles, because of this we cannot use the typical methods to mute the background processes in our data set. Whereas, the single Top decay can be muted due to the difference in final state particles through pre selection of the data. I aim to utilise a machine learning algorithm to categorize [2] between the background processes that are not muted and the decay processes of our signal.

1. Data collection

In my project we collect 11 main features of the final state particles that hit the detector. All these features are reconstructed in each respective ATLAS sub detector.[9] These features are:
MVPres- This is the relativistic mass of the two leading B-meson jets that hit the atlas detectors.
METHT- The transverse energy imbalance divided by the scalar sum of the transverse momenta of the final-state particles.
MVH – This is the invariant mass of 2 leptons and 2 B-jets jets that hit the detector.
Ntags – The number of B-jets that hit the detector in an event.
MLL – The invariant mass of the 2 Leptons that hit the detector.
PTL2 – Transverse momentum of the second Lepton.
PTB1 – Transverse momentum of the first B-Jet from the collision.
PTb2 - Transverse momentum of the second B-jet from the collision.
PTV - Transverse momentum of the hypothetical mother particle reconstructed by summing the 4-momenta of the two leptons and the two b-jets.
PTH: Transverse momentum of the Higgs boson candidate, reconstructed from summing the 4-momenta of the two b-jets.
dEtabb - Difference of pseudo rapidity of the B-mesons
dEtaLL - Difference of pseudo rapidity if the Leptons
dPhiBB - Difference in angle of the 2 B-mesons
dPhiLL - Difference in angle of the 2 Leptons
The data is also separated from each other with respect to the masses of each signal point.[9]

From the Data set we are only focusing on the final state particles. This pre selection is only the leptons and b-mesons. This is to make the analysis on the categorisation easier as we can mute more background processes allowing the machine learning algorithm to focus on categorising between fewer background processes. Because of this the machine learning algorithm is specialised to only the lepton and b-meson particles.[9] All of the data in this paper is sourced from Monte-Carlo simulations.

2. Data preparation

To make sure the data is structured in a way that the neural net can understand we have to transform it in one way. We must re scale the data points between the values of -1 and 1.[2][12] This is because the framework we are using to create the neural net assumes that the data points are be between these two values. Another way we structure the data is to remove the signal mass from the data, so that the results we get from the neural net are mass independent. This will hopefully allow us to make an algorithm that does not have a mass dependence which means we can use this on more data without the need for retraining to consider different masses. We are however testing with and without the masses to see which provides us the best results. Making the Neural net mass in dependant it allows the algorithm to be applied to any model regardless of the mass of the Higgs Boson it is modelling after. However making the neural network mass dependant would make the categorisation worse for different masses as the background processes that would dominate would be different. For example the neural net output trained between the signal mass range between 300 and 2000 GeV. However in the case of keeping the mass it would provide the most accurate categorisations in the signal mass range that it was trained. This will be apparent when we see the values of significance for each of the training data sets.

C. Machine Learning

Machine learning is a tool to be able to be able to solve a problem without being programmed explicitly [2]. This allows for the program to be able to spot trends that we have not yet seen. It allows us to filter a data set without having a great knowledge of the deeper connections to what the main features are. This is important as we do not have to hard code in the complex quantum mechanical calculations to make it more apparent to the program which particles are false positives and which particles are true positives. Our use case of machine learning in this project is to use it as a categorisation algorithm.[2] This is because there are a lot of factors that goes into determining if a particle stems from our signal or if it came from background processes that occur in the detector. To classically tell these points from each other would require quantum mechanical signs that would take up a lot of unneeded resources. Alternatively, we are able to use machine learning to teach itself the trends from the training set as we have a large data set. [1][2][8]

Collecting the data points allows us to reverse engineer the collision. The problem occurs when not all the data points have to do with the decay of the large Higgs Boson. This is important as there are many background processes that like to flood the detectors with particles that we are not looking for. The first step is to ignore all the particles that are not products of the decay process we are looking for.[1] This is done by the pre selection of our data set.[9]

The next problem is to categorise whether or not the final state particles belong to a signal process or a background process. The classical way to solve this problem is to treat it like a cutoff problem. We look at a graph

of an input variable of our choosing with all the data points previously known and from the distribution of the signal and background we would choose appropriate cut off points to reduce the amount of background processes in our result. We would make cutoff's for each input variable to hone in on the amount of data points that would be most effective to use in our calculations..[9] However making 11 sets rigid cuts on a data set would greatly reduce the data points we can use and in order to calculate the mass of the Higgs Boson to a greater degree we need more data points. We need to utilise categorisation methods which are less rigid.

The ML approach utilises a machine learning algorithm to categorise signal points from background points by utilising a Neural network which uses binary cross entropy. The output of the Neural network is a discriminant which we cut on or perform a common fit of the background and the signal simulation of the data to determine the data points that we are using.

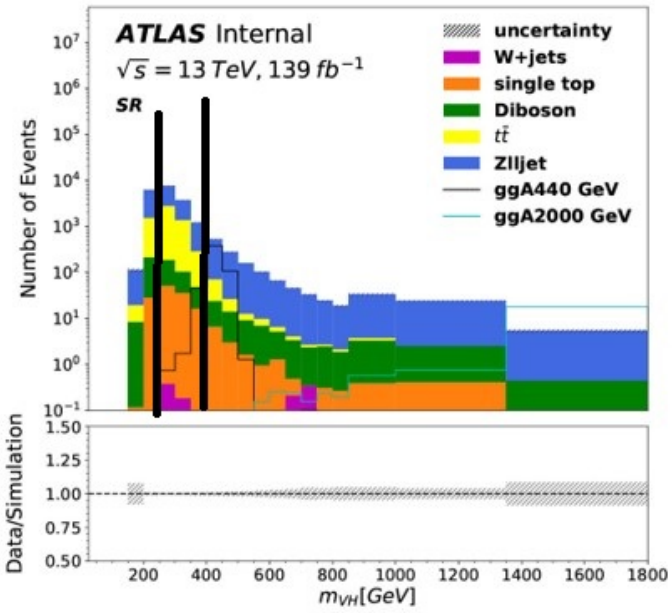
1. *Advantages of machine learning*

A machine learning algorithm allows us to extrapolate data and spot subtle trends which are not apparent to us. It does this by training on a large data set of known outcomes to develop its neural network. Moreover it allows us to tackle complex problems without having a deeper knowledge of the theoretical side. Some of the pros of making a machine learning algorithm would be that the accuracy of an algorithm will gradually get better over time as we slowly increase the data set that we collect. Another pro would be that the program can be automated with minimal aid from a human element. However in our case the former point have negligible effect as the size of the data points will not change. The main use case of machine learning in our project is the first point, extrapolating data in order to categorise data points that we do not know the true answer for. Using this feature we can increase the signal to background ration of our s data making it more efficient.[2][12]

2. *Disadvantages of machine learning*

It is very easy to over train a Neural network. This is where the algorithm becomes too specialised to its training set and cannot accurately model any other set because of it. This is detected when the loss of the testing set diverges from the loss of the training set. The key to stop this is to correctly managing your data set and make sure that you have implemented the correct amount of drop out into your neural net. A machine learning would also need a large quantity of data points to make simple decisions accurately unlike programming a traditional algorithm.[2]

Figure 5: Classical cut to minimize background in data set



3. *Machine learning in my project*

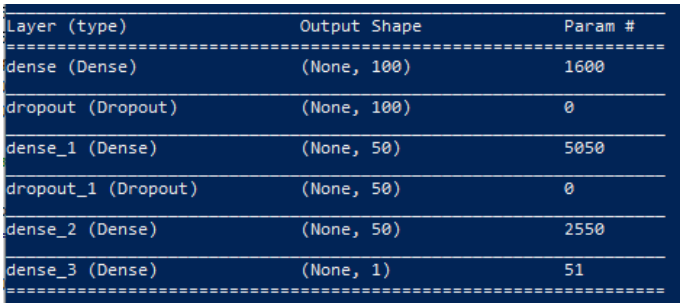
We use machine learning in in this experiment to optimize the cutoff points of the data to determine whether a point is part of a signal or background. Using a machine learning algorithm to solve this problem allows us to categorize the data points between background and signal using more than one variable as reverence, consequently making much more efficient cuts on the signal, granting the ability to use a greater percentage of the signal data points to find the heavy Higgs Boson. Thus, increasing the precision of the measurements for it. Instead of using the classical cuts which would force a lower percentage to be used. [9]

As you can see in figure 5, the classical cutoff is extremely rigid and 11 more cuts that are like this would reduce the area of data points we can use to a very small amount. By utilising a machine learning algorithm we can make one cut that will take all 11 features into account allowing us to make a more efficient cut resulting in a larger percentage of the data set to be use.

II. METHODOLOGY

What we aim to do: we want to make a machine learning algorithm that can categorize our signal from our background.
We are going to use different training data sets to find which scenario can predict the source of the data point the

Figure 6: Shape of the neural network



Layer (type)	Output Shape	Param #
dense (Dense)	(None, 100)	1600
dropout (Dropout)	(None, 100)	0
dense_1 (Dense)	(None, 50)	5050
dropout_1 (Dropout)	(None, 50)	0
dense_2 (Dense)	(None, 50)	2550
dense_3 (Dense)	(None, 1)	51

best.
The testing graphs we will use: epochs vs loss neural net output significance plots

A. Neural net Design

The first thing we set out to create was a neural net that had the correct input shape to take in 11 variables and has one output (whether or not the data point is signal or background). we have using python to create this algorithm, we will be utilising the Tensor flow API. [11][12] The machine learning algorithm we want to make will be mimicking an 11 variable function that outputs a binary number determining if a data point is either background or signal. For this reason, the shape of the neural net must be 13 - 1. 13 inputs as we take in the 11 input variables, the weights of each data point and one output, whether a data point is signal or background. Our Neural net also has to be able to cope with possible over training that can happen.[2][12] To deal with this we utilise some dropout. This burns away a few of the connection which results in the reduction of over training.[2][12][11] Other than this the Neural network is going to follow a standard design.

The activation function that my neural network will use is the Selu function. The loss function used will be binary cross entropy as the machine learning algorithm function will be used to categorizes signal points from the background.[12]

B. Dataset

1. *Tranformations*

To be able to train the data set correctly on our Neural net, we need to transform it in different ways to allow for the neural network to be able to efficiently train itself. The first transformation should be to scale all the values of the data set between -1 and 1. To do this we utilise the standard scalar function that is inbuilt in sci-kit, due to the tensor flow algorithms assumption that the data will be between the values 1 and -1.[2] The second transformation is to apply the weights for each data point. The data point weighting gives the contribution of a data point to the overall calculation. This is important as without the weights to each data point the neural network will treat each point as equal, which is not the case with our data. This is because some data from the signal will have less input in the calculations than others and to ensure that the neural network trains the data set with this in mind we scale the data with it. We also edit the normal weights for these data points when we want to normalise all the signal masses and the background. This is an effect of one of the main scenarios that we test. Essentially, we want to normalise the sums of the weights of each of the signal masses and background. The equation we use to do this is:

$$S_w = (\sum S300_w / \sum S_w) \times S_w \quad (1)$$

w = weight
$S300_w$ = Signal 300 data point weight S_w = Signal or background data point weight As you can see in Formula 1 all the signal masses are normalised to the sum of the weights for the signal mass 300. This transformation is only performed in the scenarios where we re weight the training set.

2. *Training Data*

The training data set should be a percentage of the full data set it comes from. With this is mind we take out a random 20 percent of the data set for training. We do this by using the split function in Sci-kit.[2] We take a small percentage as we do not want the algorithm to over train on the training set.
There are 4 main scenarios we want to test when we train our algorithm.
-Training with only the signal mass 300.
-Training with only the signal mass 400.
-Training with a mix of signal mass 300 and 400.
-Training with a mix of all the signal masses available to us.

Of these 4 scenarios we want to test for 2 extra features.
-Parameterise data (make the data mass dependant)
-Re-weight (normalise the weight of the data to account for differences in data set sizes for the signal masses)
This in total will account for 16 scenarios. The reason why we put extra emphasis in on the signal masses of 300 and 400 is because they are the main sources of uncertainty in the cut based analysis therefore this is the region where the ML algorithm will have greatest improvement. With these different scenarios we will find out which training set will yield the best results for our use case.

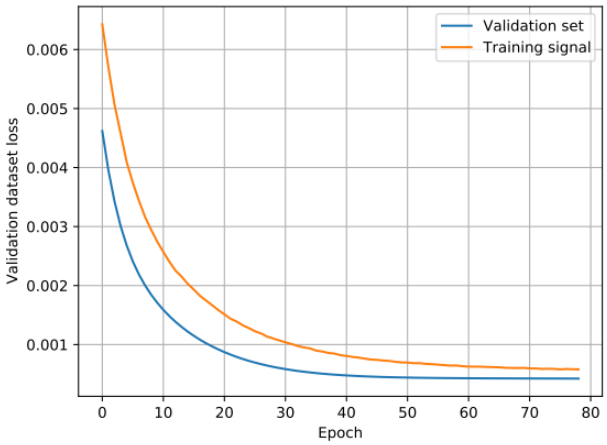
In the first case we will simply split the signal mass 300 randomly using the train test split command to allow us to take a random 20 percent of the main data set for training. We will then take the corresponding weights for each data point and utilise it in the training algorithm.[2][10]
The second case would be the same thing but with the signal mass 400 instead of the signal mass 300. The third case would be to combine both data sets. However to account for the biases of over training on a certain data set we will limit the amount of each data set that will be used in the combination. This is because if we choose 20 percent of the combined data set with no restriction there will be a higher chance that there will be more data points for the signal mass 400 than the signal mass 300. This will inherently cause the predictions for the signal mass 300 to be less accurate as we have less data points to train with. The fourth case will be a similar to the third case. This is for the same reason. In this case as the overall data set size is larger the effect is magnified for this scenario.

C. Tests

1. *Over training Test*

To test how well each of these scenarios work we want make graphs in each way a neural network can fail. The first thing want to look for is hints of over training. To be able to easily see this is to see if the loss of the neural network diverges as it trains with different epochs. This tells us if the rate at which the algorithm is learning from the training set is too high or if the training set is too specialised for a single signal mass. The possible fixes if it is over training is to reduce the learning rate of the algorithm or to increase the dropout which can both be used to be able to increase the amount of epochs we can train to without the loss diverging. The main constraint is that we will not be able to train for a greater number of epochs as the losses will eventually diverge. So for each training set we have to find the optimal amount of epochs to train for and the epochs vs loss graph is perfect for this. We will inspect the graph for any signs of divergence and conduct a binary search algorithm to find the optimal amount of epochs to train for.[2][13][12][11]

Figure 7: Loss Vs epoch



The main feature we want to look out for is if the loss of the validation set diverges away from the training loss. This is a very clear feature of over training which we are trying to stop at this point in the testing.
Figure 7 shows the ideal scenario where the loss of the training and test sets converge. This is what we expect to happen and shows us that over training is now occurring.

2. *Accuracy test*

To find the accuracy of the neural net with a given training set we have to plot the accuracy of the signal and the background. The primary output of the neural network is a discriminant. Based on this discriminant, the neural network algorithm can, if required, decide as to whether we are dealing with background or signal. The algorithm would just make a fixed cut and label everything to the left "background" and everything to the right "signal". The reason we don't want to apply such a fixed cut on the discriminant is that in the final step, when we analyze the collision data, we want to make use of all bins of the discriminant to perform a fit to the data. The left part of the discriminant would be used to control the normalization of the background simulation, and the right part would be used to search for the new heavy-Higgs signal.[9] The neural network will always sort the background to the left and the signal to the right. The ways our neural network can be inaccurate is if it has false positives or false negatives. The main problem with this would be if the two types of points are indistinguishable from each other. Then the neural network would be a failure. To be able to see this visually we will create a line graph with 2 lines:
Background - The distribution of the neural Net output for the background simulation.
Signal - This will be the predicted value for all the signal

Figure 8: neural network output

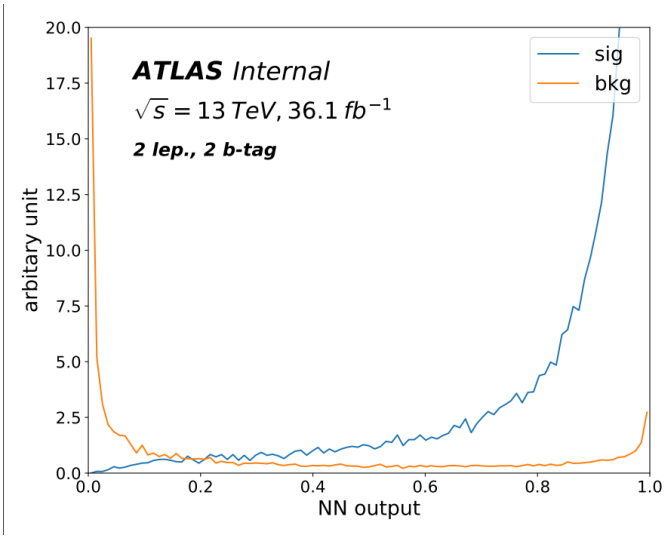
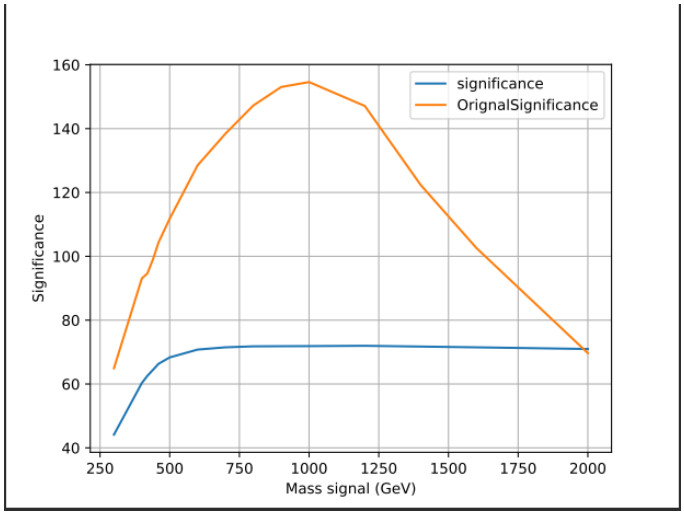


Figure 9: Significance plots



data points in the test signal.
The Y-axis will have an arbitrary scale that represents the frequency that the neural network predicted this value for different data points. The X-axis denotes the neural net output where the value 1 is the most signal like and the value 0 is the most background like. However like explained above these values can be switched around if the signal line tends towards the 0 and and the background line tends towards 1.

Figure 8 is an example of a good neural net output where the signal is clearly skewed to the right and the signal is skewed to the left. In figure 8 the main problem is that we do not have a single value to denote its accuracy. We need an identifier to calculate the overall accuracy of the neural network. To remedy this we calculate the significance from these plots to act as a numerical value for the accuracy of the neural network. Nevertheless, we can still use figure 8 to spot trends and problem areas in trouble shooting the neural network.

3. *Significance plots*

This stage is where we seriously assess the performance of the neural network output as a whole. To calculate the significance of a signal mass we use formula 1:

$$S = \sqrt{\sum_0^n 2 \times (s_n + b_n) \times \log[1 + (s_n/b_n)] - s_n} \quad (2)$$

S = Significance
s_n = Neural net signal output
b_n = Neural net background output

Formula 2 reflects the expected sensitivity of a fit performed on the neural network output on the output of the heavy Higgs Boson. The values we got from the signal is taken from applying the same model to different signal masses. This way we have a significance value for each signal mass that we can plot.[9]

Each individual significance rating has no inherent value as it does not have any units so we have no frame of reference to calibrate it. However, the true value would be with multiple points we can determine the changes in accuracy for the prediction of the signal for each different mass. The significance plots will have 2 lines if the training was done by re weighting the data and one line if it wasn't. This is because we have to use the significance for the original weight to be able to compare to the significance of the other approaches such as the classical cut so we can see if this was an improvement to the classical method. In figure 9 the X-axis is the signal masses of our data and the y axis is the an arbitrary scale for the significance. Where the higher it is, the higher the accuracy of the neural net output is. The line labelled significance is a good indicator for optimising the neural net to give better outputs, But, due to the normalisation of the weights it is not an effective tool to compare against different cutoff methods, so we need to use the original weights to give a fair comparison. For this reason, the original significance is also plotted so we can compare the results with other methods. The significance reading does give us an effective way of comparing neural network models for all our scenarios.

D. Code

The basic skeleton of the testing is going to be saved inside of a single function. The main CSV data unpacking will be done globally.

1. Testing function

The first two steps in my testing function is to re weight the data if the variation of my testing wants it and make the data mass dependant if the variations of my testing allows it. To do this I utilise 2 if statements. One if statement deals with including the mass feature. For the background it chooses a random signal mass for every data point. For the re weighting we used formula one to change each identical weight after saving the originals weights in its own data frame. All of these changes and additions to the features is done through the use of Pandas data frames.[10]
The next step is to construct the training data, concatenate all the signal data with the background, Set all the signal points to 1 and background to 0 and scale all the values between the values of 1 and -1. To do this we use the concatenate function to fuse multiple data frames together and shuffle all that data points together.[10] Next, we separate the weights from the features that will be used to train the model. Now that we have separated the weights from the features for all the data frames we split our training data frame into using the train test split function from sci kit. We do this to get the testing set which we use to check if the model is over training.
I then create the neural network and train it with 1000 epochs and forcing the training to stop when it sees no change in loss after 3 epochs. When creating the model I use the Adam optimiser function, the Selu activation and use binary cross entropy as the loss function.[12] Now this is where I construct the various graphs I will be using to compare my method with others. Just like planned I make an epoch vs loss graph, neural net output graph and significance plots. To construct the plots for epoch vs loss and significance vs mass signal I utilise matplotlib to create we the scale and make the entire graph.[13] To get the lists of values for the epochs vs loss graph the model.fit function in Keras outputs the validation set loss and training set loss. For the significance's I utilise a pre defined function called significance binned which was courtesy of Ton Qiu. For the neural net output graphs I will make use of a function that was pre programmed by Ton Qiu the PhD student I work with.[9] This function is called curve plot. All of this will be inside one testing function and this will be called for each scenario I want to test.

2. Test code

First I will need to import all of the modules I will need to run my function described above:[10][112][13][9]
-Pandas
-Sci-kit
-NumPy
-Tensorflow
-Matplotlib
-Curveplot
Then I unpack all the CSV files so that I can utilise the Data Frames constructed in the different testing scenarios. Then I test my code for all 16 scenarios that was described:
-Parameterised and re-weighted data trained on a mix of all signal masses.
-Re-weighted data trained on a mix of all signal masses.
-Parameterised data trained on a mix of all signal masses.
-Untouched data trained on a mix of all signal masses.
-Parameterised and re-weighted data trained on a mix of 300 and 400 signal masses.
-Re-weighted data trained on a mix of 300 and 400 signal masses.
-Parameterised data trained on a mix of 300 and 400 signal masses.
-Untouched data trained on a mix of 300 and 400 signal masses.
-Parameterised and re-weighted data trained on 400 signal mass.
-Re-weighted data trained on 400 signal mass.
-Parameterised data trained on 400 signal mass.
-Untouched data trained on 400 signal mass.
-Parameterised and re-weighted data trained on 300 signal mass.
-Re-weighted data trained on 300 signal mass.
-Parameterised data trained on 300 signal mass.
-Untouched data trained on 300 signal mass.
The code for all of these sections can be found in the appendix.

III. RESULTS AND DISCUSSIONS

A. Background and signal distribution

The first graph I want to talk about is a plot of the background at various signal masses. This sis so we can get an idea of the main regions where the background to signal ratio is at its highest.

Figure 10: MVH signal stack plot

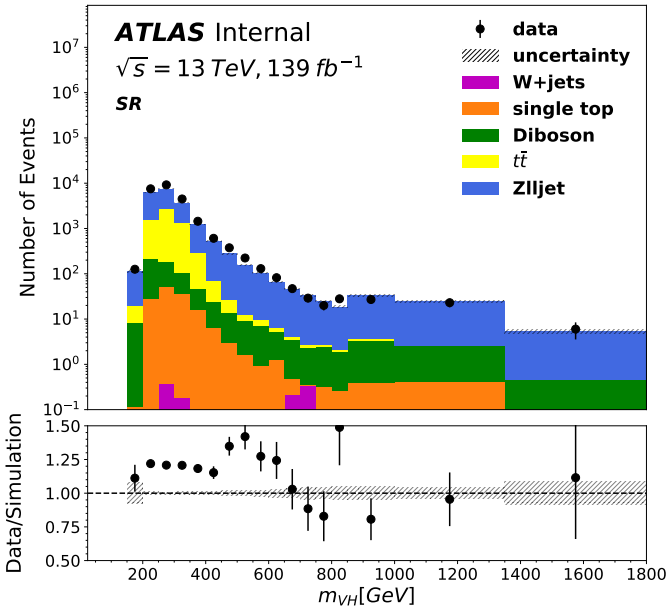
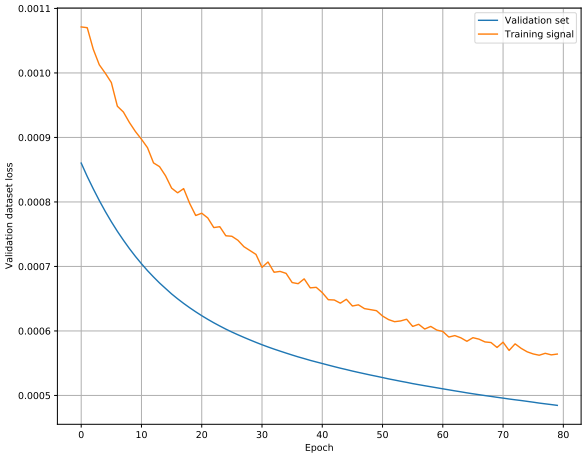


Figure 11: Epoch vs loss graph



As you can see in Figure 10 the main problem area as expected was the low mass region as the background to signal ratio is the highest there. As we can see from the graph the two main background processes that cause trouble are the top anti top annihilation as discussed in the introduction and the Zll jet. With the latter having a greater contribution to the background to signal ratio, We can also see how the overall ratio drops significantly for signal masses past 400 GeV and that the top anti top annihilation process disappears by signal mass 1000. From this we can infer that any type of analysis will have a lower accuracy in the lower mass regions and have an easier time with the higher mass regions. Also due to the nature of the data set if a ML algorithm trained with the lower mass regions it would be more equipped to deal with a larger variety of background processes as there are more background processes to filter out. This could result in better performances in masses that have not been tested in this project.

B. Over training

In the next graph we want to check if at any point the model we are training starts to over train. To be able to see this easily we will plot the losses over epochs:

In Figure 11 we clearly see that the loss of both training and validation set sat rt to converge but is ended before they get the chance to. This is because of the way the training is coded. I have made it so that if the model sees no improvement after 3 epochs then the model will halt training making it impossible for the algorithm to over train. This is because the model will simply stop training if the the losses start to diverge. Utilising this method we can assure that there is no over training in any of the models we will test.

C. Neural net output

In the Neural network output graph we will show graphically how the Ml algorithm has predicted the split between signal and background is. For this I want to show 3 varieties of these graph. I want to show the behaviour between low signal masses, medium signal masses and high signal masses:

Figure 12: ML output for 2000 GeV signal mass

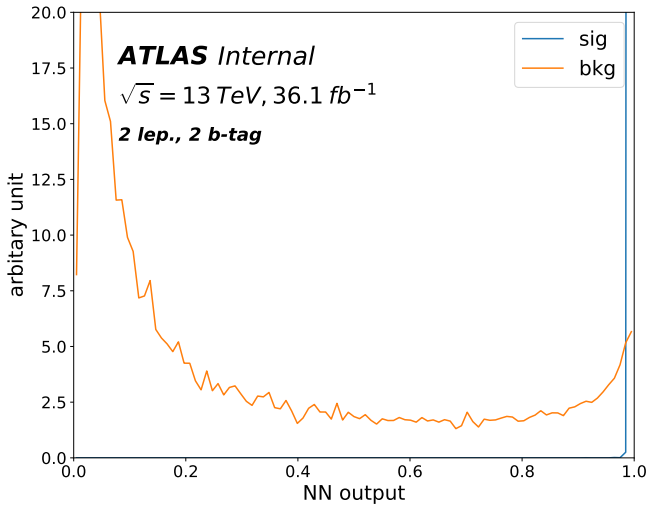


Figure 12 shows how the distribution predicted by the Ml algorithm shows a large spike in the final bin for the signal. This affects the significance readings as the true nuances of the shape is not shown as the resolution is not high enough even with 100 bins used in total. What this means is that often the accuracy of the ML algorithm calculated from these values will plateau as soon as we get the majority of signal points in the final bin. This will seriously affect our ability to analyse the performance of this approach in the higher mass regions

Figure 13: ML output for 400 GeV signal mass

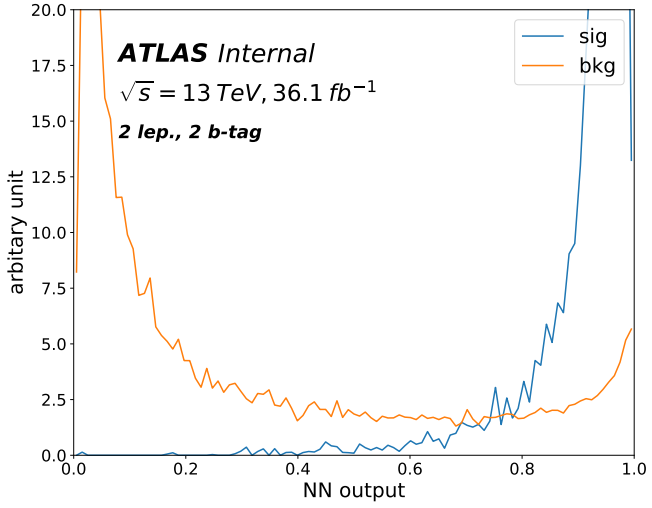
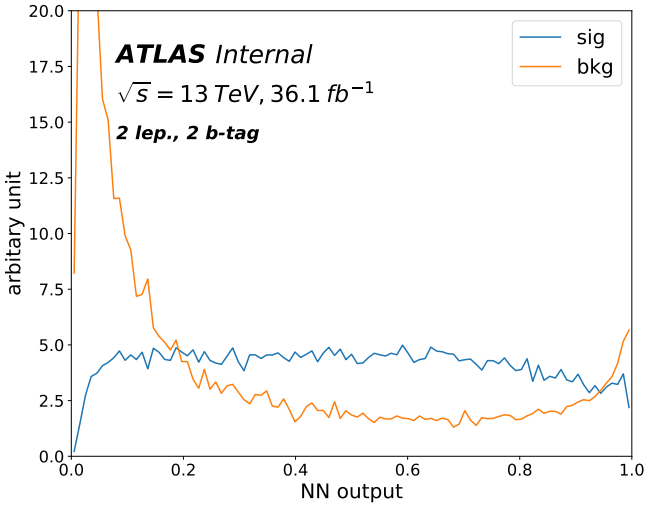


Figure 13 shows how the increased resolution lets us see exactly where the signal peak is. However we can see that the effects that occur in the higher mass regions start to show in the signal region of 400 and as you will see in the significance plots come into full effect at around the 1000 GeV signal mass region. You could argue that the ML algorithm simply gets to the maximum accuracy at a lower mass region showing that it is very effective in categorising signal from background.

Figure 14: ML output for 300 GeV signal mass

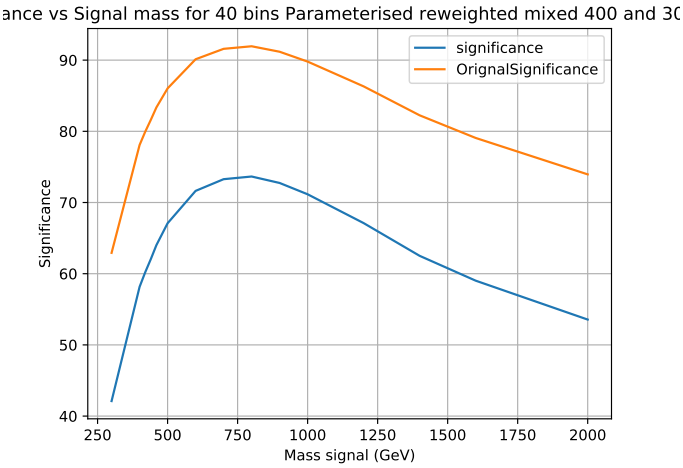


As seen in Figure 14 we see that there is a vast difference in the signal prediction for the lower mass signal for the ML algorithm. The values tend to be more shifted toward the left side predicting a lot of false negatives. This may be due to the fact of some background processes that are apparent in low mass signal not being trained for by the training set. For our neural network outputs the accuracy is calculated by calculated the scale factors to scale the background ans signal distributions to match the true distributions which are known prior. This will give a final figure of merit to use which is significance.newline

D. Significance plots

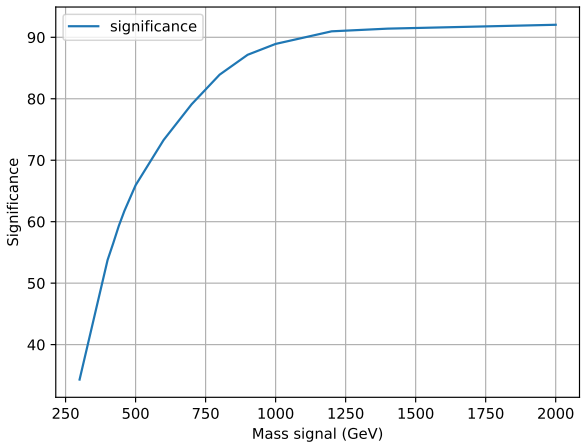
The significance plots is where we seriously evaluate all the models of our neural network. The are 3 main points that I want to out line:
-The effect of parameterisation and reweighting on the significance
-The best significance outputs for all training sets
-A comparison to the significance of the classical cut based method.

Figure 15: significance when the training sets weights are re-weighted



In figure 15 we can see that the significance drops off at the higher mass signals as the weights have been normalised. I believe that the normalisation reduces the effect the higher masses have on the training of the model. This in turn would reduce the accuracy of the higher mass signals as the data set have not been modified.

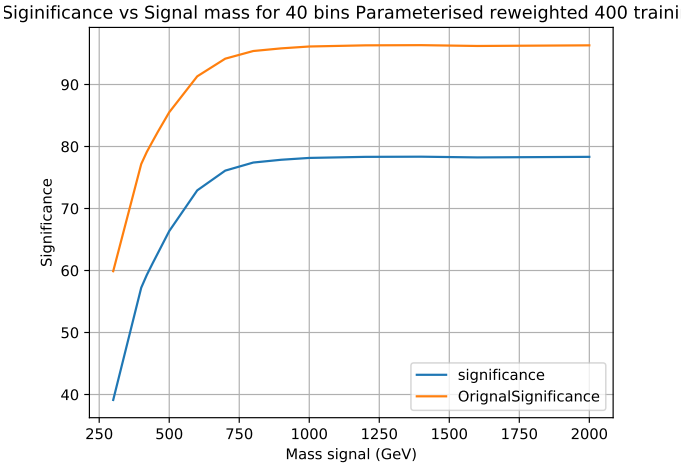
Figure 16: significance when the training sets weights are not re-weighted



In figure 16 we see that the model does not peak in the lower mass signals and the significance constantly grows. However the significance and the rate that the significance increases in the lower mass region is a lot greater in figure 15.

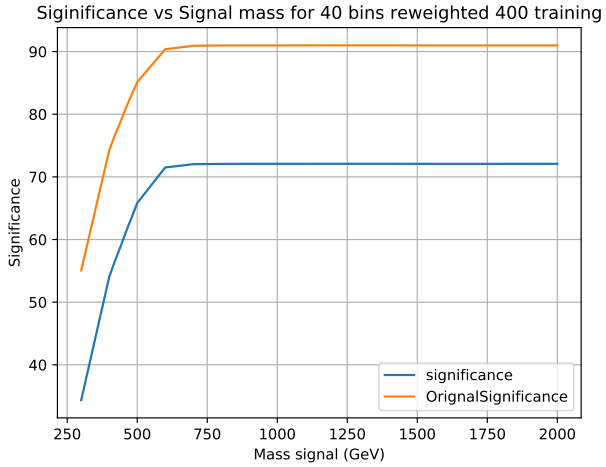
As you can see reweighteing the data set seems to improve the significance ratings of the lower mass signals but at the cost of the significance of the higher mass signals. As you will see the best version off each of the models used will have re-weighted data. This is because the gains of significance outweighs the losses of significance for the higher mass signals.

Figure 17: significance when the training sets weights are parameterised



In Figure 17 we see that the parameterisation does not reduce the significance of the higher mass signal. it increases the overall significance over all the signal masses.

Figure 18: significance when the training sets weights are not parameterised



In figure 18 we see that the significance increases dramatically until the signal mass 600 but then plateaus very

hard and the increase in significance becomes very small compared to what it was

As we can see from the above examples the parameterisation increases the overall significance however the machine learning algorithm will become too specialised to the 300 - 2000 GeV signal mass region. This is a a problem as we do not yet know the theoretical weight of the heavier Higgs Boson particle therefore we nee to make an algorithm that can be used in a large variety of signal masses and parameterisation makes the algorithm mass dependant.

Figure 19: Best significance for 400 GeV signal mass training set

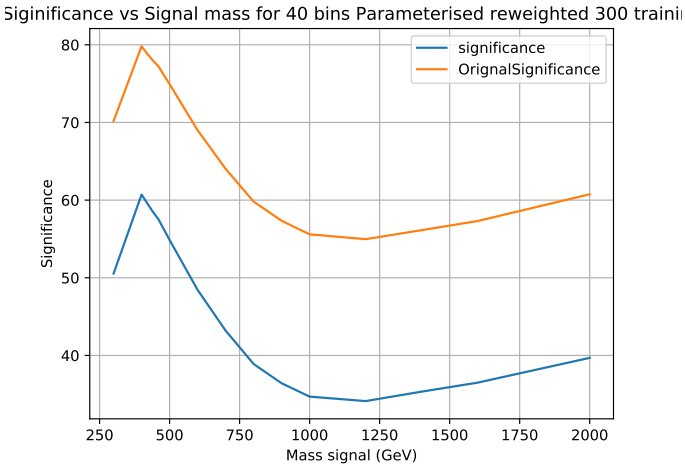


Figure 20: Best significance for 300 GeV signal mass training set

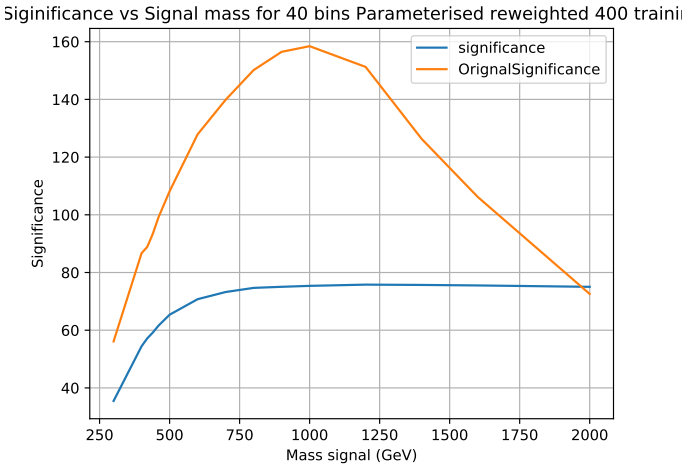


Figure 21: Best significance for 400 and 300 GeV signal mass training set

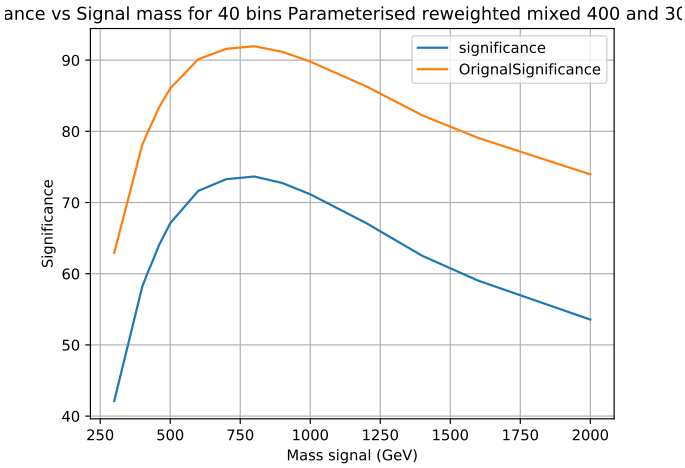
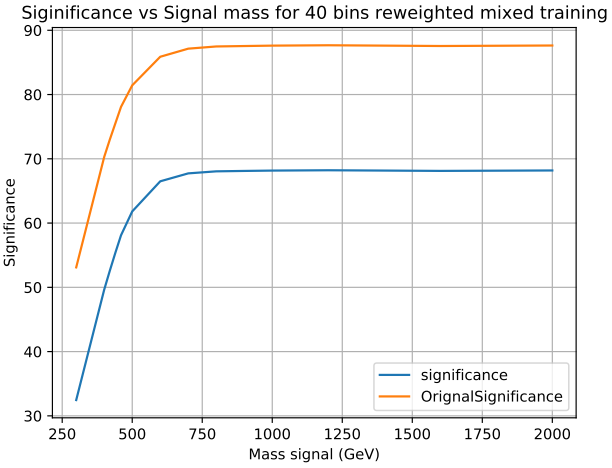
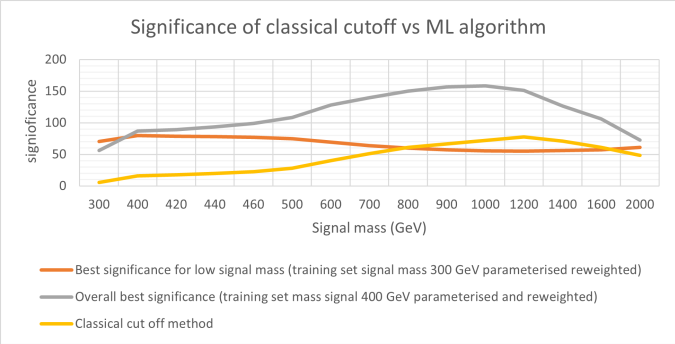


Figure 22: Best significance for mixed signal mass training set



As you can see every each of the best models has reweighted the data set. As eluded to before this is because the increase in significance in the lower mass regions is very high compared to the loss in significance in the higher mass regions. The also parameterise the data which is not an ideal way to train the model when being used on different signal mass ranges but provides the highest significance ratings in this specific case.

Figure 23: Best significance for mixed signal mass training set



In Figure 23 we see that for signal masses below 700 GeV the significance is always greater than the classical cutoff. For the machine learning algorithm the training set 300 model predicts the signal masses of 800 GeV and above worse showing. the difference in significance when the Ml algorithm trained with 300 GeV dips below the classical cut is smaller than when the Ml algorithm has a better significance which means that in the specified mass range it is much better than the classical cut method, We see that for both ML algorithms the significance stops off for the higher signal masses however the classical cut off method increases in this range. This is most likely due to the normalisation of the weights.As you can see the Overall best significance is consistently higher than the classical cut off method and is only beaten by the low mass ML algorithm in the low mass region.

E. Conclusion

In conclusion we can clearly see that the machine learning algorithm is better than the classical cut off method in this categorisation problem. This is shown in Figure 23. We also see that the best training set that will give us the highest rating overall is with the training set of 400 and the training set that gives the best predictions for the lower mass signals is with signal mass 300. However the difference in significance between the training set of 300 and the training set of 400 in the mid signal mass region is extremely high and could be a reason for the ML model trained on signal mass of 300 GeV is not used even for the lower signal masses. We also find that reweighting the data set reduces the significance for the larger signal masses but increases the significance for the lower signal masses by a greater amount. Parameterising also increases the significance however we are not able to test for the reduction in the models usefulness in particle interactions that are outside the range of energy our monte-carlo simulations have given us. So we cannot measure how much mass dependence is a detriment to the models prediction in a random data set.

ACKNOWLEDGEMENTS

[1] Matthew D. Schwartz,Quantum Field Theory and theStandard Model, (Cambridge University Press, NewYork, 2014).

[2] Aurelien Geron,Hands-On Machine Learning withScikit-Learn, Keras and TensorFlow, (O'Reilly mediaInc, Canada, 2019).

[3] CMS collaboration,The CMS experiment at the CERNLHC, (Institute of Physics Publishing, J. Instrum. 3,S08004 (2008)).

[4] Lyndon Evans and Philip Bryant,LHC Machine, (TheInstitute of Physics publishing 2008 JINST3 S08001).[5] Frank W. K. Firk,Introduction to Relativistic Colli-sions, (The Henry Koerner Center for Emeritus Faculty,Yale University, New Haven CT 06520).

[6] W.-D. Schlatte,Observation of a new particle in thesearch for the Standard Model Higgs bosonwith the AT-LAS detector at the LHC, (Physics Letters, Section B:Nuclear, Elementary Particle and High-Energy Physics(2012)).

[7] J.A. Bagger,Theory and phenomenology of two-Higgs-doublet models, (Physics Reports Volume 516, Issues1–2, July 2012, Pages 1-102).

[8] M. Tanabashi et al. (Particle Data Group),Review ofParticle Physics, (Regents of the University of California, Phys. Rev. D 98, 030001 (2018)).

[9] Tong Qiu,Search for a heavy pseudo scalar Higgs bosondecaying into a Z boson and a light Higgs boson with theATLAS detector, (Queen Mary University of London,First year PhD progression report, May 2019).

[10] pandas-dev,Pandas 0.25 documentation, (Jul 18, 2019Version: 0.25.0).

[11] Pedregosa, F. and Varoquaux, G. et al. , Scikit-learn: Machine Learning in Python, (Journal ofMachine Learning Research, 2011).

[12] Martín Abadi, Ashish Agarwal, Paul Barham, et al.,Tensor-Flow: Large-Scale Machine Learning on HeterogeneousSystems, (2015. Software available from tensorflow.org).

[13] Hunter, J. D,Matplotlib: A 2D graphics environment,(IEEE COMPUTER SOC, 2007).

[14] Irwan Bello,Neural Optimizer Search with Reinforcement Learning, (arXiv:1709.07417 2017).[15] Lutz Prechelt,Early Stopping - But When?, (OrrG.B., Müller KR. (eds) Neural Networks: Tricks of theTrade. 2002).[16] Callaway, D. J. E. ,Triviality Pursuit: Can Elementary Scalar Particles Exist?, (Physics Reports. 167 (5):241–320. (1988)).

IV. APPENDIX

Figure 24: Mass separation of our data

```
#This part adds masses to the data set making it parameterised
if masssep == 1:
    test_signal_300["mass"] = 300
    test_signal_400["mass"] = 400
    test_signal_420["mass"] = 420
    test_signal_440["mass"] = 440
    test_signal_460["mass"] = 460
    test_signal_500["mass"] = 500
    test_signal_600["mass"] = 600
    test_signal_700["mass"] = 700
    test_signal_800["mass"] = 800
    test_signal_900["mass"] = 900
    test_signal_1000["mass"] = 1000
    test_signal_1200["mass"] = 1200
    test_signal_1400["mass"] = 1400
    test_signal_1600["mass"] = 1600
    test_signal_2000["mass"] = 2000
    #the background will have a random mass assigned to each data point
    masslist = []
    for i in range(len(background_all)):
        masslist.append(signal_mass[random.randint(0,14)])
    background_all.insert(17,"mass",masslist)
```

Figure 25: Normalisation of weights

```
#this will scale the weights to make it so that each signal mass is equivalent regardless of the dataset size
if weightscaling == 1:
    #this saves the orignal weights which we will use for the significance plots at the end
    orignal_test_signal_300 = pd.concat([test_signal_300["weight"]],ignore_index=True)
    orignal_test_signal_400 = pd.concat([test_signal_400["weight"]],ignore_index=True)
    orignal_test_signal_420 = pd.concat([test_signal_420["weight"]],ignore_index=True)
    orignal_test_signal_440 = pd.concat([test_signal_440["weight"]],ignore_index=True)
    orignal_test_signal_460 = pd.concat([test_signal_460["weight"]],ignore_index=True)
    orignal_test_signal_500 = pd.concat([test_signal_500["weight"]],ignore_index=True)
    orignal_test_signal_600 = pd.concat([test_signal_600["weight"]],ignore_index=True)
    orignal_test_signal_700 = pd.concat([test_signal_700["weight"]],ignore_index=True)
    orignal_test_signal_800 = pd.concat([test_signal_800["weight"]],ignore_index=True)
    orignal_test_signal_900 = pd.concat([test_signal_900["weight"]],ignore_index=True)
    orignal_test_signal_1000 = pd.concat([test_signal_1000["weight"]],ignore_index=True)
    orignal_test_signal_1200 = pd.concat([test_signal_1200["weight"]],ignore_index=True)
    orignal_test_signal_1400 = pd.concat([test_signal_1400["weight"]],ignore_index=True)
    orignal_test_signal_1600 = pd.concat([test_signal_1600["weight"]],ignore_index=True)
    orignal_test_signal_2000 = pd.concat([test_signal_2000["weight"]],ignore_index=True)
    #This is where we rewight the data
    test_signal_300["weight"] = (test_signal_300["weight"].sum()/test_signal_300["weight"].sum())*test_signal_300["weight"]
    test_signal_400["weight"] = (test_signal_300["weight"].sum()/test_signal_400["weight"].sum())*test_signal_400["weight"]
    test_signal_420["weight"] = (test_signal_300["weight"].sum()/test_signal_420["weight"].sum())*test_signal_420["weight"]
    test_signal_440["weight"] = (test_signal_300["weight"].sum()/test_signal_440["weight"].sum())*test_signal_440["weight"]
    test_signal_460["weight"] = (test_signal_300["weight"].sum()/test_signal_460["weight"].sum())*test_signal_460["weight"]
    test_signal_500["weight"] = (test_signal_300["weight"].sum()/test_signal_500["weight"].sum())*test_signal_500["weight"]
    test_signal_600["weight"] = (test_signal_300["weight"].sum()/test_signal_600["weight"].sum())*test_signal_600["weight"]
    test_signal_700["weight"] = (test_signal_300["weight"].sum()/test_signal_700["weight"].sum())*test_signal_700["weight"]
    test_signal_800["weight"] = (test_signal_300["weight"].sum()/test_signal_800["weight"].sum())*test_signal_800["weight"]
    test_signal_900["weight"] = (test_signal_300["weight"].sum()/test_signal_900["weight"].sum())*test_signal_900["weight"]
    test_signal_1000["weight"] = (test_signal_300["weight"].sum()/test_signal_1000["weight"].sum())*test_signal_1000["weight"]
    test_signal_1200["weight"] = (test_signal_300["weight"].sum()/test_signal_1200["weight"].sum())*test_signal_1200["weight"]
    test_signal_1400["weight"] = (test_signal_300["weight"].sum()/test_signal_1400["weight"].sum())*test_signal_1400["weight"]
    test_signal_1600["weight"] = (test_signal_300["weight"].sum()/test_signal_1600["weight"].sum())*test_signal_1600["weight"]
    test_signal_2000["weight"] = (test_signal_300["weight"].sum()/test_signal_2000["weight"].sum())*test_signal_2000["weight"]
    background_all["weight"] = (test_signal_300["weight"].sum()/background_all["weight"].sum())*background_all["weight"]
```

Figure 26: Data transformations

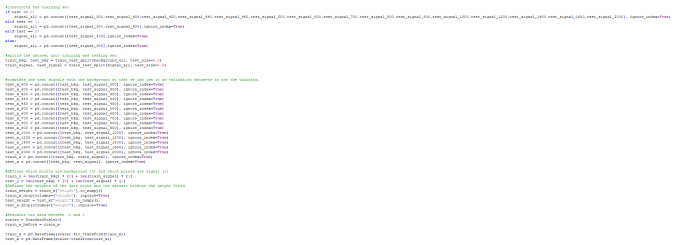


Figure 27: Model creation and training

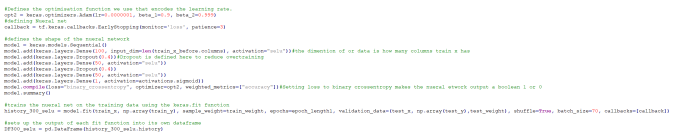


Figure 28: Plotting of epochs vs loss graph

```
#plots each data frame as a line into the epochs vs loss graph
plt.plot(DF300_selu[["val_loss"]],label = "Validation set")
plt.plot(DF300_selu[["loss"]],label = "Training signal")
plt.legend()
plt.grid(True)
plt.xlabel("Epoch")
plt.ylabel("Validation dataset loss")
plt.gca()
plt.savefig('epochVSloss ' + fileaddition +".pdf")
```

Figure 29: Plotting of neural net output graph

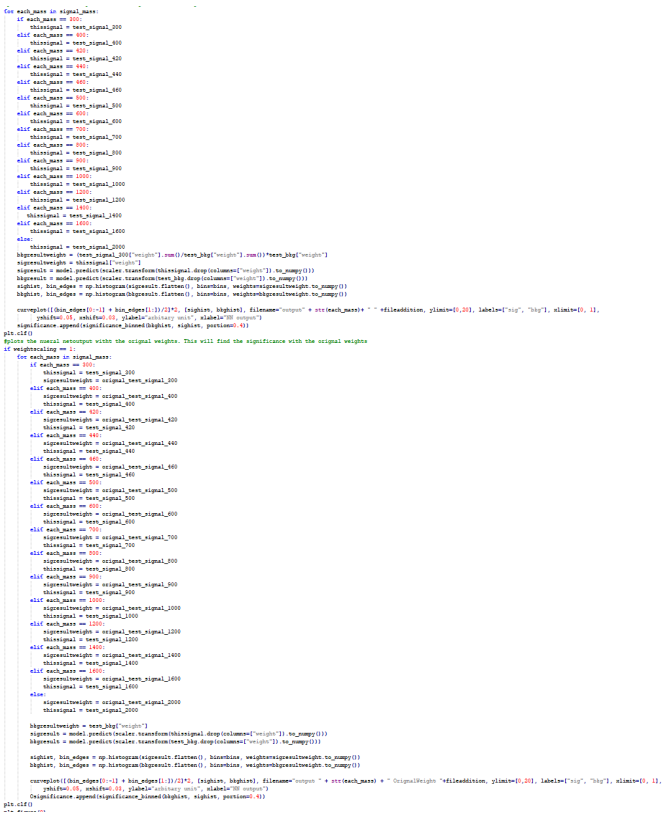


Figure 30: Plotting of significance plots

```
#plots the significance against signal mass
plt.plot(signal_mass,significance,label = "significance")
#plots the significance with the orignal weights against the signal mass
if weightscaling == 1:
    plt.plot(signal_mass,Osignificance,label = "OrignalSignificance")
plt.legend()
plt.grid(True)
plt.xlabel("Mass signal (GeV)")
plt.ylabel("Significance")
plt.gca()
plt.savefig('Significance plots '+fileaddition+".pdf")
plt.close()
```

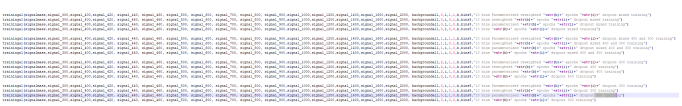
Figure 31: Module importing

```
#Imports all the packages we need to train a machine learning algorithm
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, MinMaxScaler
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
from tensorflow import keras
from tensorflow.keras import activations
import matplotlib.pyplot as plt
import time
import os
import sys
import random
lib_path = os.path.abspath(os.path.join(__file__, '..', '..'))
sys.path.append(lib_path)
sys.path.append('../package')
from package.curveplot import curveplot
import math
```

Figure 32: Data set unpacking

```
#Opens the csv files and unpacks them into respective dataframes
for each in signalmass:
    df_temp = pd.read_csv(str(each) + ".csv", index_col=0)
    df_temp.drop(columns=["nTags", "MCChannelNumber"], inplace=True)
    if each == 300:
        signal_300 = pd.concat([df_temp], ignore_index=True)
    elif each == 400:
        signal_400 = pd.concat([df_temp], ignore_index=True)
    elif each == 420:
        signal_420 = pd.concat([df_temp], ignore_index=True)
    elif each == 440:
        signal_440 = pd.concat([df_temp], ignore_index=True)
    elif each == 460:
        signal_460 = pd.concat([df_temp], ignore_index=True)
    elif each == 500:
        signal_500 = pd.concat([df_temp], ignore_index=True)
    elif each == 600:
        signal_600 = pd.concat([df_temp], ignore_index=True)
    elif each == 700:
        signal_700 = pd.concat([df_temp], ignore_index=True)
    elif each == 800:
        signal_800 = pd.concat([df_temp], ignore_index=True)
    elif each == 900:
        signal_900 = pd.concat([df_temp], ignore_index=True)
    elif each == 1000:
        signal_1000 = pd.concat([df_temp], ignore_index=True)
    elif each == 1200:
        signal_1200 = pd.concat([df_temp], ignore_index=True)
    elif each == 1400:
        signal_1400 = pd.concat([df_temp], ignore_index=True)
    elif each == 1600:
        signal_1600 = pd.concat([df_temp], ignore_index=True)
    else:
        signal_2000 = pd.concat([df_temp], ignore_index=True)
backgroundall = pd.read_csv("background.csv", index_col=0)
backgroundall.drop(columns=["nTags", "MCChannelNumber"], inplace=True)
#defines the parameters fir the function below
bins5 = np.linspace(0,1,100)
dropoutlist = [0.2,0.4]
i=0.4
b=1000
```

Figure 33: Final testing



Git lab link to all plots made: https://git.ph.qmul.ac.uk/ap18020/project-graphs