Training Neural Networks for Likelihood Ratio Estimation for the Higgs Boson Machine Learning Challenge with Various Strategies to Account for Missing Variables

Martin Føll

Department of Physics,
University of Oslo, N-0316 Oslo,
Norway

In this project we do the Higgs boson machine learning challenge (Adam-Bourdarios *et al.*, 2015) to discover the Higgs boson using a neural network, and to calculate the discover significance with the method of a search region and from the profil likelihood function that is estimated from the neural network. The dataset used for the challenge was used in different ways to train the neural network by inserting different values into the dataset for missing values, removing features with missing variables and splitting the dataset into three disjoint datasets which dependend on the number of jets in the events. The dataset that gave the highest discovery significance was the *FillMean* dataset which gave $Z = 429.963$ from the likelihood estimation. However, the *JetsTwo* dataset gave the highest AUC equal to 0.932, but because of less statistics it gave lower discovery significance than *FillMean*. The GitHub repository with the scripts used to produce the results in this report can be found at `https://github.com/martinfoell/UiO-FYS5429-Higgs-ML-Challenge`.

CONTENTS

I. INTRODUCTION

Our best understanding of particle physics is described by the Standard Model that describes the particles that we know and the interactions between them. In 2012 the Higgs particle was discovered by the ATLAS and CMS collaboration at CERN and is the latest particle that was discovered. In 2014 the ATLAS released the Higgs boson machine learning challenge where the objective was to use machine learning to discover the Higgs boson in a dataset consisting of simulated data with background events from a tau tau final state with the signal events that comes from a Higgs boson decaying into the tau tau pair. In the theory section we describe the theory behind neural networks and how the statistics is used in particle physics to discover particles with the use of a search region and the profile likelihood ration function. Then we explain how a neural network can be used to estimate the profile likelihood ration function which then can be used to calculate the discovery significance. In the method section we describe the Higgs dataset and its challenges with missing variables, and where we propose several methods for how to deal with the missing variables that gives different datasets that the neural network trains on. In the result section we present the output from the neural networks and compare them to each other. There we also implement the two methods for calculating the discover significance with the search region and from the estimation of the profile likelihood ratio function.

II. THEORY

A. Neural network

A neural network consists of neurons, also called nodes, in a network which are inspired by the neurons in the brain (Hjorth-Jensen, 2023). The node takes input from

the neighbouring nodes and uses an activation function f to calculate the input nodes with the weights w that describes the connection between the nodes. The output y that is connected to a set of nodes x_i with weights w_i is gives an

$$y = f\left(\sum_{i=1}^{n} w_i x_i\right) = f(u) \tag{1}$$

and is illustrated in Figure 1a. A neural network consist of layers with connected neurons, or also called nodes.

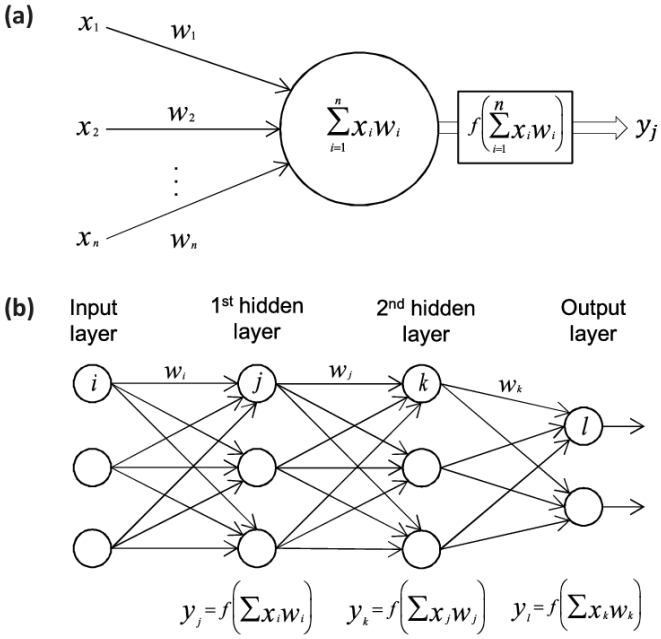


FIG. 1: Neural network, from (Hjorth-Jensen, 2023)

The output y of a node is given via the activation function f as

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b_i\right) = f(z), \tag{2}$$

where x_i is the input from the neurons in the preciding layers, and b_i is called the *bias* which is included in the case of a zero weight w_i or input x_i in the activation function. In e.g the first layer of a neural network we calculate the weighted sum z_i^1 for each node in the layer over the input coordinates x_j as

$$z_i^1 = \sum_{j=1}^{M} w_{ij}^1 x_j + b_i^1, \tag{3}$$

where M denotes all the possible inputs to a given node i in the first layer. We call the layer *fully connected* if all the nodes in the layer takes all the nodes from the previous layer as input, and M is then equal to the number of nodes in the previous layer. The activation function f_i for each node i takes the value of the activation function in Eq. (3) and gives us the output y_i^1 of all neurons in layer 1 as

$$y_i^1 = f(z_i^1) = f\left(\sum_{j=1}^{M} w_{ij}^1 x_j + b_i^1\right), \tag{4}$$

with the assumption that all the nodes in the same layer has the same activation function, and we therefore drop the subsctipt i on f_i. If the layers has different activation functions we can denote it with a superscript l for a layer l. The output of the node i for layer l therefore becomes

$$y_i^l = f(z_i^l) = f\left(\sum_{j=1}^{N_{l-1}} w_{ij}^l y_j^{l-1} + b_i^l\right), \tag{5}$$

where N_{l-1} is the number of nodes in the previous layer if the layer is fully connected. With Eq. (5) we can calculate the output y_i^l from layer l from the output y_i^{l-1} from the previous layer $l-1$. Given the input x_n from the input layer with n nodes, the output from layer $l+1$ can therefore be calculated as

$$y_i^{l+1} = f^{l+1}\left[\sum_{j=i}^{N_l} w_{ij}^{l+1} f^l\left(\sum_{j=i}^{N_{l-1}} w_{jk}\left(\right.\right.\right.$$
$$\left.\left.\left. ... f^1\left(\sum_{j=i}^{N_0} w_{mn}^1 x_n + b_m^1\right)...\right) + b_j^l\right) + b_i^{l+1}\right]. \tag{6}$$

Eq. (6) shows that the MPL is nothing more than an analytic function which is given by the real valued map $\hat{x} \in \mathbb{R}^n \to \hat{y} \in \mathbb{R}^m$. This concludes the feed forward method. Now we go over to the famous back propagation algorithm (cite) which is used to update the weights and biases in the neural network. To summarize the weights w_{ij}^l and biases b_j are updated according to

$$w_{jk}^l \leftarrow w_{jk}^l - \eta \delta_j^l y_k^{l-1} \tag{7}$$
$$b_j^l \leftarrow b_j - \eta \delta_j^l, \tag{8}$$

where δ_j^l is the error that is calculated from the derivative of the cost function $\mathcal{C}$ and η is the learning rate. A more detailed description of the back propagation algorithm can be found in (Hjorth-Jensen, 2023).

B. Discovery statistics and the profile likelihood ratio in particle physics

In particle physics we often test a so called *background hypythis* that test the physics we know against the *signal + background hypotisis* that test unknown physics that could possible discover by rejecting the background

hypothisis. One way of doing this is to calculate the discover significance Z which counts the number of standard derivations and is the square of some test statistics q_0. In a search region where there are b background events and s signal events the discover significance is given as

$$Z = \sqrt{2(s+b)\ln\left(1+\frac{s}{b}\right) - s}, \tag{9}$$

and is often used in particle physics. Another way of using the test statistics is to use the *profile likelihood ratio* and the signal strength μ which measures how strong the signal is. To test a hypothisis value for the signal strength μ we use the *profile likelihood ratio* (Cowan *et al.*, 2011)

$$\lambda(\mu) = \frac{L(\mu, \hat{\hat{\boldsymbol{\theta}}})}{L(\hat{\mu}, \hat{\boldsymbol{\theta}})}, \tag{10}$$

where in the numerator the value of θ that maximizes L for a specific μ is denoted by the quantity $\hat{\hat{\boldsymbol{\theta}}}$, and is therefore the conditional maximum-likelihood (ML) estimator that estimates $\boldsymbol{\theta}$, and therefore a function of μ. The denominator is the unconditional likelihood function where $\hat{\mu}$ and $\hat{\boldsymbol{\theta}}$ are their ML estimators.

By Wilks' theorem we can use the test statistics

$$q_0 = \begin{cases} -2\ln\lambda(0), & \text{if } n > b, \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

The appropiate likelihood ratio in searches for new particles is the ratio of the likelihood for the signal+background hypothisis and the likelihood for the background hypothisis and is given generally as

$$Q = \frac{L_{s+b}(\mu)}{L_b} \tag{12}$$

This likelihood ratio can be expressed in terms of the background only likelihood function L_b and signal only density function L_s and is given as (Read, 2000)

$$Q = e^{-s_{exp}} \prod_{i=1}^{n} \left(1 + \frac{s_{exp} L_s(x_i)}{b_{exp} L_b(x_i)}\right), \tag{13}$$

where b_{exp} and s_{exp} are the expected background and signal events, and the product is taken over a given experimental result x_i for each event i over the total number of events n. We insert the signal strength parameter μ in front of the expected number of signal events s_{exp}, which gives us

$$Q(\mu) = e^{-\mu s_{exp}} \prod_{i=1}^{n} \left(1 + \frac{\mu s_{exp} L_s(x_i)}{b_{exp} L_b(x_i)}\right). \tag{14}$$

The profile log-likelihood ratio $t(\mu)$ is therefore equal to

$$\begin{aligned} t(\mu) &= -2\ln Q(\mu) \\ &= \mu s_{exp} - \sum_{i=1}^{n} \left(1 + \frac{\mu s_{exp} L_s(x_i)}{b_{exp} L_b(x_i)}\right), \end{aligned} \tag{15}$$

when expressed in terms of L_s and L_b. In the profile likelihood ratio in Eq. (10) we see that the estimator value that maximizes the signal+background likelihood function is $\hat{\mu} = 1$ in the unconditional likelihood function in the denominator in Eq. (10). Using this the derivation in Eq. .. in Appendix gives us the relation

$$\begin{aligned} -2\ln\lambda(\mu) &= -2\ln Q(\mu) + 2\ln Q(1) \\ &= t(\mu) - t(1), \end{aligned} \tag{16}$$

If the estimated value $\hat{\mu}$ that maximises an *approximated* signal+background likelihood function is not equal to the expexted estimate $\hat{\mu} = 1$, as used to derive Eq. (16), we use the estimated value $\hat{\mu}$ instead in the relation above. This gives us the same relation as above, but where we replace 1 by $\hat{\mu}$ in the last term

$$\begin{aligned} -2\ln\lambda(\mu) &= -2\ln Q(\mu) + 2\ln Q(\hat{\mu}) \\ &= t(\mu) - t(\hat{\mu}), \end{aligned} \tag{17}$$

which is the more general form. This gives us the test statistic

$$q_0 = \begin{cases} t(0) - t(\hat{\mu}), & \text{if } n > b_{exp}, \\ 0, & \text{otherwise}, \end{cases} \tag{18}$$

and the discovery significance Z is then given as

$$Z = \sqrt{q_0} = \begin{cases} \sqrt{t(0) - t(\hat{\mu})}, & \text{if } n > b_{exp}, \\ 0, & \text{otherwise}. \end{cases} \tag{19}$$

C. Likelihood estimation

The likelihood ratio function can be estimated by a neural network with a method developed by G. V. Moustakides and K. Basioti (Moustakides and Basioti, 2019) which transforms the output of the neural network to a likelihood ratio by solving an optimalization problem. For a random vector X with the probability function $f_0(X)$ we want to optrimalisze the following loss function

$$\mathcal{J}(u) = \mathrm{E}_0[\phi(u(X)) + \mathrm{r}(X)\psi(\mathrm{u}(X))], \tag{20}$$

where $\mathrm{E}_0[\cdot]$ is the expectation value w.r.t. $f_0(X)$, and $\phi(z)$ and $\psi(z)$ are scalar functions of the scalar z, and $\mathrm{u}(X)$ and $\mathrm{r}(X)$ are scalar functions of X were $\mathrm{r}(X)$ takes values in a *known* interval $\mathcal{I}_r$. The optimalization problem we want to solve is

$$\min_{\mathrm{u}(X)} \mathcal{J}(u) = \min_{\mathrm{u}(X)} \mathrm{E}_0[\phi(u(X)) + \mathrm{r}(X)\psi(\mathrm{u}(X))], \tag{21}$$

such that when we have a known scalar function $\omega(r)$ of the scalar r, we want to determine the two functions $\phi(z)$ and $\theta(z)$ such that the global minimizer in Eq. (**??**) is equal to

$$\mathrm{u}(X) = \omega(\mathrm{r}(X)). \tag{22}$$

In (Moustakides and Basioti, 2019) they show that the nessesary condition to achive the optimilization that satisfies this condition is that

$$\phi'(\omega(\mathrm{r})) + \mathrm{r}\psi'(\omega(\mathrm{r})) = 0. \tag{23}$$

If $\omega(r)$ is stighly monotone we also have the result

$$\phi'(\omega(\mathrm{r})) + \omega^{-1}(z)\psi'(\omega(\mathrm{r})) = 0, \tag{24}$$

for $z \in \omega(\mathcal{I}_r)$ where $\omega^{-1}(z)$ is the inverse of r. They further show that when we chose a function $\rho(z)$ with the condition $\rho(z) < 0$ for all $z \in \omega(\mathcal{I}_r)$ and define $\phi(z)$ and $\theta(z)$ as

$$\psi'(z) = \rho(z), \text{ and } \phi'(z) = \omega^{-1}(z)\rho(z), \tag{25}$$

then the minima of $\phi(z)+r\theta(z)$ in Eq. (21) for $z \in \omega(\mathcal{I}_r)$ is uniquely located at $z = \omega(r)$. This is what we wanted to achive in Eq. (22). When we chose

$$\omega(z) = \frac{z}{z+1} \text{ and } \rho(z) = -\frac{1}{z}, \tag{26}$$

and use Eq. (25) we get the following expressions for $\phi(z)$ and $\theta(z)$,

$$\phi(z) = -\log(1-z) \text{ and } \psi(z) = -\log(z). \tag{27}$$

This combination of $\phi(z)$ and $\theta(z)$ gives us the binary cross-entropy method and is among the most popular methods for classification problems. When we define the scalar function $\mathrm{r}(X)$ in Eq. (20) to be the likelihood ratio

$$\mathrm{r}(X) = \frac{f_1(X)}{f_0(X)}, \tag{28}$$

of two probability distributions $f_0(X)$ and $f_1(X)$ the loss function $\mathcal{J}(u)$ in Eq. (20) becomes

$$\begin{aligned}\mathcal{J}(u) &= \mathrm{E}_0[\phi(u(X)) + \mathrm{r}(X)\psi\big(\mathrm{u}(X)\big)] \\ &= \mathrm{E}_0[\phi(u(X))] + \mathrm{E}_1[\psi\big(\mathrm{u}(X)\big)],\end{aligned} \tag{29}$$

where the expextation values E_0 and E_1 are w.r.t the probability functions $f_0(X)$ and $f_1(X)$ respectively. We are now interested in finding a way of using a neural network to estimate the transformation $\omega(\frac{f_1(X)}{f_0(X)})$ when we have two data samples $\{X_1^0, ..., X_{n_0}^0\}$ and $\{X_1^1, ..., X_{n_1}^1\}$ that are sampled from the *unknown* probability densities $f_0(X)$ and $f_1(X)$ respectively, and when $\omega(r)$ is a *known* scalar function. Two approximations is needed to do this. For the first one we replace the statistical expectation values in Eq. (29) with the averages over the available data, while for the second we replace the function $\mathrm{u}(X)$ with the output of a neural network $\mathrm{u}(X, \theta)$, where θ summatizes the network parameters. The first approximation requires a dataset with a large sample size, while the second approximation requires that the neural network is sufficently rich such that it can approximate any non-linear function. When applying the two approximations above, the loss function in Eq. (29) is approximated to

$$\mathcal{J}(u) \approx \hat{\mathcal{J}}(\theta) = \frac{1}{n_0}\sum_{i=1}^{n_0}\phi(\mathrm{u}(X_i^0, \theta)) + \frac{1}{n_1}\sum_{i=1}^{n_1}\phi(\mathrm{u}(X_i^1, \theta)) \tag{30}$$

The optimalization problem is now transformed into an optimalization problem over the neural network parameters θ as

$$\begin{aligned}\min_{\mathrm{u}(X)} \mathcal{J}(u) &\approx \min_{\theta} \hat{\mathcal{J}}(\theta) \\ &= \min_{\theta}\left\{\frac{1}{n_0}\sum_{i=1}^{n_0}\phi(\mathrm{u}(X_i^0, \theta)) + \frac{1}{n_1}\sum_{i=1}^{n_1}\phi(\mathrm{u}(X_i^1, \theta))\right\},\end{aligned}$$

and when optimized the optimal parameters θ_o will give a neural network that has approximated the function $\omega(r)$, that is

$$\mathrm{u}(X, \theta_o) \approx \omega(\mathrm{r}(X)), \tag{31}$$

provided that it does not reach a local minima.

D. Estimating the profile log-likelihood ratio with neural network

We can use the method of estimating a likelihood ration described in the previous section when discovering new particles by estimating the profile log-likelihood ratio in Eq. (10). When we are interested to test the signal+background hypothisis against the background only hypothisis, which is given in the likelihood ratio function in Eq. (15), the likelihood ratio L_s/L_b is optimal to estimate when we have two datasets X_s and X_b which contains the signal and background data respectively

III. METHODS

A. The Model architecture

The architecture of the neural network consist of two hidden layers that each has 1000 nodes and where we use the RelU activation function. The input layer has a varying number of nodes depending on how the higgs dataset is used which will be described in Section ... The output of the neural network is given from one node where we use the Sigmoid activation function to keep the output between 0 and 1. We use the binary cross entropy loss function which also will be used to estimate the log likely ratio function. The L2 reguralization is used.

B. The Higgs dataset and collision physics

The dataset used for the HiggsML challenge consist of the output of simulated collisions, called *events*, and

is provided by the ATLAS experiment at CERN. The HiggsML dataset consist of 35 features which are given in Table ... (remember to put in Weight, count) in which of them contains the information about the kinematics of the outgoing particles in the events. We do not include the features `EventId`, `Weiht`, `KaggleSet` and `KaggleWeight` in the training of the neural network since they are not relevant for the physics of the collisions. In order to understand how we can use the dataset in the neural network, and the challenges that it gives we will give a brief introduction to the physics of the collisions in the ATLAS detector that has been used to simulate the events for the dataset.

The ATLAS detector is one of the detectors at the Large Hadron Collider (LHC) where two opposing proton beams collide and the outgoing particles from the collision is measured by the detector and is called the final state particles. Figure 2 shows the referance frame of the sylindrical ATLAS detector where the two proton beams going in opposite direction along the z-axis collide in the origin of the coordinate system.

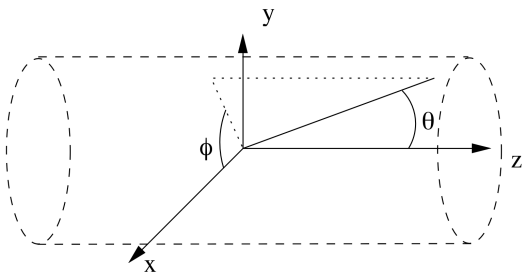


FIG. 2: ALTAS reference frame

The azimuthal angle ϕ is used to measuasre the direction of the particles in the x-y plane while the polar andlge θ is used to meusure the direction of the particle from the z-axis. In particle physics we often use what is called the *psuedorapididy* η instead of θ, and is given as $\eta = -\ln(\tan(\theta/2))$. The momentum of a particle meausured in the x-y plane is called the *transverse momentum* and is denoted p_T. The final state particle present in the events used for the HiggsML dataset is a tau tau pair that could come from the decay of e.g a Z boson or the Higgs boson. We call an event where the tau tau final state has decayed from a Higgs boson a *signal event* and an event where the two tau's comes from the decay of other particles like the Z boson a *background event*.

The goal of the Higgs challange is to be able to distingusihe the signal and background events from each other in order to discover the Higgs boson, and it is therefore a binary classification problem. In the HiggsML dataset the signal and background events are denoted by s and b respectively in the feature named `Label`, and we replace s and b by 1 and 0 respectively such that the dataset can be used in the neural network. This is also why we use one output node to classify the event.

In addition to the tau particle there may also be jets present in the event that comes from the hadronisation of quarks. The features include the kinematics from the particles like the transverse momentum of the taus and jets in addition to the angles ϕ and η from Figure ... . Not all events have jets, so for these events the features assisated to the leading and subleading jet does not have any values. Table ... shows the features in the dataset marked with a checkmark if they have a value for the different events with 0, 1 or 2-3 jets. We call the variables that does not have a value for every event a *missing variable* and becomes a challange for the neural network since it requires that every feature has a value. We describe how to deal with the missing variabels in the data set in Section . In addition some events also has what is called *jets* that is produced from the hadronization of the quarks in the detector. It is common to order the jets after their transverse momentum p_T, and we call the jet with the highest p_T the *leading jet* and the jet with the next to highest p_T the *subleading jet*. When it comes to the HiggsML dataset we will study closer how we can deal with the events in the dataset that has either no subleading jet, or both no subleading and leading jet. In the HiggsML datasets the number of jets is described by the feauture `PRI_num_jet` and takes the values 0, 1, 2 and 3.

TABLE I: Fraction of jets

n_{jets}	Events	Background [frac.]	Signal [frac.]
0,1,2 and 3	818238	0.658	0.342
0	327371	0.747	0.253
1	252882	0.643	0.357
2 and 3	237985	0.552	0.448

We observe that the fraction of signal and background events is not balanced with the execption of the events with 2 and 3 jets where they are more balanced.

..

C. Construction of Datasets and Missing variables

To deal with the missing variable in the dataset we will use three different methods that can be summarized as;

I. Inserting values into the entries with missing values

II. Remove the features that has missing variables

III. Split the data set into multiple disjoint data sets according to the number of jets, and then remove the features with missing values in the disjoint data sets

Inserting values

We can use several methods to insert values into the entries that has missing values in the data set. Two of the methods that we use here is to either insert the mean value from each jet variable in the missing entries or insert zero in all the missing entries. We will reffer to these methods as *fillMean* and *fillZero* respectively. For the third method that we use is to use the observation that the ϕ variables is almost uniformly distributed between $-\pi$ and π as shown in Figure ... From this observation we use to insert a random variable between $-\pi$ and π for the missing entries in the ϕ variables and then insert the mean value in the other jets variables. We refer to this method as *fillPhiRandom*. In all these datasets none of the feautures are removed so there are 30 feautures in each of these datasets.

Removing features

Another way of dealing with the missing variables is to remove the features that has missing values in them. These feautures include the jet variables and the DERmass variable. In the first method with removing feautures we remove all the jet feautures to get rid of the missing variables in the data set. Because the DERmass variable only has 15% missing variables and is not directly conneted to the jets we do not remove it, but insert the mean value instead. We refer to this method as *removeJets*. The number of feautures in this dataset is reduced from the to 20 feautures from the original Higgs dataset. In the second model we also in addition remove all the ϕ features since they have an almost uniform distrubution, and therefore a large variance. By doing this we get a measure on how important ϕ features are for the prediction of the background and signal events, and we refer to this method as *removePhi*. The number of feautures in this dataset is therefore reduced to 25 feautures compared to the original dataset.

1. Disjont datasets

The third way that we will use to deal with the missing variables is to split the data sets into three disjiont data sets according to the number of jets in the events, and then remove the feautures that has missing variables in the three cases respectively. The dataset is split into the events with 0, 1 and 2-3 jets respectively. In the case with 2-3 jets, none of the features are removed, while for 1 the features related to the subleading jet are removed, and for 0 jets both the features related to the leading and subleading jets are removed. In the datasets with 0 and 1 jets we also get some redundant features that either is unnecessary or that two feautures contain the same information. This is the case for the features PRI_jet_num and PRI_jet_all_pt for both the events with 0 and 1 jets. The feauture PRI_jet_num counts the number of jets in the event and is therefore unnecessary since it contains 0 or 1 for the events with 0 and 1 jets respectively. The feauture PRI_jet_all_pt gives us the sum of the p_T of all the jets in the event. In events with 0 jets the feature PRI_jet_num contains only 0 values since there are not jets in those events, and are therefore unnecessary. For the events with 1 jet the feature PRI_jet_all_pt gives the same value as the p_T of the leading jet in the event given in the feauture PRI_jet_leading_pt, and the information in PRI_jet_all_pt is therefore redundant. We will refer to the datasets with 0, 1 and 2-3 jets as *JetsNone*, *JetsOne* and *JetsTwo* respectively, and contains 18, 21 and 30 feautures respectively.

n_{jets}	Events	Background [frac.]	Signal [frac.]
0,1,2 and 3	818238	0.658	0.342
0	327371	0.747	0.253
1	252882	0.643	0.357
2 and 3	237985	0.552	0.448

TABLE II: Fraction of jets

IV. RESULTS

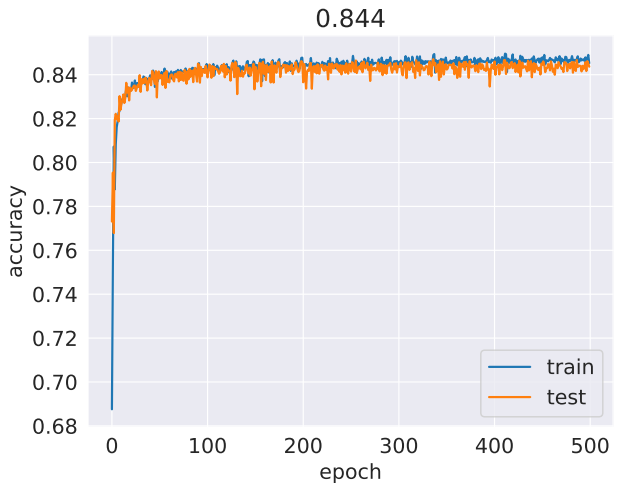
The results that follows are obtained from a grid search where we present best results with the accuracy in a grid with the values of the learning rate $\lambda \in$ [.....] and $\eta \in [1, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-5}]$. The other results from the grid search can be found in the GitHub repository[1]. I all of the training of the neural networks for the different datasets explained in Section III.C we use a batch size of 50 000 and train it for 500 epochs.

A. FillMean dataset

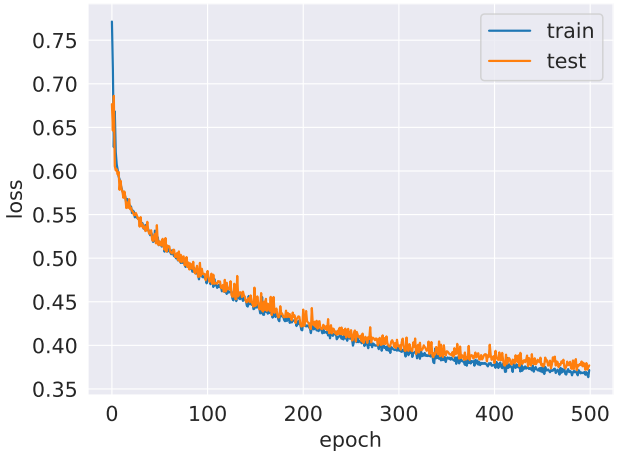
Figure 3 shows the output from the neural network with the dataset *FillMean* described in Section III.C with the accuracy in Figure 3a, loss in Figure 3b and the ROC curve in Figure 3c. The values for the hyperparameters that gave the best results for the neural network from the grid search are $\eta = 1$ and $\lambda = 0.0001$. We see that for both the accuracy and the loss from the training and test data are similar to each other respectively indicating that there is little over training. In addition the accuracy is stable over the epochs while the loss seems to approach a minimum. This indicates that the neural network has learned much of what it can learn from the dataset. The

[1] test

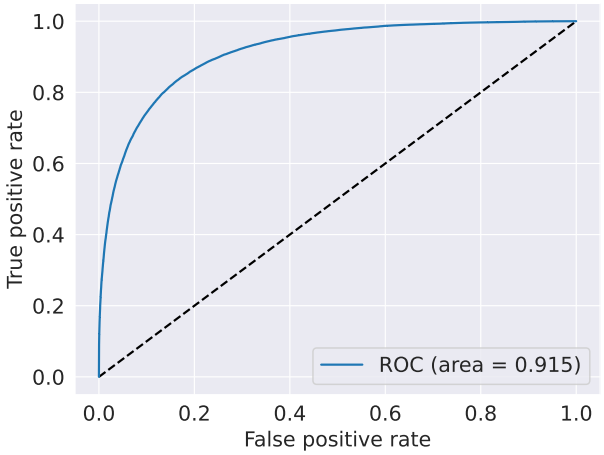
accuracy of the training dataset is 0.844 after the 500th epcoh, while the area under the ROC curve (AUC) in Figure 3c is 0.915.



(a) Train (blue) and test (orange) accuracy.



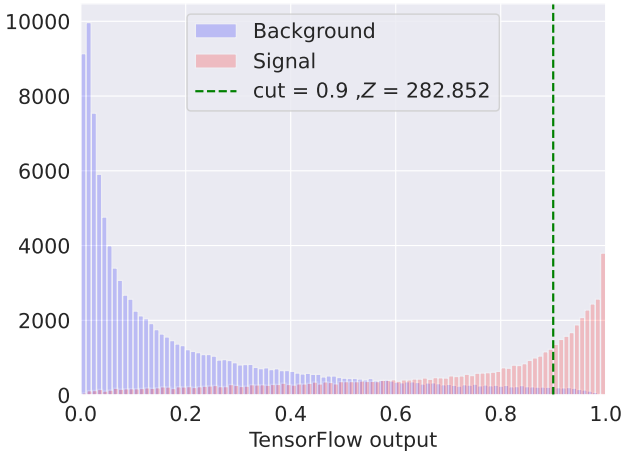
(b) Train (blue) and test (orange) loss.



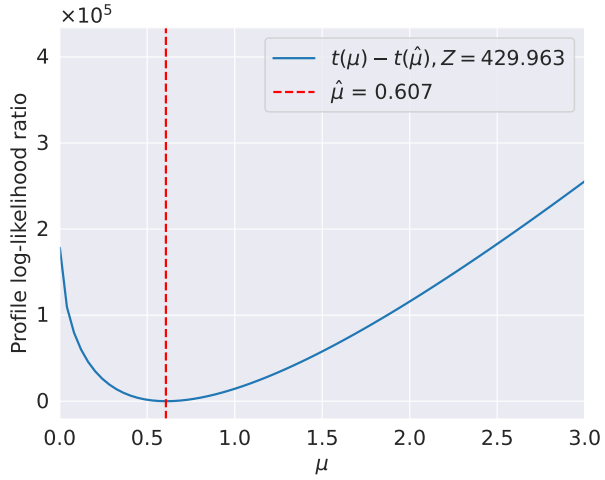
(c) ROC curve of the neural network output.

FIG. 3: The accuracy (a), loss (b) and the ROC curve (c) of the neural network output from the *FillMean* dataset with $\lambda = 1$ and $\eta = 0.0001$.

Figure 4 shows the distribution of the true background and signal events over the output from the neural network in Figure 4a and the log-likelihood ratio of the signal + background and the background estimation $t(\mu)$ in Figure 4a which is shifted by the minimum of $t(\mu)$ with $t(\mu_{min})$. We observe that the distribution of the true background and signal events in Figure 4a are mostly skewed towards 0 and 1 respectively which tells us that most of the events in the dataset has been correctly classified. To calculate the Z score we do a cut at 0.9 as shown by the green dashed line Figure **??**. Counting the background and signal events above 0.9 gives and using Eq. ... gives us $Z = ...$ The estimated likelihood ratio from the neural network is used to find the In Figure 4a we do a cut at 0.9 and obtain $Z_{cut} = 282.852$ when we have used Eq. ... with the number of background and signal events above 0.9 in Figure 4a. From Figure 4c we get the $Z_{likelihood} = 429.963$ by using Eq. ... when $\mu_{min} = 0.607$. The μ value where $t(\mu)$ has it minimum is not at $\mu = 1$ which would be expected from the signal + background hypotosis, and therefore indicates that the neural network has not learned the profile likelihood ratio correctly. The condition for the likelihood ratio estimation from Section II.C was that the loss function should reach a minima in the training, and Figure 3b shows that the loss function seems to reach a minima, but it that it has not converged to one.



(a) dd



(b) dd

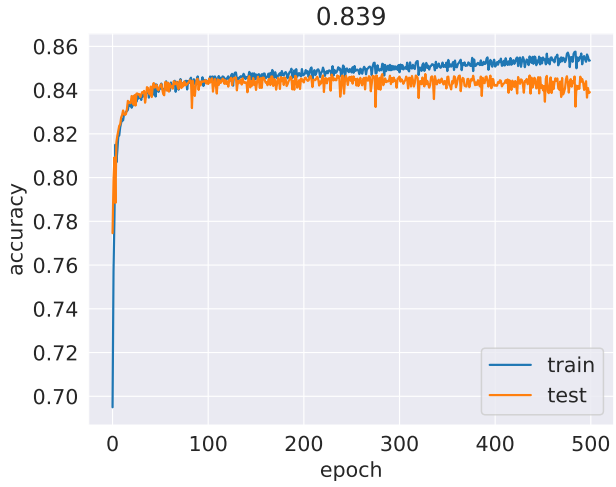
(c)

FIG. 4: FillMean

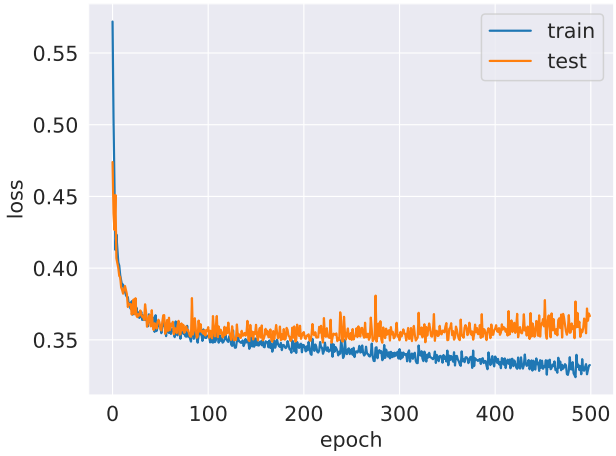
B. FillZero

Figure 5 shows the output from the neural network with the dataset *FillMean* described in Section III.C where missing entries are filled with zero and shows the accuracy in Figure 5a, loss in Figure 5b and the ROC curve in Figure 5c. The values for the hyperparameters that gave the best results for the neural network from the grid search are $\eta = 1$ and $\lambda = 10^{-6}$. We see that there is a descrepancy between the values from the training and the testing for both the accuracy and the loss where the accuracy and loss becomes stable for the test dataset, while the accuracy contunies grow and the loss continues to decrease for the training dataset. This can be a sign of the model starting to overfitt the data. The accuracy of the training dataset is 0.839 after the 500th epcoh, while the AUC in Figure 5c is 0.913. Filling the missing entries with zero in particle physics is always not physical since objects, like a jet that is missing would not have any momentum or a direction, and it would therefore cause a bias by inserting a value in the dataset, even if it is zero.

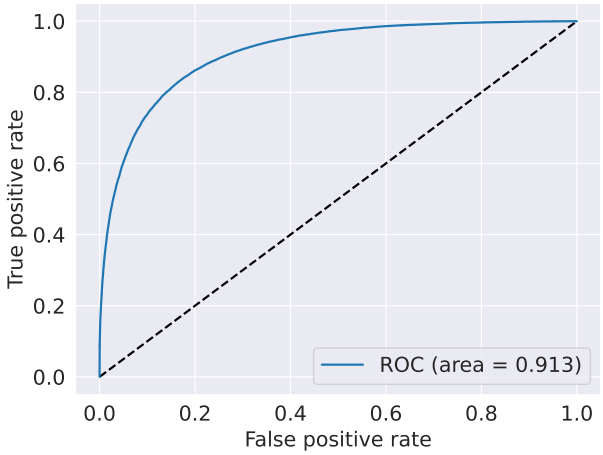
Figure 6 shows the distribution of the true background and signal events over the output from the neural network in Figure 6a, and the log-likelihood ratio of the signal + background and the background estimation $t(\mu)$ in Figure 6a. We observe that the distribution of the true background and signal events in Figure 6a are mostly skewed towards 0 and 1 respectively which tells us that most of the events in the dataset has been correctly classified. In Figure 6a we do a cut at 0.9 and obtain $Z_{cut} = 290.612$ when we have used Eq. ... with the number of background and signal events above 0.9 in Figure 6a. The log-likelihood function in Figure 6b shifted by the minimum of $t(\mu)$, $t(\mu_{min})$, where the minima of the log-likelihood is located at $\mu_{min} = 0.647$. The minima is not located at $\mu = 1$ which is the expected signal strength under the signal+background hypothosi, and this indicates that the neural network has not learned the likelihood ratio from from the dataset correctly. By using Eq. ..., the $Z_{likelihood}$ is found to be $Z_{likelihood} = 483.050$. The condition for the likelihood ratio estimation from Section II.C was that the loss function should reach a minima in the training, and Figure 5b shows that the loss function seems to reach a minima, but it that it has not converged to one.



(a) Train (blue) and test (orange) accuracy.

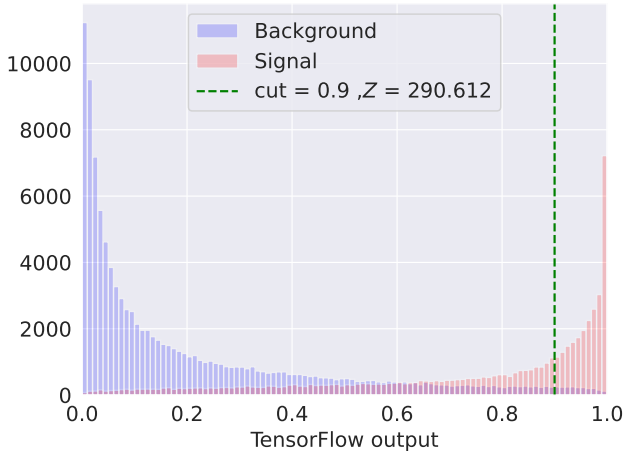


(b) Train (blue) and test (orange) loss.

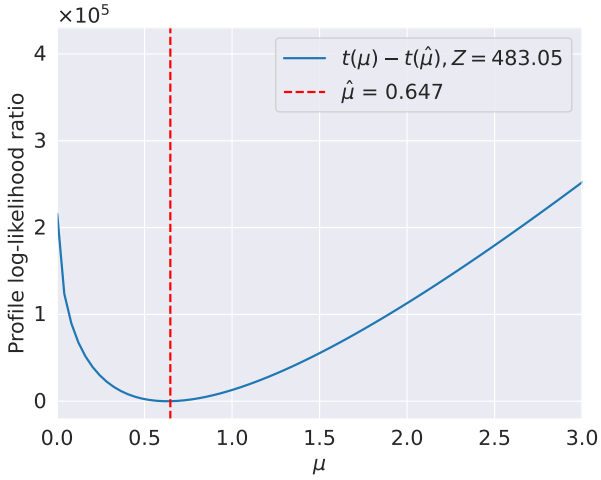


(c) ROC curve of the neural network output.

FIG. 5: The accuracy (a), loss (b) and the ROC curve (c) of the neural network output from the *FillZero* dataset with $\lambda = 1$ and $\eta = 0.0001$.



(a) dd



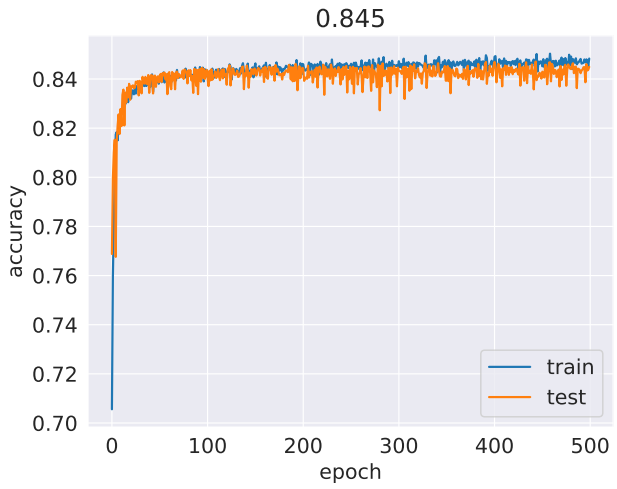
(b) dd

FIG. 6: FillZero

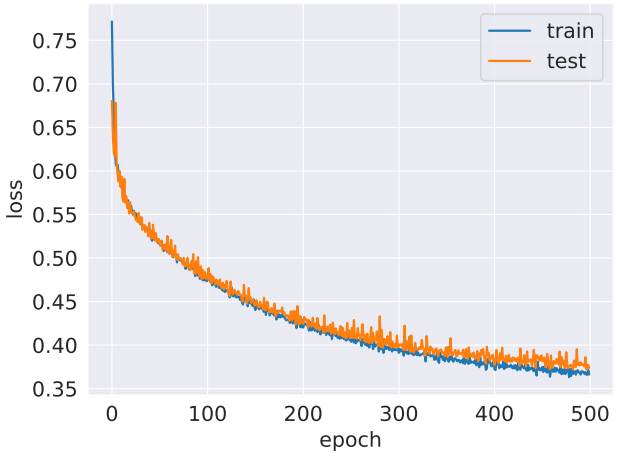
C. FillPhiRandom

The results from the neural network for the *FillPhiRandom* dataset described in Section III.C are shown in Figure 7 where Figure 7a shows the accuracy, Figure 7b shows the loss, and Figure 5c shows the ROC curve. The values for the hyperparameters that gave the best results for the neural network from the grid search are $\eta = 1$ and $\lambda = 0.0001$. When we compare these results with the ones from the *FillMean* dataset we see that they are almost identical. Both the accuracy and the loss follows the same shape, and the training and testing results are almost the same. The accuracy of the training dataset is 0.845 after the 500th epcoh, while the AUC in Figure 7c is 0.915, which also are almost identical to the results from the *FillMean* dataset. The fact that the results from the training on the *FillMean* and *FillPhiRandom*

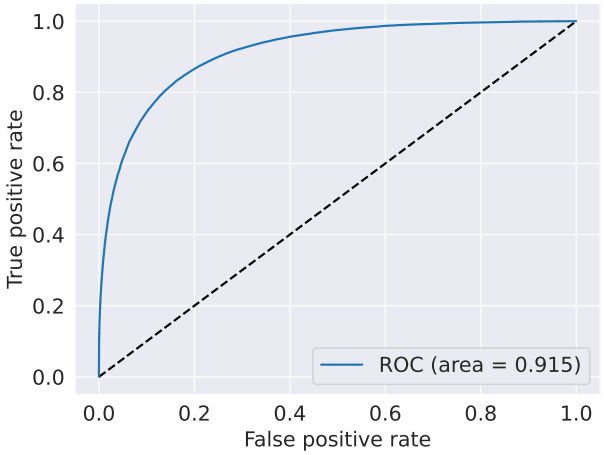
datasets almost gives the same reslults can be undersood from the fact that the only varialbes that have different values between them is the ϕ variables which are almost uniform as shown in Figure ... in Appendix. This means that filling the missing entries with the mean of that variable or a random number between $-\pi$ and π indicated that the ϕ feautures are not of high importance compared to the other features that the datasets include. From a physics view (if time).



(a) Train (blue) and test (orange) accuracy.

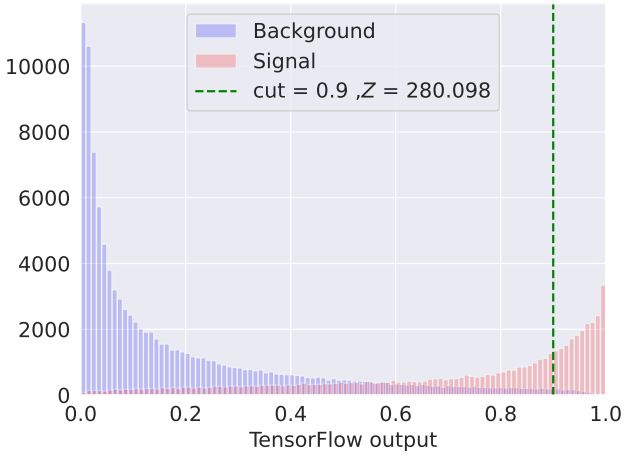


(b) Train (blue) and test (orange) loss.

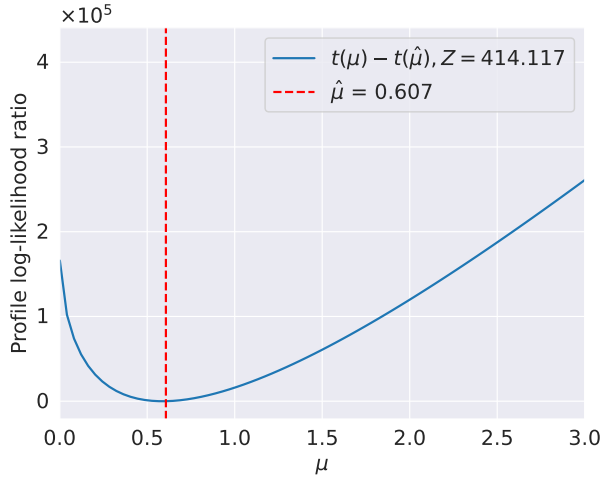


(c) ROC curve of the neural network output.

FIG. 7: The accuracy (a), loss (b) and the ROC curve (c) of the neural network output from the *FillPhiRandom* dataset with $\lambda = 1$ and $\eta = 0.0001$.



(a) dd

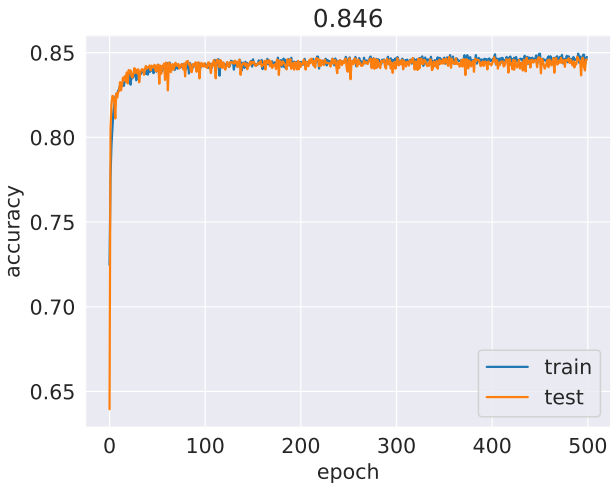


(b) dd

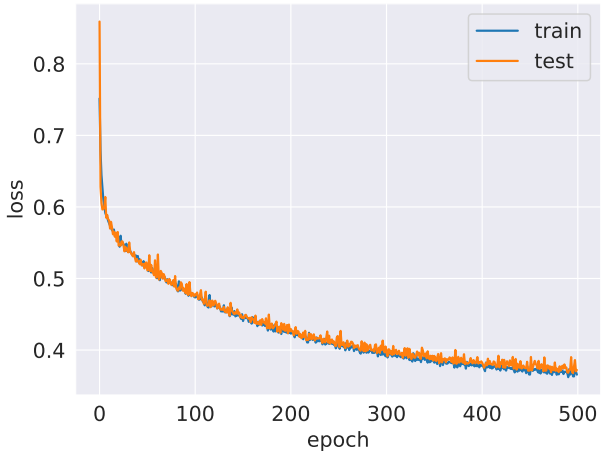
FIG. 8: FillPhiRandom

D. RemovePhi

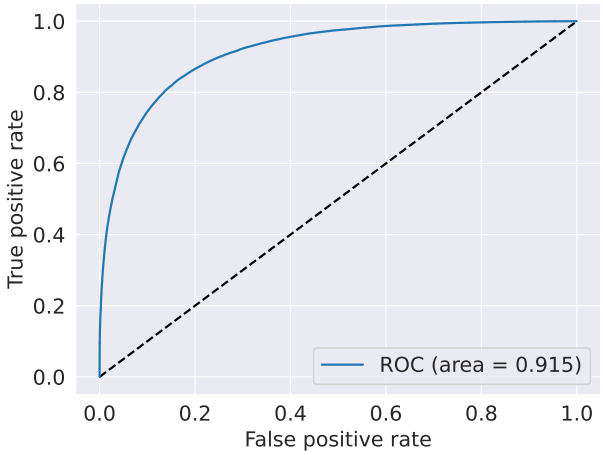
For the dataset *RemovePhi* where we remove the ϕ angle features the results of the neural network are shown in Figure 9a with the accuracy in Figure 9a, loss in Figure 9b and the ROC curve in Figure 11c. The values for the hyperparameters that gave the best results for the neural network from the grid search are $\eta = 1$ and $\lambda = 0.0001$, similar to the training on the *FillMean* and *FillPhirandom* datasets. In this case as well the results from the accuracy, loss and ROC curve are similar to the results for the *FillMean* and *FillPhiRandom* datasets, and accuracy of the training dataset is 0.844 after the 500th epcoh, while the area under the ROC curve (AUC) in Figure 3c is 0.915. This gives us another inducation that the ϕ features are not important for the training since the results are the almost the same when we include them in the dataset. We see that for both the accuracy and the loss from the training and test data are similar to each other respectively indicating that there is little over training. In addition the accuracy is stable over the epochs while the loss seems to approach a minimum. This indicates that the neural network has learned much of what it can learn from the dataset. The accuracy of the training dataset is 0.844 after the 500th epcoh, while the area under the ROC curve (AUC) in Figure 3c is 0.915.



(a) Train (blue) and test (orange) accuracy.

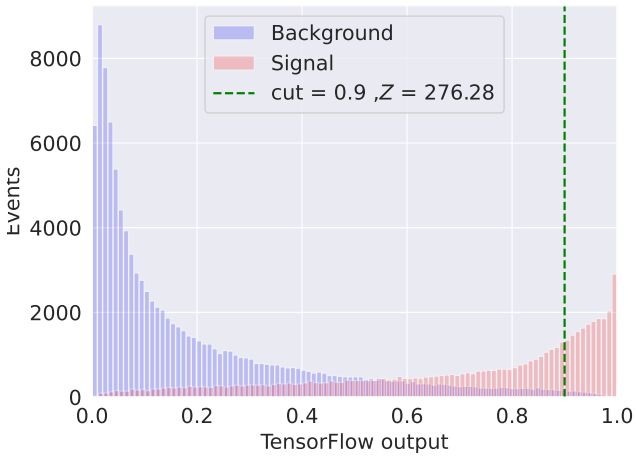


(b) Train (blue) and test (orange) loss.

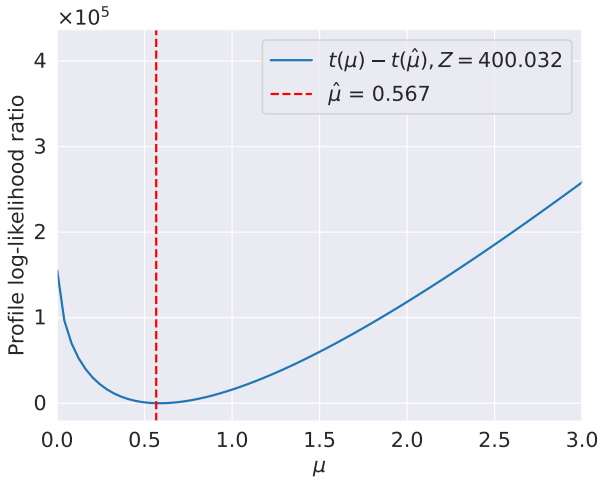


(c) ROC curve of the neural network output.

FIG. 9: The accuracy (a), loss (b) and the ROC curve (c) of the neural network output from the *RemovePhi* dataset with $\lambda = 1$ and $\eta = 0.0001$.



(a) dd



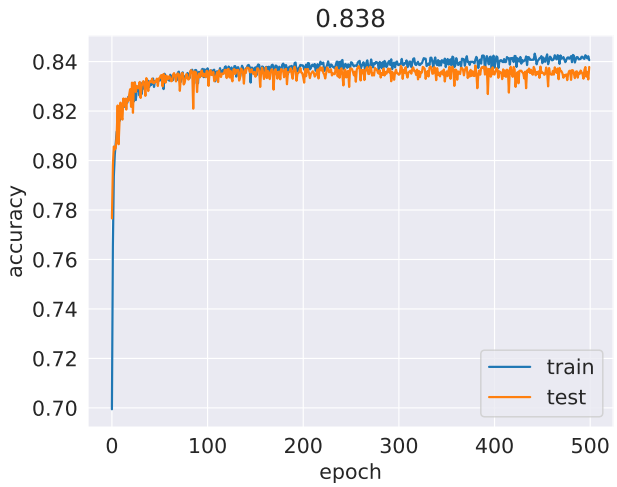
(b) dd

FIG. 10: RemovePhi

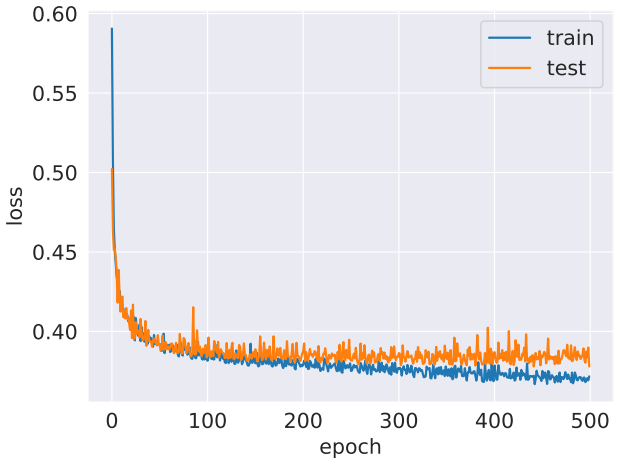
E. RemoveJets

Figure 11 shows the output from the neural network with the dataset *RemoveJets* described in Section III.C where we remove the jets feautures, with the accuracy in Figure 11a, loss in Figure 11b and the ROC curve in Figure 11c. The values for the hyperparameters that gave the best results for the neural network from the grid search are $\eta = 1$ and $\lambda = 0.0001$. We see that for both the accuracy and the loss from the training and test data are similar to each other respectively indicating that there is little over training, but that there are a small descrepancy between them. In addittion to this the accuracy and loss converges to a value which indicates that the model has learned what it can from the dataset. That it converges faster than for e.g *FillMean* comes from the fact that it has fewer input feauters which leads to less

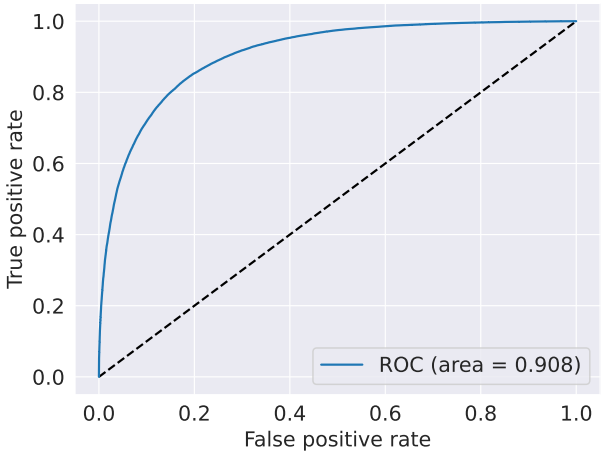
information. This shows that in this case it is not an advantage to remove feautures that has missing variables. If the variables had a larger fraction of missing entries then it would maybe advatougus to remove these feautures since any filling of these variables might lead to more noise in the dataset. In addition the accuracy is stable over the epochs while the loss seems to approach a minimum. This indicates that the neural network has learned much of what it can learn from the dataset. The accuracy of the training dataset is 0.844 after the 500th epcoh, while the area under the ROC curve (AUC) in Figure 3c is 0.915.



(a) Train (blue) and test (orange) accuracy.

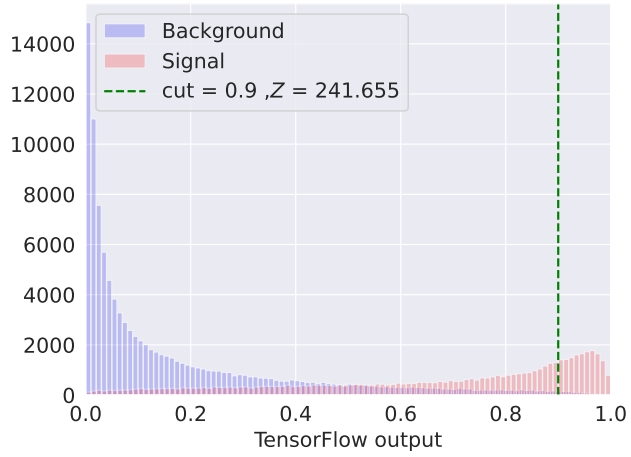


(b) Train (blue) and test (orange) loss.

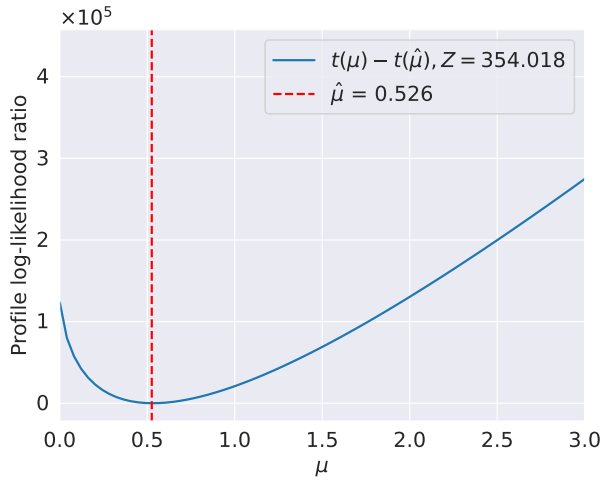


(c) ROC curve of the neural network output.

FIG. 11: The accuracy (a), loss (b) and the ROC curve (c) of the neural network output from the *RemoveJets* dataset with $\lambda = 1$ and $\eta = 0.0001$.



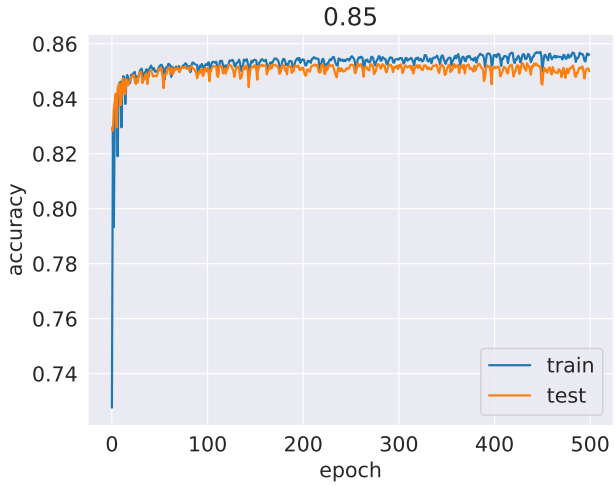
(a) dd



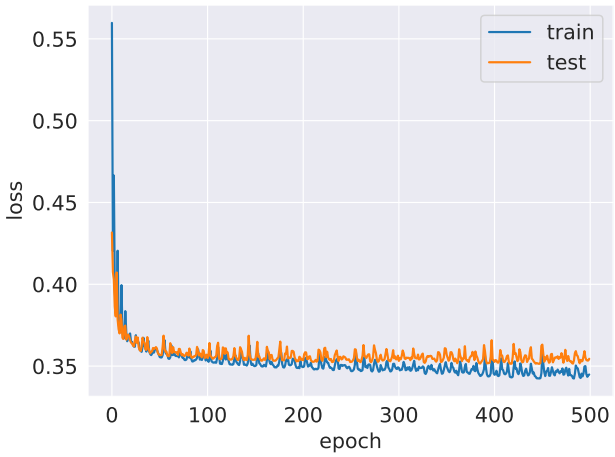
(b) dd

FIG. 12: RemoveJets

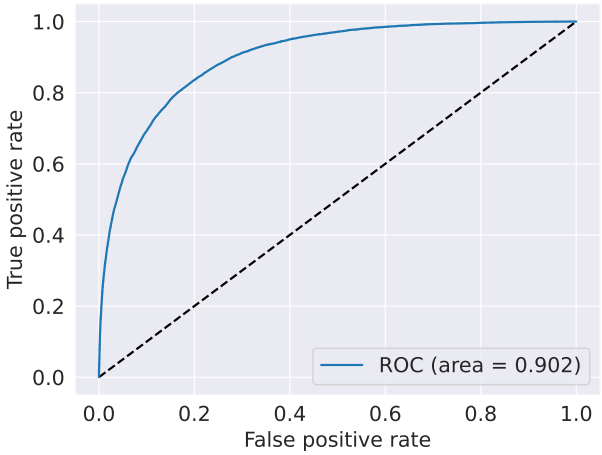
F. JetsNone



(a) Train (blue) and test (orange) accuracy.

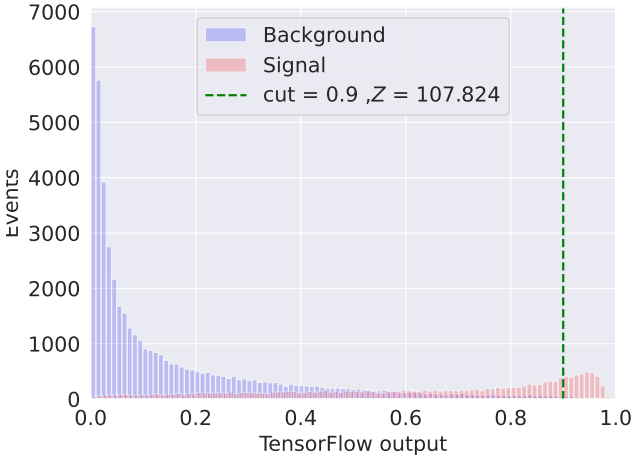


(b) Train (blue) and test (orange) loss.

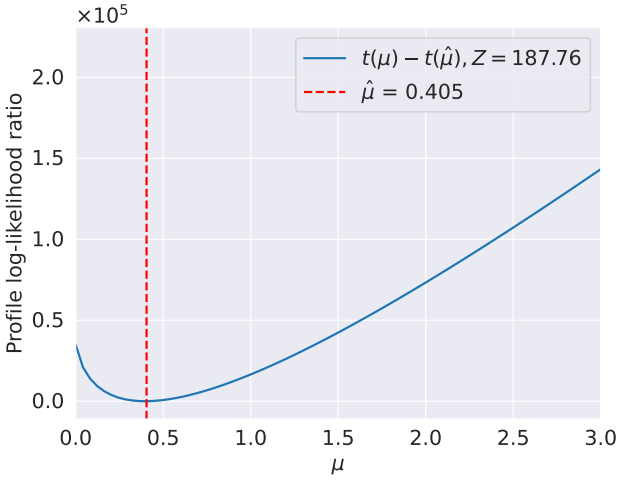


(c) ROC curve of the neural network output.

FIG. 13: The accuracy (a), loss (b) and the ROC curve (c) of the neural network output from the *JetsNone* dataset with $\lambda = 1$ and $\eta = 0.0001$.



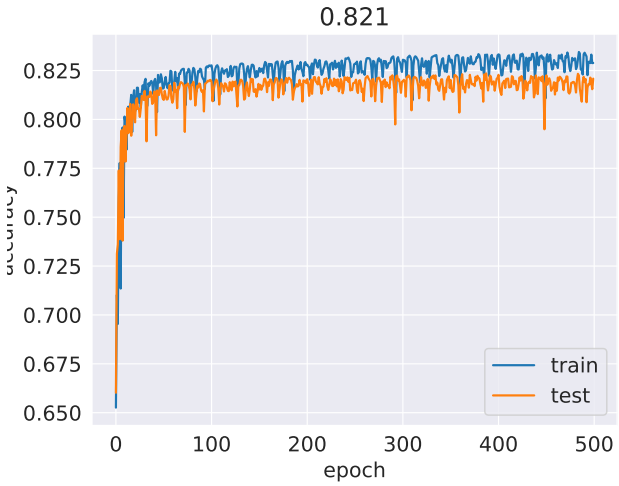
(a) dd



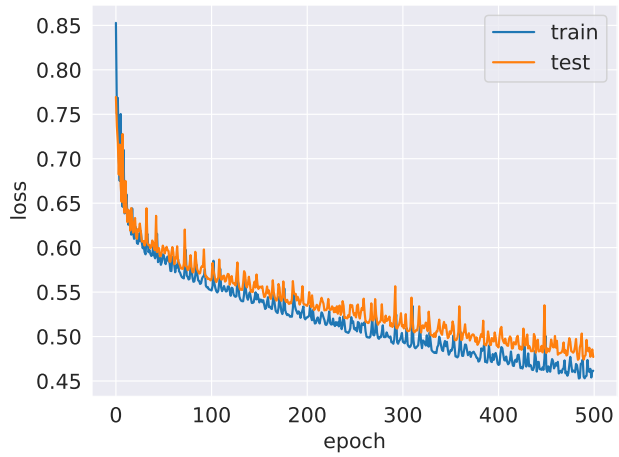
(b) dd

FIG. 14: JetsNone

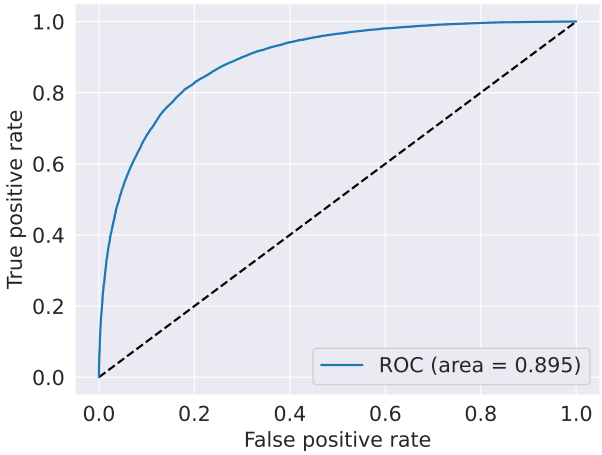
G. JetsOne



(a) Train (blue) and test (orange) accuracy.

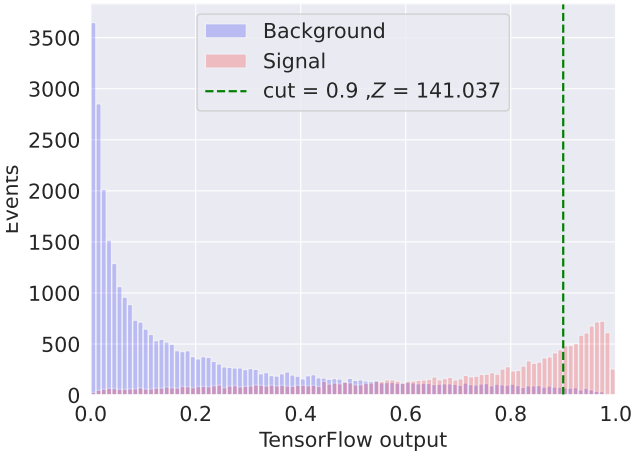


(b) Train (blue) and test (orange) loss.

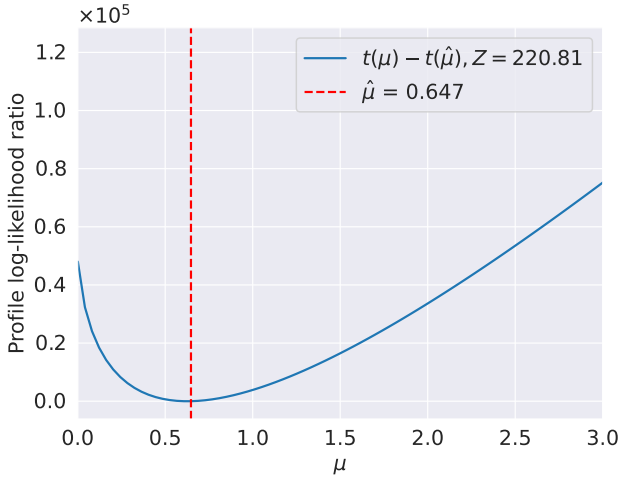


(c) ROC curve of the neural network output.

FIG. 15: The accuracy (a), loss (b) and the ROC curve (c) of the neural network output from the *JetsOne* dataset with $\lambda = 1$ and $\eta = 0.0001$.



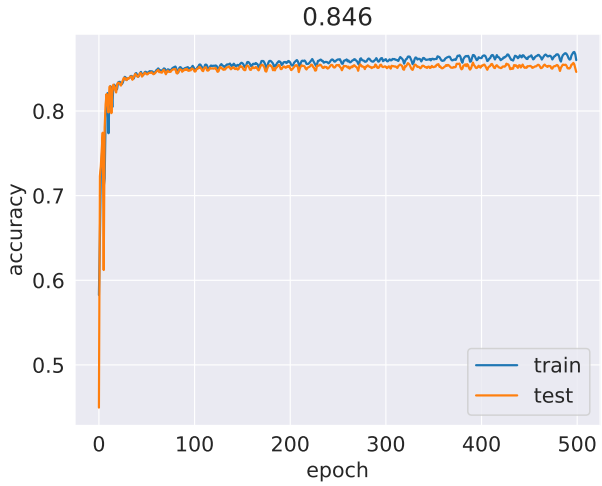
(a) dd



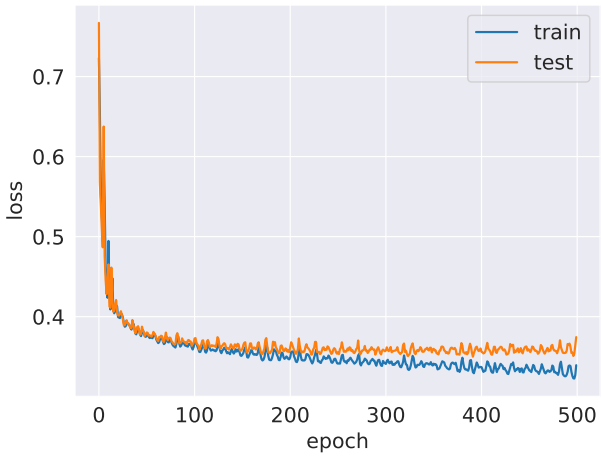
(b) dd

FIG. 16: JetsOne

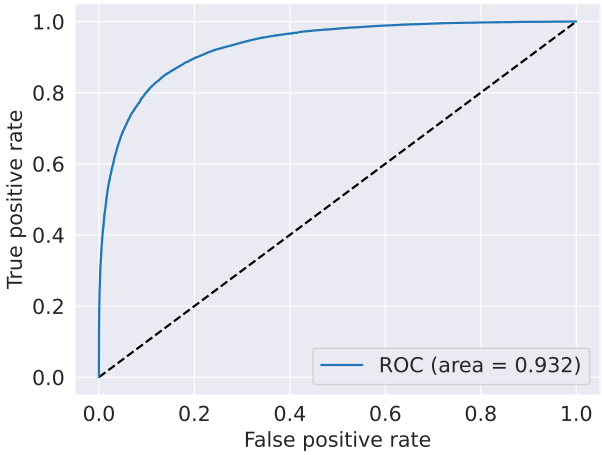
H. JetsTwo



(a) Train (blue) and test (orange) accuracy.

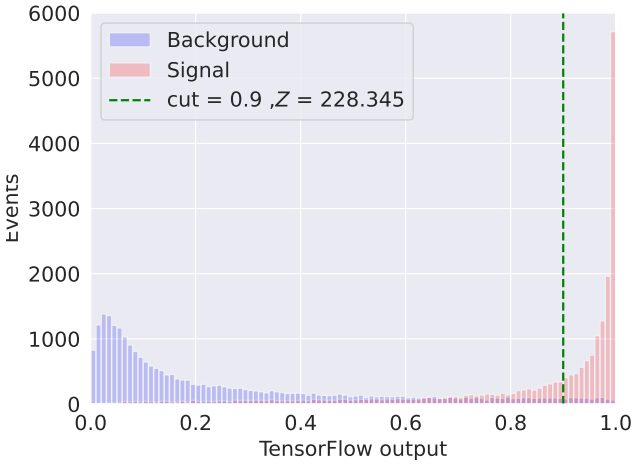


(b) Train (blue) and test (orange) loss.

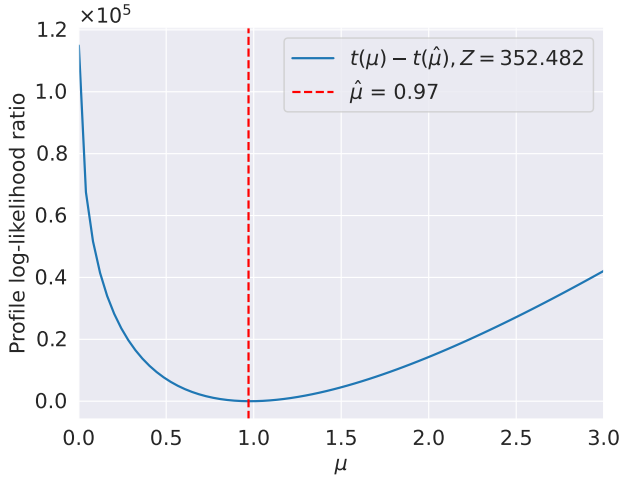


(c) ROC curve of the neural network output.

FIG. 17: The accuracy (a), loss (b) and the ROC curve (c) of the neural network output from the *JetsTwo* dataset with $\lambda = 1$ and $\eta = 0.0001$.



(a) dd



(b) dd

FIG. 18: JetsTwo

To compare and summarize the results we see that among the datasets *FillMean*, *FillZero* and *FillPhiRandom*, both *FillMean* and *FillPhiRandom* gives similar results when we look at the accuracy, loss and the AUC. In the case of *FillZero* it distingushes from the others that it learns the dataset faster and gives a lower AUC compared to the *FillMean* and the *FillPhiRandom* datasets.

When we compare the results from the two datasets *RemovePhi* and *FillMean* we observe that removing the ϕ feautures does not influence the results from the training compared with *FillMean* where we keep the ϕ feautures. This tells us that the ϕ feautures are not among the most important feautures of the Higgs dataset.

In the case of the dataset *RemoveJets* when we have removed the feautures that has missing entries (except MMHiggs) we get a lower performance of the trained model compared to *FillMean* which indicates that the neural network learns from the jet feautures in the dataset and is not influenced to much by the noise when inserting the mean in the dataset. The best result with the accuracy was for the *Jets2* dataset which had the most features among *Jets0*, *Jets1* and *Jets2*, and was also the dataset that was the most balanced with background and signal events. However it did not have the best discovery significanse since it had less statistics than e.g *FillMean*. But the likelihood estimation gave the profile likelihood function that had $\hat{\mu} = 0.97$ which was closest to the expected signal strength equal to 1 indicating that it learned the likelihood ratio better than for the other datasets.

TABLE III: Summary of the results for the different datasets

Dataset	accuracy	AUC	Z_{cut}	$Z_{likelihood}$	μ_{min}
FillMean	0.844	0.915	282.852	429.963	0.607
FillZero	0.839	0.913	290.612	483.050	0.647
FillPhiRandom	0.838	0.908	282.852	429.963	0.607
RemoveJets	0	0.908	282.852	429.963	0.526
RemovePhi	0	0.915	282.852	429.963	0.567
JetsNone	0	0.902	282.852	429.963	0.405
JetsOne	0	0.895	282.852	429.963	0.647
JetsTwo	0	0.932	282.852	429.963	0.970

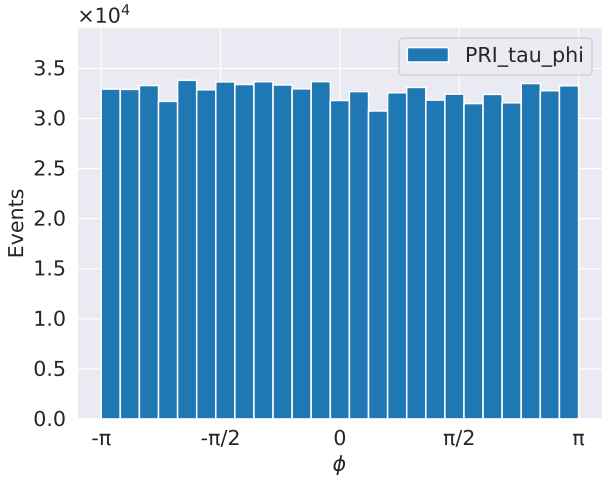
V. CONCLUSION

We have in this project tested various ways of handling the HiggsML dataset which are used to train a neural network, and the output is used to calculate the discovery significanse by using a search region and via likelihood estimation. The dataset that gave the highest discovery significance was the *FillMean* dataset which gave $Z = 429.963$ from the likelihood estimation. However, the *JetsTwo* dataset gave the highest AUC equal to 0.932, but because of less statistics it gave lower discovery significance than *FillMean*. In addition, for all the other datasets except *JetsTwo* the estimated signal strength $\hat{\mu}$ is not close to the expected signal strength equal to 1 indicating that it has not learned the likelihood ratio properly.

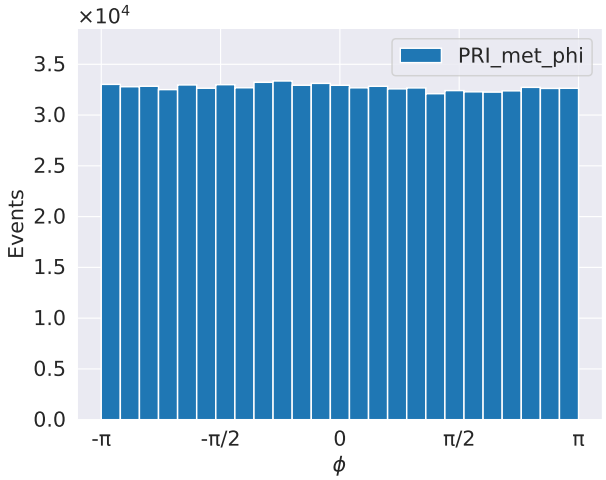
VI. APPENDIX

TABLE IV: Overview of the features in the HiggsML dataset and which of the features that has values in them for the different cases of number of jets.

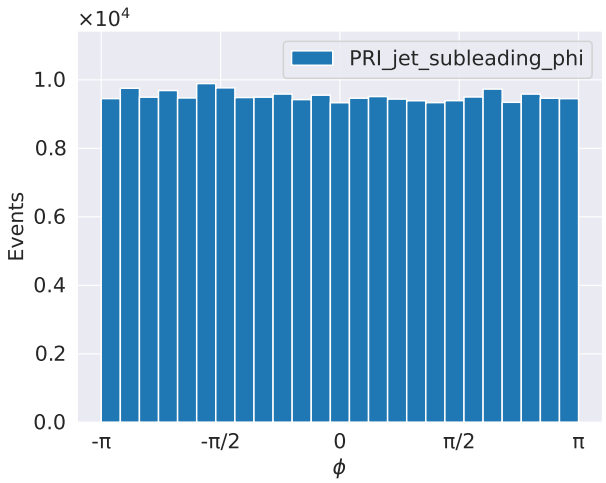
Features \ n_{jets}	0	1	2/3
EventID	✓	✓	✓
Weight	✓	✓	✓
KaggleSet	✓	✓	✓
KaggleWeight	✓	✓	✓
Label	✓	✓	✓
DER_mass_MMC	✓	✓	✓
DER_mass_transverse_met_lep	✓	✓	✓
DER_mass_vis	✓	✓	✓
DER_pt_h	✓	✓	✓
DER_deltar_tau_lep	✓	✓	✓
DER_pt_tot	✓	✓	✓
DER_sum_pt	✓	✓	✓
DER_pt_ratio_lep_tau	✓	✓	✓
DER_met_phi_centrality	✓	✓	✓
PRI_tau_pt	✓	✓	✓
PRI_tau_eta	✓	✓	✓
PRI_tau_phi	✓	✓	✓
PRI_lep_pt	✓	✓	✓
PRI_lep_eta	✓	✓	✓
PRI_lep_phi	✓	✓	✓
PRI_met	✓	✓	✓
PRI_met_phi	✓	✓	✓
PRI_met_sumet	✓	✓	✓
PRI_jet_num	✓	✓	✓
PRI_jet_all_pt	✓	✓	✓
PRI_jet_leading_pt		✓	✓
PRI_jet_leading_eta		✓	✓
PRI_jet_leading_phi		✓	✓
DER_deltaeta_jet_jet			✓
DER_mass_jet_jet			✓
DER_prodeta_jet_jet			✓
DER_lep_eta_centrality			✓
PRI_jet_subleading_pt			✓
PRI_jet_subleading_eta			✓
PRI_jet_subleading_phi			✓



(a) Caption for image 1



(c) Caption for image 3



(e) Caption for image 4

FIG. 19: Main caption for the figure.

$$
\begin{aligned}
-2\ln\lambda(\mu) &= -2\ln\left(\frac{L_{s+b}(\mu=0,\hat{\hat{\boldsymbol{\theta}}})}{L_{s+b}(\mu=1,\hat{\hat{\boldsymbol{\theta}}})}\right) \\
&= -2\ln\frac{L_{s+b}(\mu,\hat{\hat{\boldsymbol{\theta}}})}{L_{s+b}(\mu=1,\hat{\hat{\boldsymbol{\theta}}})} \\
&= -2\ln L_{s+b}(\mu,\hat{\hat{\boldsymbol{\theta}}}) - 2\ln L_{s+b}(\mu=1,\hat{\hat{\boldsymbol{\theta}}}) \\
&= -2\ln L_{s+b}(\mu,\hat{\hat{\boldsymbol{\theta}}}) - 2\ln L_{s+b}(\mu=1,\hat{\hat{\boldsymbol{\theta}}}) \\
&\quad + 2\ln L_b(\hat{\hat{\boldsymbol{\theta}}}) - 2\ln L_b(\hat{\hat{\boldsymbol{\theta}}}) \\
&= -2\ln\frac{L_{s+b}(\mu,\hat{\hat{\boldsymbol{\theta}}})}{L_b(\hat{\hat{\boldsymbol{\theta}}})} - 2\ln\frac{L_{s+b}(\mu=1,\hat{\hat{\boldsymbol{\theta}}})}{L_b(\hat{\hat{\boldsymbol{\theta}}})} \\
&= -2\ln Q(\mu) + 2\ln Q(1) \\
&= t(\mu) - t(1), \qquad (32)
\end{aligned}
$$

(Moustakides and Basioti, 2019)

REFERENCES

C. Adam-Bourdarios, G. Cowan, C. Germain, I. Guyon, B. Kégl, and D. Rousseau (2015), "The Higgs boson machine learning challenge," in *Proceedings of the NIPS 2014 Workshop on High-energy Physics and Machine Learning*, Proceedings of Machine Learning Research, Vol. 42, edited by G. Cowan, C. Germain, I. Guyon, B. Kégl, and D. Rousseau (PMLR, Montreal, Canada) pp. 19–55.

G. Cowan, K. Cranmer, E. Gross, and O. Vitells (2011), "Asymptotic formulae for likelihood-based tests of new physics," The European Physical Journal C **71** (2), 10.1140/epjc/s10052-011-1554-0.

M. Hjorth-Jensen (2023), "Week 41 constructing a neural network code, tensor flow and start convo- lutional neural networks," .

G. V. Moustakides, and K. Basioti (2019), "Training neural networks for likelihood/density ratio estimation," arXiv:1911.00405 [eess.SP].

A. L. Read (2000), "Modified frequentist analysis of search results (The CL(s) method)," in *Workshop on Confidence Limits*, pp. 81–101.