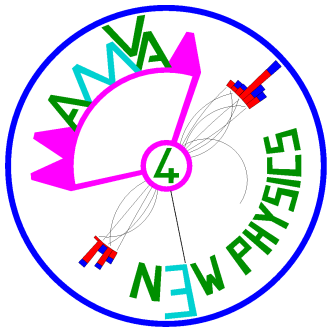
This Report is part of a project that has received funding from the **European Union's Horizon 2020 research and innovation programme under grant agreement Nº675440**



AMVA4NewPhysics ITN

Work Package 1 - Deliverable 1.4

Final Activity Report: Classification and Regression Tools in Higgs Measurements

AMVA4NewPhysics authors

October 31, 2018

Abstract

This document describes the performance of the optimized algorithms developed for application to the classification and regression problems of WP1.

Contents

1 Introduction

While machine-learning (ML) techniques saw a growing expansion since the first years of the new millennium, with an increasing range of applications to industry and academia, in experimental high-energy physics (HEP) only a sporadic use of "multivariate analysis" (MVA) approaches was initially made. The situation changed around 2012, when there was a major paradigm shift, epitomized by the use of boosted decision trees (BDTs) for four separate tasks in the landmark CMS discovery of the Higgs boson decay to photon pairs [1]. Following that success, BDTs and similar multivariate analysis tools began to be more used and accepted in HEP data analyses.

The change was slightly slower in the case of deep neural networks (DNN), which had been outperforming BDTs since around 2009 (see e.g. Ref. [2] and Ref. [3]), but whose perception as imperscrutable black boxes by the physics community made them a less agreed-upon tool for measurements and searches. This perception has however become unimportant today, mostly because the higher performance of DNNs has become harder to renounce.

In recent years, ML innovation has been the driver of solutions to the domain-specific problems in HEP, such as object reconstruction [4], collision and detector simulation [5], and particle identification [6]. Although these tasks are specific to the very special environment of particle physics, their solutions normally rely on applying and adapting techniques developed outside of HEP. Those techniques are continually being refreshed and updated, and are normally presented on benchmark datasets for some specific task, such as image recognition on ImageNet [7]. However, it is not always obvious whether improvements on such datasets would be reflected by similar success when applied in other domains. We consider this a strong motivation to study how large an improvement can be obtained on a very well-studied use case, the one of discriminating the Higgs boson signal from $H \to \tau\tau$ decays as measured by the ATLAS detector at the CERN LHC. This study is reported in Section 2.

A different specificity of HEP applications of multivariate analysis techniques concerns the task of measuring as accurately as possible a physical quantity in the presence of nuisance parameters. The latter are parameters that describe the system under measurement, or the measuring apparatus, or the technique employed in the estimation process. Nuisance parameters are not interesting per se, yet their imprecise knowledge impacts the accuracy with which the parameters of interest of a system can be estimated. The treatment of nuisance parameters in the optimization of a measurement strategy is a difficult problem, one which none of the proposed solutions have proven completely successful at handling. In Section 3 we offer a potentially groundbreaking solution to it which has become possible thanks to recent advances in the technology of DNN architectures and software.

In Section 4 we discuss in more detail the specific use case of a search for the signal due to associated production of a Higgs boson and a top-quark pair. Since the 2012 Higgs discovery this process has attracted a huge interest in the physics community, both because of its intrinsic spectacular nature (a pair of the heaviest fundamental matter particles is produced in association with the Higgs boson, giving rise to a fireworks-like decay chain) and because of its rarity and the consequent difficulty of demonstrating its existence.

While in principle the problem of spotting a small signal S in large amounts of data X ridden by backgrounds B is solved optimally by studying the likelihood ratio between the two hypotheses, L=S(x)/B(x), L is not directly accessible. ML tools can solve this problem by *generative* means -i.e. by constructing approximations to the densities S(x) and B(x), or by *discriminative* means which try to directly approximate the likelihood ratio L. The Matrix-Element Method (MEM) attempts to estimate the likelihood of an event as being due to signal or background by working out the probability of each hypothesis based on a direct calculation of the quantum-mechanical contributing processes, and incorporating the effects of

the measurement apparatus by means of a parametrization of the relative "transfer functions". This technique proves very effective in the case at hand, as shown by the illustrated results, which are based on real data acquired by the ATLAS collaboration.

1.1 Higgs physics at the Large Hadron Collider

There are four major ways of producing Higgs bosons in proton-proton collisions at the Large Hadron Collider (LHC), which is currently operating at $\sqrt{s} = 13$ TeV. They are summarized below in Figure 1 and in the following list:

- ggF: gluon-gluon fusion has the highest cross-section σ and is the most abundant process,
- VBF: vector boson fusion is characterized by two jets emitted at small angle from the proton beams
- WH, ZH: the radiation of a Higgs boson (Higgsstrahlung) from W and Z bosons is the third most common process,
- $t\bar{t}H$: Higgs production in association with top quarks is the rarest of the main Higgs production processes at the LHC

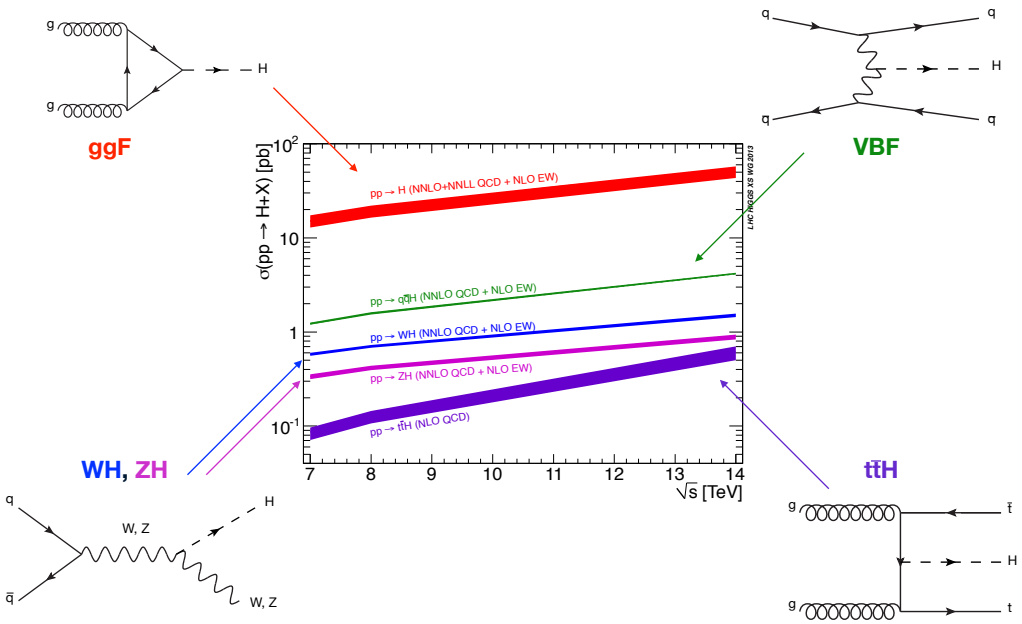


Figure 1: Dominant processes for Higgs production and the relative cross-sections in proton-proton collisions as a function of centre of mass energy. Bands indicate uncertainties in the cross-section calculation [8]

The Feynman diagram for ggF contains a loop, in which virtual particles are exchanged. Multiple particles contribute to this loop. In the Standard Model (SM), the contributions are dominated by the top quark due to its high mass, and thus this process offers a way to determine the top quark Yukawa coupling y_t. It is however plausible that yet undiscovered particles beyond the Standard Model contribute to this loop as well, thus a measurement of

y_t via ggF could be contaminated by effects originating from unknown physics beyond the Standard Model.

The $t\bar{t}H$ process on the other hand allows for a direct tree-level determination of the top quark Yukawa coupling, which makes it extremely appealing to measure despite its small cross-section. An advanced technique for a search for the $t\bar{t}H$ production process is described in section 4.

2 $h \to \tau\,\bar{\tau}$ classification

2.1 Introduction

In 2014 the Higgs ML challenge was launched on Kaggle. Participants were tasked with working on a specific kind of problem one often faces in HEP, the search for a rare signal by classification of data based on its features. The competition drew a lot of attention, with almost 2000 entries. Given the level of expertise and effort that went into the solutions, the challenge forms a viable method to benchmark models and quantitatively measure the impact of new methods and approaches.

In this section we examine the benefits and cross-domain applicability of several recent advances in deep-learning, including: a method to quickly optimise the learning rate; newer activation functions; learning rate scheduling; data augmentation; and alternative ensembling techniques. Following these experiments we present a solution which is able to achieve better performance than the winning solution under competition circumstances: an AMS increase from 3.808 to 3.818, where AMS, defined in Sec. 2.2.2, is an approximate measure of the significance of a Higgs boson signal in the data. In addition to providing a better result, the solution can be trained in less than 10 % of the time on less specialised hardware.

We begin with a more detailed description of both the general problem and the specific challenge (Sec. 2.2). An overview of neural networks is presented in Sec. 2.3. From Sec. 2.4 through Sec. 2.8 we describe the baseline model and the various improvements, reporting performances on validation data. In Sec. 2.9 we report the performance on the testing data, expanding to a fuller comparison with the winning solutions in Sec. 2.10. The investigation and the main results are summarised in Sec. 5.1.

The framework and notebook experiments are freely available at `https://github.com/GilesStrong/QCHS-2018`.

2.2 Challenge description

2.2.1 Overview

At particle colliders such as the LHC [9], sub-atomic particles or ions are accelerated to high energies and made to collide with one another. The resulting collisions are attributable to fundamental-physics interactions between the particles, and their study can be used to compare theoretical models to reality in order to better understand the nature of the universe. The energy of these collisions gives rise to particles which are recorded with specialised instruments called *particle detectors*. Since there are many possible processes which could have given rise to a particular final state, and because the detectors only capture information of the products of the collision, there is no way to tell exactly how the final state was created. However, each of the contributing processes is likely to have some specific signature which is reflected in its outgoing products, e.g. a process which gives rise to an unstable particle will have decay products which can reconstruct its mass, but due to experimental limits, such as the finite resolution of the detector, other processes can still give rise to collisions which reproduce the signature of desired process. Effectively, the data is unlabelled with respect to the class of process which gave rise to it.

When looking to test some theory, such as the presence of a new *signal* process, one normally defines some signal-enriched region, in which the process being searched for is expected to contribute appreciably. One can then examine the rate at which events populate this region and compare it to the rate expected if the new process were not occurring and the region were only being populated by known processes (background). This is done via a test of the signal+background hypothesis against the null hypothesis of background only. The catch is

that, due to the data being unlabelled, the background and signal rates are unknown, and only their sum is known. In some situations the signal signature (e.g. a particle mass with some determined resolution) is well known, and the background rate can be extrapolated from, or interpolated between, signal-depleted regions, but this is not always the case.

Instead, what can be done is to simulate particle collisions and the experimental equipment using a combination of analytical and stochastic modelling (see e.g. Ref [10]), a process referred to as Monte Carlo generation and detector simulation. Since one is able to generate collisions for specified processes, one now has a labelled approximation of the collider data and can estimate the signal and background rates in the signal-enriched region, applying correction factors if need be to improve the modelling and marginalising over any remaining unknown (nuisance) parameters in the hypothesis test.

The exact specification of the signal-enriched region is an important task which can have a strong influence on the outcome of a search; if the region is too wide the included signal becomes washed out by background, while if it is too narrow it may end up not be populated at all. The main problem is the high dimensionality of the data; due to the high energy of the collisions, each collision can result in hundreds to thousands of particles, and each particle can produce multiple 'hits' and energy deposits in the detector. Traditionally, the first step is to reduce the dimensionality by *reconstructing* the hits and deposits back into known physics objects (fundamental particles, jets, missing energy, et cetera). The next stage is to select objects from the data which correspond to the expected final states of the signal process (and cut away events which fail to produce such objects). The signal region can then be defined using some theoretically or phenomenologically motivated function of the properties of these final states and other objects in the event.

As mentioned in Sec. 1, machine learning techniques are becoming more and more used in high energy physics analyses, due to the ease with which they can discover high-dimensional patterns in data and use them to learn to solve some problem, such as learning to separate particle collisions by class. This ability, however, has limits and it is currently beyond our computational capacity to run such event-level classification algorithms directly on the detector recordings. The contemporary compromise is to still perform the particle reconstruction and event selection, but then to feed the features of the selected particles as inputs to a machine-learning based classifier and use its response to help define the signal-enriched region.

The development, training, and inference of such algorithms is still a difficult task and a source of experimentation in its own right. Launched in 2014, the "Higgs ML" challenge , hosted on Kaggle (`https://www.kaggle.com/c/higgs-boson`), was designed to help stimulate outside interest and expertise in solving such high-energy physics problems. Participants were tasked with searching for a signal process, which was the Higgs boson decaying to a pair of tau leptons, against a background comprised of several more common processes. The competition was highly successful with 1785 teams competing, and helped to introduce many new methods to HEP, as well as produce more widely used tools, such as XGBoost [11].

The data used in the challenge consist of simulated particle collisions, generated to mimic those expected at the LHC during its 2012 running. These are fed through a detailed, Geant 4 [12, 13]-based simulation of the ATLAS detector [14], and finally though the ATLAS reconstruction algorithms, resulting in a set of physics objects per simulated collision. These *events* are then filtered to select those compatible with containing the semi-leptonic decay of a pair of tau leptons, i.e. events which contain a hadronically decay tau lepton and either an electron or a muon. The properties of the reconstructed physics objects are then recorded in columnar-data format, with each row corresponding to a separate collision event, along with the process label and a weight used to normalise the contributions of the events. Both the label and the weight are only known due to the event being simulated.

The top solutions to the challenge made heavy use of ensembling techniques, combining tens

to hundreds of models. The bases of the models were mostly either decision trees/forests, or neural networks. A lot of work seemed to go into feature engineering and selection, with a new fit-based high-level feature (CAKE: `https://www.kaggle.com/c/higgs-boson/discussion/10329`) being developed towards the end of the competition. This appeared to improve the results for tree-based models, but not so much for DNN solutions, indicating that sufficiently well trained networks were able to learn their own versions of it. The other focus appeared to be optimising the way in which models were ensembled, e.g regularised greedy forests [15].

2.2.2 Scoring metric

The performance of a solution is measured using the Approximate Median Significance [16], as computed in Eq. 1:

$$\text{AMS} = \sqrt{2\left(s + b + b_r\right)\log\left(\left(1 + \frac{s}{b + b_r} - s\right)\right)}, \tag{1}$$

in which s is the sum of weights of true positive events (signal events determined by the solution to be signal), b is the sum of weights of false positive events (backgrounds events determined by the solution to be signal), and b_r is the a constant term which was set to 10 for the competition. This provides a quick, analytical value which is an approximation of the expected signal strength one would obtain after a full hypothesis test of signal+background versus background only. In High Energy Physics, an observed significance of five sigma or more, can be evidence of a new discovery.

The common practice for these kinds of problems is to cut events from the data in order to remove preferentially the background events whilst retaining the signal events, in order to improve the AMS of the remaining data. Either a single cut can be used on some highly discriminating feature of the data, or multiple cuts can be used over several features. The common machine-learning approach is to adopt the former procedure by using the features of the data to learn a new single highly discriminating feature, place a threshold on it, and then only accept events which pass the threshold. The feature learnt in this approach is simply the predicted class distribution of events, in which background events will be clustered towards zero, and signal events towards one. The AMS can then be optimised by only accepting events with a class prediction greater then some value.

The threshold cut can easily be optimised by scanning across the range of possible cuts (either at fixed steps, or checking at each data point) and picking the value which maximises the AMS. This is likely, however, to be optimal only for the dataset on which it is optimised, and performance can be expected to drop when applying the cut to unseen data. This is reflected in the challenge by requiring the solutions to predict on test data for which the class labels are not provided. It is important then that one is more careful when choosing a cut, in order to generalise better to unseen data. The approach adopted here is to maximise the cut on many bootstrap resamplings of some validation dataset, and then take the final cut as the arithmetic mean of the set of cuts.

2.2.3 Data and processing

Although the full dataset created for the challenge has now been made public [17], in order to provide an accurate comparison to the previous scores only the subset which was made available is used here, both for training and testing. The training and testing sets consist of 250 000 and 550 000 events, respectively. Both contain a mixture of both classes: signal ($h \to \tau\,\bar{\tau}$); and background ($t\,\bar{t}$, $Z \to \tau\,\bar{\tau}$, and W boson decay).

Each event is characterised by two sets of training features: primary - low level information such as tau p_T; and derived - higher-level information calculated via (non)-linear combinations of the low-level features or from hypothesis-based fitting procedures. A full list and description of the features is available is Ref [18]. The coordinate system of the data is originally provided in terms of (p_T, η, ϕ), however initial tests and past experience dictates that NN-based models perform better when vectors are mapped into a Cartesian coordinate system, due to the cyclical nature of the azimuthal angle, and the non-linear nature of the pseudorapidity, η:

$$\eta = tanh^{-1}\left(\frac{p_z}{|\bar{p}|}\right),$$

where p_z is the component of 3-momenta along the beam axis, and $|\bar{p}|$ is the magnitude of the 3-momenta.

The data also contain information about the hardest two jets in each event. For cases in which there were not enough jets reconstructed, the features are set set to default values of -999. In order to prevent these values from having an adverse effect on the normalisation and standardisation transformations which will later be applied, these values were replaced with zeros. Similarly, any infinities or missing values (indicated by NaN) which were present in the data following the coordinate transformations were also replaced with zeros.

Since scoring the testing data requires optimising a threshold on some data, and also to provide some way to compare architectures without relying on the public score of the testing data, an $80:20$ random split into training and validation sets is performed on the original training data.

After the feature processing the data consists of 31 features. The training data features are transformed to have a mean of zero, and a standard deviation of one. The exact same transformation is then applied to both the validation and testing data. The training and validation datasets are then each split into ten folds via random stratified splitting on the event class. The testing data is also split into ten folds, but via simple random splitting. This was done to move the data into a format compatible with existing tools and algorithm implementations, and to make the process of augmenting the data easier. Additionally, we henceforth redefine one epoch as being a complete use of one of these folds, and during training will load each fold sequentially.

Each event in the training and validation data also contains a *weight*. This is normalised to be used to evaluate the AMS, but also reflects the relative importance of the particular event. It is a product of the production cross-section of the underlying process and the acceptance efficiency of the initial skimming procedure. Since the model is unlikely to achieve perfect classification, it is important that it focuses on learning to classify the higher weighted events better than other, less important events. This can be achieved by applying the weights as multiplicative factors during the evaluation of the loss. This method of loss weighting can also be used to account for class imbalances in the data. Keras can automatically globally reweight to balance the classes, however its method of applying per-event weights does not account for the number of events of each class. To correctly apply the per-event weights, a copy of each event weight is created (the original weights are still needed to compute the AMS). The copied weights are then renormalised by dividing each weight in each fold by the sum of the weights of its class in its fold, i.e. in each fold, the sum of weights in each class is now one.

2.3 DNN overview

Whilst the history of these algorithms extends back to 1957, it was only in 2010 that they really started to become competitive in terms of other possible machine learning (ML) tools, such as boosted decision trees and support-vector machines, particularly in the realms of image and speech recognition; their recent renaissance being due to several changes in their design.

In essence, a NN attempts to learn a mathematical function which maps a selection of *input features* to some desired target function. This is accomplished by a series of matrix operations involving learned weights, and non-linear transformations, on the inputs. It can be visualised as layers of *neurons*, each of which receives inputs from all neurons in the previous layer. Such an illustration is shown in Fig. 2. Layers between the input and output layers are referred to as *hidden*, and when the number of hidden layers goes beyond one, the network is normally referred to as *deep* (DNN).

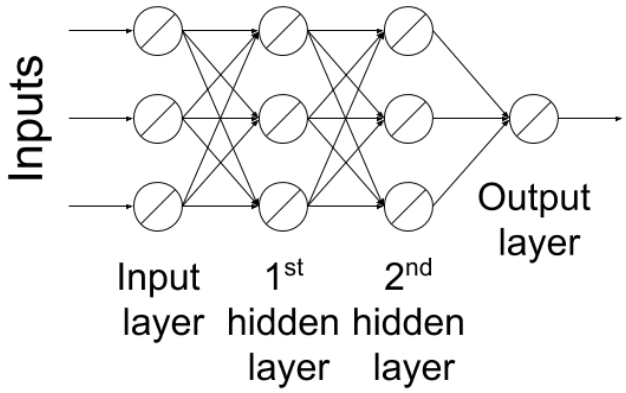


Figure 2: Conceptual illustration of a deep artificial neural network.

Two common use cases for neural networks are for regression and classification. In regression, the target function is continuous and the NN is tasked with reproducing this function by predicting its values for a given set of inputs. In classification, the target function is a set of discrete classes, and the NN attempts to map inputted data into these classes by taking advantage of the differences of the classes' PDFs when they are projected along feature axes.

Whereas other ML are built around linear methods, and rely on *high-level* features being constructed for them from non-linear combinations of the basic features (kernel method), NNs have direct access to non-linear responses, and are able to learn more easily their own high-level features. This non-linear response comes from the *activation functions* which are applied to the connections between neurons in the network.

Similar to other supervised ML algorithms, NNs are first exposed to a training dataset which is used to adjust the free parameters (the weights) in the architecture and learn the desired function. NNs are now normally trained via a process called back-propagation [19]. This is a two-stage process in which a set of training points are entered. The response of the network is then evaluated (forward propagation) and the outputs produced. These are then compared to the true target values, and a quantitative measure of the network's current performance is evaluated using a formula called the loss function. Optimisation of the network is normally arranged in terms of a minimisation problem, in which a minimum of the loss function must be found. Whilst the field of functional optimisation is fairly advanced, the aim of most state-of-the-art optimisers is to find the global minimum. For neural networks the difference in performance between the global minimum and local minima is minimal, and so it is often more efficient to reach any local minima, than to locate the global minimum. For this task, the comparatively simple *stochastic gradient decent* (SGD) algorithm is ideal.

SGD works by evaluating the gradient of the loss function at the current coordinates in parameter space (via forward propagation), and then taking steps down the gradient. By taking advantage of the fact that NNs are fully continuously differentiable functions from the output to the inputs, the effect of each free parameter in the network can be analytically evaluated by back-propagation. Each parameter can then be adjusted optimally, and after enough iterations, the response of the network should converge to that of the desired function.

There are several choices to be made when constructing NNs, and a brief overview of the choices used in the architectures developed in this note are listed below.

2.3.1 Activation function

The activation functions used in a network act to provide a (non)-linear response of each neuron based on the weighted sum of their inputs, in order to provide the neuron output:

$$y = Act\left(\bar{x} \cdot \bar{w}\right),$$

where y is the output of the neuron, equal to the activation function (Act) acting on the dot product of the inputs to the neuron ($\bar{x}$) and its weights ($\bar{w}$).

Whilst the sigmoid or hyperbolic tangent (tanh) functions were used for many decades, the default choice for the internal activation-functions in contemporary ML is now the rectified linear-unit (ReLU) [20]. This consists of a unit-gradient linear function for positive activations, and a flat line at zero for negative activation. This solves one of the problems of the sigmoid and tanh functions, which was gradient saturation, where for large absolute values of activation, the output was approximately equal to one. At saturation the gradient of the back-propagated loss-function gradient would become zero, and no training could take place. By removing gradient saturation in half of the activation region, the ReLU allows larger networks to be constructed, and more quickly trained.

The ReLU is affected by two problems: dead ReLUs, where the output of the neuron is constantly zero due to its activation always being negative; and non-zero-centred outputs, which leads to inefficient learning of the correct weights. More recent activation functions attempt to fix some of these problems, and the potential improvements of a range of functions are investigated in Sec. 2.5.

2.3.2 Weight initialisation

The weights in the network must start from some initial value; however, since uniform initialisation would lead to each neuron acting in the same way, with no possibility of learning, the weights are normally sampled randomly from a distribution. Care must be taken when specifying this distribution, such that the levels of activation within the network are not too low and not too high. Ideally, unit Gaussian inputs should remain unit Gaussian throughout the network. The Glorot/Xavier [21] initialisation was derived to ensure this for linear activation functions, and is also applicable to sigmoid and tanh functions. For ReLU-based functions, the version modified by He [22] is optimal.

2.3.3 Pre-processing

The initialisation schemes mentioned above are mathematically derived under the assumption unit Gaussian inputs. By transforming the input features such that they have zero mean and standard deviation equal to one, the weight initialisations become more optimal, and the time required for the network to converge decreases.

2.3.4 Loss function

As mentioned before, the loss function quantises the level of performance the NN currently appears to offer. Several choices can be made, however in this note we use *binary cross-entropy* (BCE) for classification tasks, and *mean square-error* (MSE) for regression.

The BCE is calculated according to:

$$\text{BCE} = -\frac{1}{N}\sum_{n=1}^{N}\left[y_n \log \hat{y}_n + (1-y_n)\log\left(1-\hat{y}_n\right)\right], \tag{2}$$

where N is the number of predictions, $\hat{y}_n$ is a vector of predictions (in our case the prediction of event class, $\hat{y} \in [0,1]$), and y_n is a vector of true values (here, the actual event class, $y \in \{0,1\}$).

2.3.5 Optimisation

Standard SGD works for training the NN, however in recent years, various additions and modifications have been made. The optimisation algorithm used in this section is called ADAM [23]. This makes use of both momentum from past updates and an accumulated store of past gradient values to continually adapt its learning rate to make larger updates when the loss function is slow to change, and smaller steps when it is rapidly changing.

2.3.6 Learning rate

According to Ref.[24], the learning rate (LR) is one of the most important parameters to set when training a neural network. Effectively, it equates to the step size the network makes over the loss surface at each update point, e.g. for standard SGD:

$$\bar{w\prime} = \bar{w} - lr \times \bar{\nabla}_w,$$

where w are the the weights of the network, and $\bar{\nabla}_w$ are the gradients of the loss with respect to the weights. If it is set too low, the network can either under-fit the data by taking too long to converge to a good enough configuration, or over-fit by finding features of the training data which are not representative of the general distribution of the data, spoiling the models performance when applied to unseen data. Conversely, if the learning rate is set too high, the network may be unable to move into loss minima due to overshooting them, and in more extreme cases can even start to diverge.

The LR range-test, introduced by Smith in Ref. [25] (and again in Ref. [26]), provides a quick way to discover the approximate value of the optimal LR. It involves starting training at a very small LR, and over the course of one epoch, increasing the LR to some high value. The loss of the network can then be examined as a function of the LR. For small rates, the loss can be expected to be flat, or slowly decreasing; the network under-fits and is unable to learn properly. At some point the LR will become large enough to allow the network to train, indicated by a falling loss. Finally the LR becomes too large, resulting in a plateau in the loss swiftly followed by a sharp rise as the network diverges. The optimal learning rate is then the highest value at which the loss is still decreasing.

Even after finding an initially optimal LR, it is unlikely that it will remain optimal throughout training; a common approach is to decrease the LR as training progresses, in order to hone in on lower loss values. Whilst this traditionally involves manually setting the LR during training, Ref. [25] and Ref. [27] both describe automated schedules for the learning rate; the former focussing improving the speed of training (and is further improved in Ref. [28], and later in Ref. [26]), and the latter trying to discover minima which generalise better to unseen data.

2.3.7 Ensembling

Ensembling is an approach in which multiple ML algorithms are combined to offer a greater performance for a larger range of inputs. Where ensembling is used in this note, the responses of the constituent NNs are weighted according to their performance as demonstrated on a separate

testing set. This allows the most performant NNs to have the greatest influence, but still be supported by other, weaker NNs in input regions where it is less performant.

2.3.8 Cross validation

For training, k-fold cross-validation is used. This involves splitting the training sample into k equally sized portions. The NN is then trained and tested k times from scratch, each time using a different portion for testing and the remaining $k-1$ for training. This provides a data efficient method of training which can have many applications such as: showing the general response of a NN, rather than how it happened to respond to a particular set of data; parameter optimisation; and training and evaluating many NNs for an ensemble.

For classifiers, the *stratified* version is preferable, which attempts to preserve the fraction of data classes in each fold. This ensures that each NN receives a balanced training, which will make its performance comparable to other train/test cycles.

2.4 Baseline model

2.4.1 Architecture

The model consists of a fully connected feed-forward neural network with 4 hidden layers, each of 100 neurons. The hidden layer neurons apply ReLU activation functions [20], with weights initialised according to He's prescription [22]. The output of the network is a single sigmoid neuron with Glorot initialisation [21]. The binary cross-entropy of predictions is minimised via stochastic gradient descent with ADAM [23] using a minibatch size of 256.

2.4.2 Learning rate optimisation

An initial test is performed to validate the use of LR range test [25]. This involves training networks for 19 epochs for three different learning rates (1×10^{-5}, 1×10^{-3}, and 1×10^{-1}), five times for each LR. We then compare the mean loss reached by the networks after training. These are detailed in Tab. 1. From these short training runs, it appears that a learning rate around 1×10^{-3} works best.

Running a LR range test from 1×10^{-5} to 1×10^{-1} results in Fig. 3. The optimum learning rate is the highest one at which the loss is still decreasing, and the results indicate that this is approximately 1×10^{-3}, which is in agreement the with results of the rough tests we initially performed. It is also worth noting that each of the initial checks took around 100 seconds to run, whereas the LR range test finished in just 20 seconds.

2.4.3 Performance

Single model 10 models are trained in 10-fold cross-validation (using the predefined folds described in Sec. 2.2.3) with a constant LR of 1×10^{-3} (as was found to be optimal in Sec. 2.4.2). Training continues until the loss on the validation fold does not decrease for 50 epochs and then

LR	Mean final loss
1×10^{-5}	$4.63 \times 10^{-5} \pm 6 \times 10^{-7}$
1×10^{-3}	$3.41 \times 10^{-5} \pm 2 \times 10^{-7}$
1×10^{-1}	$8.02 \times 10^{-4} \pm 2 \times 10^{-6}$

Table 1: Losses reached after training for 19 epochs for a range of learning rates.

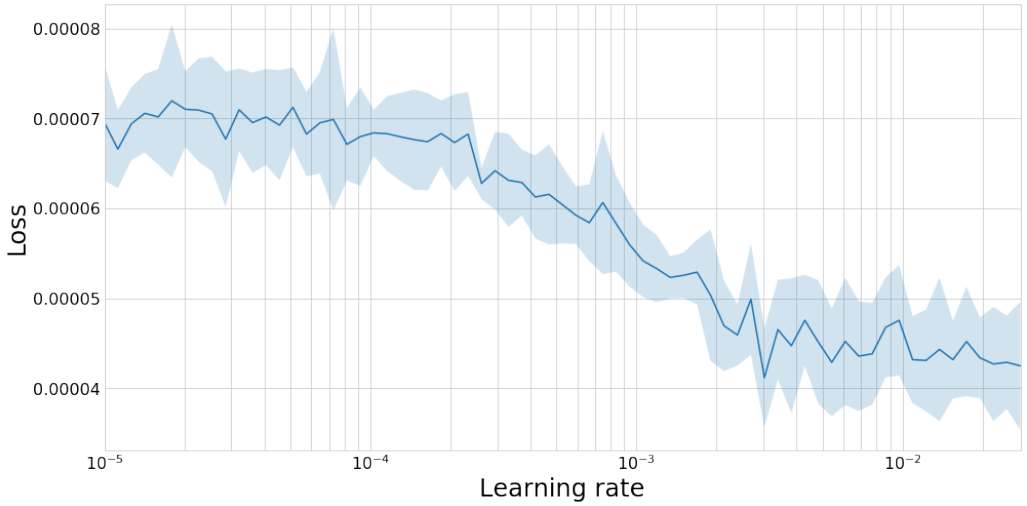


Figure 3: Loss as a function of learning rate for 10 LR range tests. The line indicates the mean loss, and the shaded region shows its standard deviation.

Metric	Single model	Ensemble	Improvement [%]
Overall maximum AMS	3.44	3.72	8.3
Mean maximal AMS	3.5 ± 0.1	3.8 ± 0.1	8.6
AMS at mean optimal cut	3.39	3.64	7.4
Inference time [s/event]	1.7×10^{-5}	1.7×10^{-4}	−890

Table 2: Validation-performance comparison between using the single best model and using a weighted ensemble of all models.

the state of lowest validation-fold loss is loaded. The mean final loss and the mean maximal AMS on the validation folds are found to be $3.29 \times 10^{-5} \pm 2 \times 10^{-7}$ and 3.49 ± 0.03, respectively. Using an `Intel i7-6500U` CPU with Keras `v2.1.6` and non-source-compiled TensorFlow `v1.8.0`, training takes 950 s.

Taking the model which reached the lowest loss on its validation fold, and applying it to the validation dataset we find the AMS to peak at 3.44. Using 512 bootstrap resamples of the validation data results in a mean maximal AMS of 3.5 ± 0.1, and an AMS corresponding to the mean optimal cut of 3.39.

Ensemble Instead, combining all 10 models in an ensemble by weighting their predictions according to the reciprocal of their final loss on their validation folds results in a large improvement in validation performance, as shown in Tab. 2.

2.5 Different activation functions

2.5.1 Function overview

As stated in Sec. 2.3, whilst ReLU is a good default choice for an activation function, it does exhibit several problems, which more recent functions attempt to address. The Parametrised ReLU (PReLU) [22] can feature a non-zero negative gradient, which is learnt via backpropagation during training. The *Scaled Exponential Linear Unit* (SELU) [29] uses careful derived scaling coefficients to keep signals in the network approximately unit Gaussian, without the

Activation function	Learning rate	Mean final loss
ReLU	1×10^{-3}	$3.41 \times 10^{-5} \pm 2 \times 10^{-7}$
PReLU	1×10^{-3}	$3.40 \times 10^{-5} \pm 2 \times 10^{-7}$
SELU	3×10^{-4}	$3.44 \times 10^{-5} \pm 2 \times 10^{-7}$
Swish	2×10^{-3}	$3.34 \times 10^{-5} \pm 1 \times 10^{-7}$

Table 3: Losses reached after training for 19 epochs for a range of activation functions. The learning rates were found via a L range-test.

need to for manual transformation via batch normalisation [30] . It is recommended to use the `lecun` initialisation scheme [31] for neurons using the SELU activation function. The Swish function [32] was found via a reinforcement leaning based search, and features the interesting characteristic of having a region of negative gradient, allowing outputs to decrease as the input increases: $\text{Swish}\,(x) = x \cdot \text{sigmoid}\,(x)$. The recommended weight initialisation scheme for Swish is the same as the one for ReLU, i.e. He. A comparison of these functions is shown in Fig. 4.

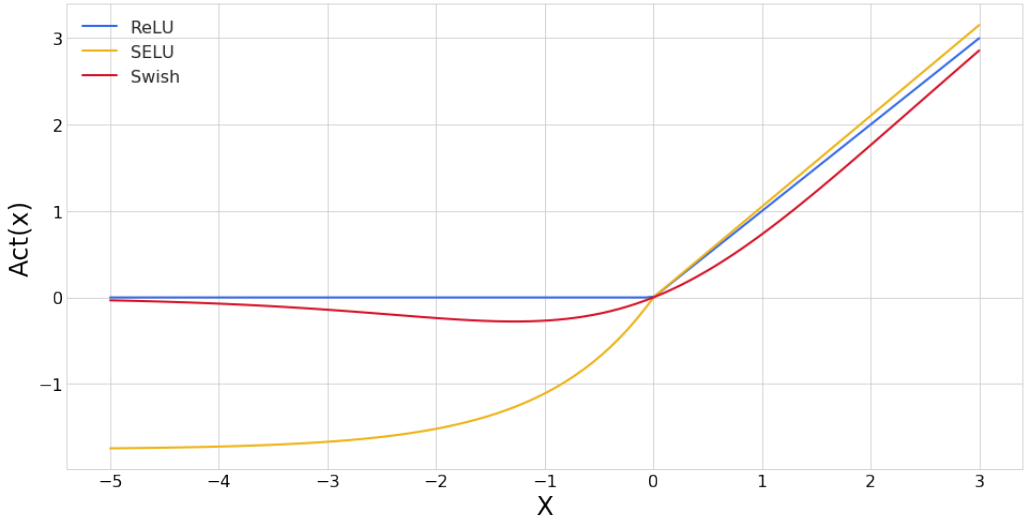


Figure 4: Responses ($Act(x)$) of the three different activation functions tested, for a range of input values, x.

2.5.2 Comparison tests

Similar to the LR comparisons performed in Sec. 2.4.2, rather than performing full trainings, we compare the network performances after only 19 epochs training and again on the average of five networks. For each activation function, the LR is optimised via a range test. The results are summarised in Tab. 3. From these results it appears as though Swish should be expected to outperform the baseline ReLU model.

2.5.3 Performance

Again we train an ensemble of 10 models, with same stopping criterion. Full training takes 1050 s and the mean final loss and mean maximal AMS on the validation folds are $3.26 \times 10^{-5} \pm 2 \times 10^{-7}$ and 3.57 ± 0.03, respectively. Comparing the Swish ensemble to the ReLU baseline in Tab. 4, we find a minor to systematic improvement over the ReLU based model.

Metric	ReLU model	Swish model	Improvement [%]
Overall maximum AMS	3.72	3.78	1.5
Mean maximal AMS	3.8 ± 0.1	3.8 ± 0.1	0
AMS at mean optimal cut	3.64	3.72	2.3
Inference time [s/event]	1.7×10^{-4}	2.0×10^{-4}	-18

Table 4: Validation-performance comparison between just Baseline ReLU and the new Swish models.

2.6 Learning-rate scheduling

2.6.1 Schedule overview

In Section 2.3 we mentioned that a common approach during network training is to adjust the learning rate in order to converge to a more optimal weight configuration. Here we explore the warm restarts approach by Loschilov and Hutter, 2016 [27], in which the LR decays according to a cosine function of the minibatch iteration, and restarts at the initial LR when the LR would become zero. Figure 5 details such a schedule, with the addition of a multiplicity factor of two, meaning that after each restart the decay rate is halved. Reference [33] suggests that discontinuities in the LR schedule allow the model to explore multiple minima across the loss surface. The cycle multiplicity factor allows the model to eventually converge in one of these minima, by increasing the time spent at small learning rates.

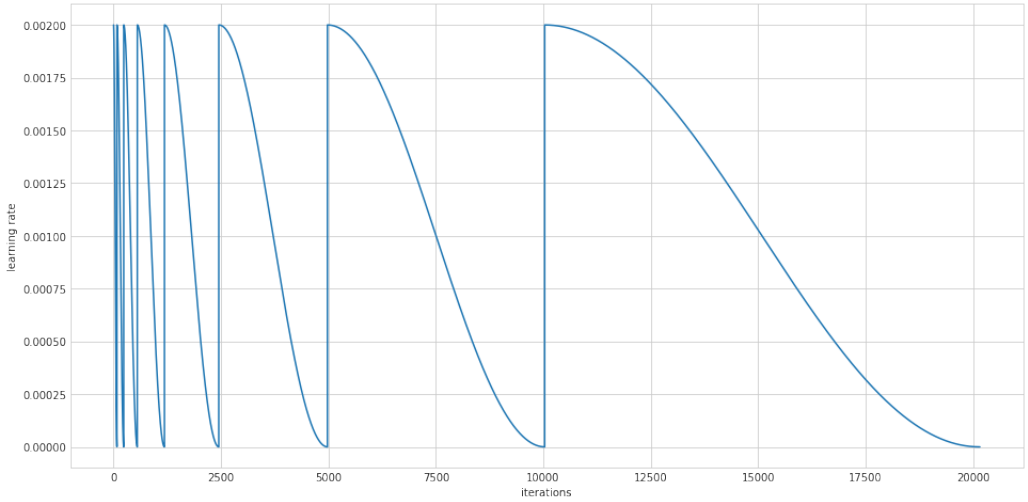


Figure 5: Example of a learning rate schedule following the cosine annealing with restarts prescription.

2.6.2 Performance

We train an ensemble of 10 Swish-based models using cosine annealed learning rates with a cycle multiplicity of two and an initial learning rate found via an LR range test (2×10^{-3}). The stopping criterion is modified such that training continues until the learning rate completes an entire cycle with no improvement in validation loss. At this point the training enters *redux decay*; the weight state with lowest validation loss is loaded, and the LR is set to the corresponding value at that state. Training then continues without cosine annealing until ten

Metric	Constant LR	Cosine annealed LR	Improvement [%]
Overall maximum AMS	3.78	3.77	−0.17
Mean maximal AMS	3.8 ± 0.1	3.8 ± 0.2	0
AMS at mean optimal cut	3.72	3.72	0.086
Inference time [s/event]	2.0×10^{-4}	2.0×10^{-4}	0

Table 5: Validation-performance comparison between the Swish ensemble with constant and annealed learning rates.

epochs elapse with no improvement in validation fold loss. After each epoch, if there has been no improvement, then the LR is set to 80 % of its current value. The rationale behind this second stage of training is that the cosine annealing may have been too rapid for the model to converge to the minima, and so the redux decay provides a second chance to converge at a slightly slower decay.

The training time doubles to around 2000 s, however the stopping criterion of one full cycle without improvement could be refined after initial tests. The mean final loss and mean maximal AMS on the validation folds are $3.23 \times 10^{-5} \pm 2 \times 10^{-7}$ and 3.71 ± 0.03, respectively. Table 5 compares the performance of the resulting ensemble to that of Sec. 2.5 with a constant learning rate. The performance on the validation data indicates equal performance for both training methods, however the mean performance on the validation folds during training shows an improvement when the cosine schedule is used. Due to this inconclusivity we will consider both schedules for the next set of improvements.

2.7 Data augmentation

2.7.1 Augmentation overview

Data augmentation is a common technique for improving the generalisation power of a model. It involves applying class-preserving transformations to the data in order to exploit or highlight certain invariances between the input and target features. In the field of image classification, these transformations could be small rotations, flips, zooms, or colour and lighting adjustments, e.g. Ref/ [34]. Application of the augmentations may be done during training in order to artificially inflate the size of the training data (train-time augmentation), or during inference by taking the mean prediction for a set of augmentations (test-time augmentation).

At particle colliders such as the LHC [9], beams of the same type of particle collide head-on with approximately zero transverse momentum. Because of this the resulting final states can be expected to be produced isotropically in both the transverse plane (x, y) and the longitudinal axis (z). Particle detectors such as CMS [35] and ATLAS [14] are built to account for these isotropies by being symmetric in both azimuthal angle (ϕ) and pseudorapidity (η). All this is to say that the class of physical process which gave rise to the collision event is only related to the relative separation between the final states, and not the absolute position of the event.

Since the data used in this problem is simulated for such a collider and detector combination, the class is invariant to flips in the transverse or longitudinal planes, and to rotations in the azimuthal angle, as illustrated in Fig. 6. Our set of class-preserving transformations therefore consists of rotating the event in ϕ, flipping the event in the z axis, and flipping the event in the x axis; allowing flips in both x and y axes, combined with a ϕ rotation, can recover the original event, and so only one transverse axis is used to avoid redundancy.

Train-time augmentation is performed by applying a random set of augmentations before each data point is used (implemented by augmenting each fold when loaded). Test-time aug-

Mode	Train time [s]	Mean final validation-fold performance	
		loss	maximal AMS
Data augmentation + cosine LR	5400	$3.18 \times 10^{-5} \pm 2 \times 10^{-7}$	3.98 ± 0.07
Data augmentation + constant LR	2000	$3.21 \times 10^{-5} \pm 2 \times 10^{-7}$	3.9 ± 0.1
Data fixing + cosine LR	1800	$3.187 \times 10^{-5} \pm 7 \times 10^{-8}$	3.9 ± 0.1
Data fixing + constant LR	990	$3.21 \times 10^{-5} \pm 2 \times 10^{-7}$	3.81 ± 0.07

Table 6: Validation-fold performance and train-time comparison between the different sets of data augmentation and fixing trainings.

mentation consists of taking the mean prediction for each data point over a set of 32 transformations: each possible combination of flips in x and z for eight different ϕ orientations.

One may well notice that the above mentioned symmetries could be removed from the data by transforming the events such that they are oriented in a predefined position. This *data fixing* approach will also be tested by rotating and flipping the events to have the light-lepton at $\phi = 0$ and in the positive η region, and the tau jet in the positive ϕ region. Since the lepton's ϕ feature is now constant, it may be removed from the input features, meaning that classifiers trained on the fixed data only use 30 inputs.

2.7.2 Performance

We train four ensembles of 10 Swish-based models: one using the cosine annealed learning rate schedule of Sec. 2.6 and data augmentation; another using the constant learning rate of Sec. 2.5 and data augmentation; another using a constant learning rate and data fixing; and a final one using using the cosine annealed learning rate schedule of Sec. 2.6 and data fixing. Table 6 details the training performance of each set; tables 7 and 8 compare the various setups on the validation data, and Tab. 9 compares the best performing setups for both of the data schemes.

Whereas Sec. 2.6 showed similar performance for both constant and scheduled learning rates, here we see that whilst data augmentation improves both, the improvement is much larger when combined with a cosine annealed LR. It is interesting to note the similar performance of the data augmentation and data fixing schemes. It can be taken to indicate that the classifier is able to learn the invariants in the data sufficiently well via the augmentation procedure but that the data fixing procedure adequately removes the symmetries. Additionally, the data fixing method is less costly to run, in terms of train and inference time.

Metric	Constant LR	Constant LR + data augmentation	Improvement [%]
Overall maximum AMS	3.78	3.88	2.9
Mean maximal AMS	3.8 ± 0.1	4.0 ± 0.2	5.3
AMS at mean optimal cut	3.72	3.80	1.9
Metric	Cosine LR	Cosine LR + data augmentation	Improvement [%]
Overall maximum AMS	3.77	3.96	5.1
Mean maximal AMS	3.8 ± 0.1	4.0 ± 0.2	5.3
AMS at mean optimal cut	3.72	3.91	5
Metric	Constant LR + data augmentation	Cosine LR + data augmentation	Improvement [%]
Overall maximum AMS	3.88	3.96	2.0
Mean maximal AMS	4.0 ± 0.2	4.0 ± 0.2	0
AMS at mean optimal cut	3.80	3.91	3.1
Inference time [s/event]	Without TTA 2.0×10^{-4}	With TTA 4.6×10^{-3}	Improvement [%] -2200

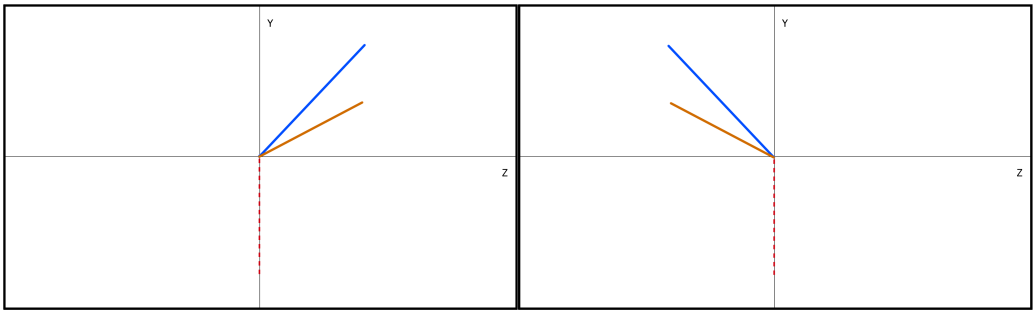
Table 7: Validation-performance comparison between the Swish ensemble with constant and annealed learning rates with and without data augmentation.

Metric	Constant LR	Constant LR + data fixing	Improvement [%]
Overall maximum AMS	3.78	3.91	3.4
Mean maximal AMS	3.8 ± 0.1	4.0 ± 0.2	5.3
AMS at mean optimal cut	3.72	3.81	2.4
Metric	Cosine LR	Cosine LR + data fixing	Improvement [%]
Overall maximum AMS	3.77	3.97	5.3
Mean maximal AMS	3.8 ± 0.1	4.0 ± 0.2	5.3
AMS at mean optimal cut	3.72	3.93	5.6
Metric	Constant LR + data fixing	Cosine LR + data fixing	Improvement [%]
Overall maximum AMS	3.91	3.97	1.5
Mean maximal AMS	4.0 ± 0.2	4.0 ± 0.2	0
AMS at mean optimal cut	3.81	3.93	3.1

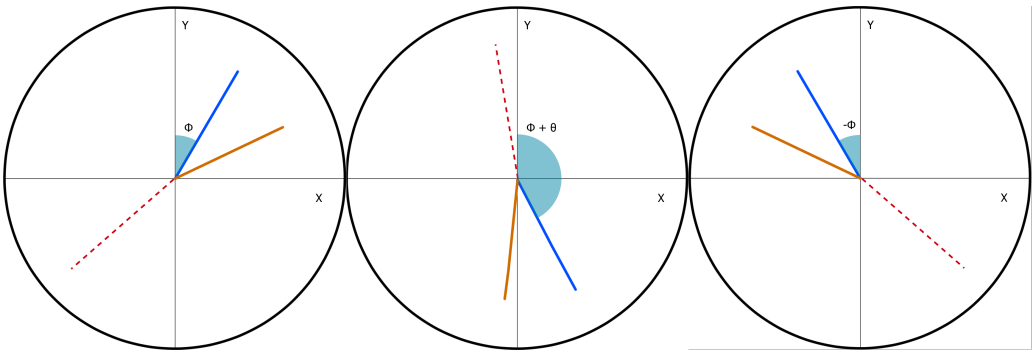
Table 8: Validation-performance comparison between the Swish ensemble with constant and annealed learning rates with and without data fixing.

Metric	Cosine LR + data fixing	Cosine LR + data augmentation	Improvement [%]
Overall maximum AMS	3.97	3.97	0
Mean maximal AMS	4.0 ± 0.2	4.0 ± 0.2	0
AMS at mean optimal cut	3.93	3.91	-0.35
Inference time [s/event]	2.7×10^{-4}	4.6×10^{-3}	-1600

Table 9: Validation-performance comparison between the data augmentation and data fixing schemes.



(a) Example event in the longitudinal-transverse plane, being flipped in the longitudinal axis. Note that the missing transverse momentum (indicated by the dashed red line) has no longitudinal component



(b) Example event in the fully transverse plane being rotated and flipped in the transverse plane.

Figure 6: Illustrations of class-preserving transformations for particle collisions.

2.8 Stochastic Weight Averaging

2.8.1 SWA overview

As mentioned in 2.6 Ref. [33] suggests that Ref. [27]'s process of restarting the LR schedule allows the network to discover multiple minima in the loss surface. The paper builds on this by introducing a method, in which an ensemble of networks results from a single training cycle by saving copies of the networks just before restarting the LR schedule, i.e. when the model is in the minima. This process of *Snapshot Ensembling* is further refined into *Fast Geometric Ensembling* (FGE) in Ref. [36], by forcing the model evolution along curves of constant loss between connected minima (simultaneously and independently discovered by Ref. [37]).

The problem with these methods is that whilst they allow one to train an ensemble in a much reduced time, one still has to run the data though each model individually, so the application-time remains the same. Reference [38] instead finds an approach which leads to an approximation of FGE in a single model. This is done by averaging the models in *weight space*, rather than in *model space*. The general method involves training a model as normal, until the model begins to converge. At this point the model continues to train as normal, but after each epoch a running average of the model's parameters is updated.

In application, one has to decide on when to begin collecting the averages: too early, and the model average is spoiled by ill-performing weights; too late, and the model does not explore the loss surface enough to allow SWA to be of use. An attempt was made to automate this decision by starting the averaging early and keeping many averages running, offset by a few epochs, and dropping those which were eventually outperformed. Although this procedure worked, it introduced additional hyper-parameters (e.g. how often to start new averages, how long to wait between comparing them), and was much slower, due to many models needing to be evaluated during training. Because of this, the implementation used here relied on training once without SWA, picking a suitable point to begin, and then rerunning allowing SWA to begin at the chosen point.

Additionally, it was found that although SWA is compatible with a cyclical LR schedule (by updating the average before each restart), for the setup used here, there were not enough restarts (average updates) to provide the potential benefit of SWA, and so a constant LR was used. Also, although SWA is meant to approximate an ensemble of models, ensembling a set of SWA models still provided improved performance.

2.8.2 Performance

We train two sets of 10 Swish-based models, both using constant learning rate. One set uses data augmentation and the other uses data fixing, and beginning SWA after 125, and 25 training epochs, respectively.

Table 10 details the training performance of each set. It is particularly interesting to compare the validation-fold loss evolution between the nominal and average weights, as shown in Fig. 7. Here one can clearly see the effect of SWA, indicated by a sudden drop in loss, followed by a flat plateau, with almost a complete suppression of the statistical fluctuations the nominal model exhibits.

Table 11 compares the various setups on the validation data, and Tab. 12 compares the best performing setup with SWA against the cosine LR with data augmentation model. We can see that SWA provides improvements for both the fixing and the augmentation schemes when a constant LR is used, however these improvements, whilst still present, decrease when a cosine LR schedule is used.

Mode	Train time [s]	Mean final validation-fold performance	
		loss	maximal AMS
SWA + augmentation + const. LR	3800	$3.18 \times 10^{-5} \pm 2 \times 10^{-7}$	4.04 ± 0.07
SWA + fixing + const. LR	2000	$3.21 \times 10^{-5} \pm 2 \times 10^{-7}$	4.1 ± 0.2

Table 10: Validation-fold performance and train-time comparison between the different sets of data augmentation and fixing trainings, with SWA.

Metric	Constant LR + data augmentation	Constant LR + data augmentation + SWA	Improvement [%]
Overall maximum AMS	3.88	3.99	2.8
Mean maximal AMS	4.0 ± 0.2	4.0 ± 0.2	0.0
AMS at mean optimal cut	3.80	3.97	4.5
Metric	Constant LR + data fixing	Constant LR + data fixing + SWA	Improvement [%]
Overall maximum AMS	3.91	3.93	0.5
Mean maximal AMS	4.0 ± 0.2	4.0 ± 0.2	0.0
AMS at mean optimal cut	3.81	3.85	1.0

Table 11: Validation-performance comparison between the Swish ensembles with constant learning rates and data augmentation/fixing with and without SWA.

Metric	Cosine LR + data augmentation	Constant LR + data augmentation + SWA	Improvement [%]
Overall maximum AMS	3.97	3.99	0.7
Mean maximal AMS	4.0 ± 0.2	4.0 ± 0.2	0
AMS at mean optimal cut	3.91	3.97	1.4

Table 12: Validation-performance comparison between the cosine LR and constant LR with SWA schemes, both using data augmentation.

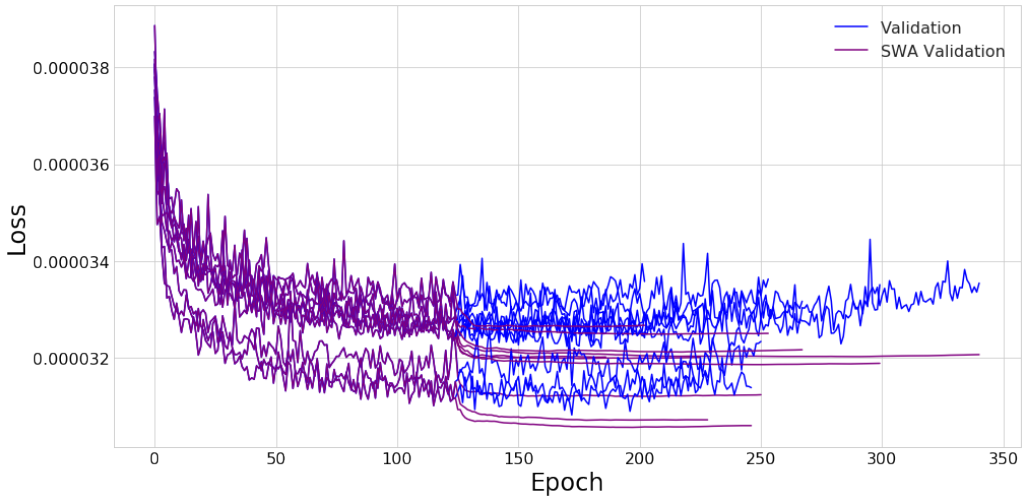


Figure 7: Validation-fold loss evolution during training for all 10 classifiers in the data augmentation scheme. SWA begins at epoch 125.

2.9 Test-set predictions

During the three-month running of the original competition on Kaggle, participants were able to submit predictions for the test data-set (which at the time was missing the true labels), and receive an AMS score on the public section of it, which corresponded to 18 % of the test data. Competitors were allowed to choose two of their entries to be run the remaining *private* test set, and their final score was the higher of the two.

With the challenge now finished, both the public and private scores are immediately available on entry submission. Additionally, the competition data were made public with all the truth labels along with flags and specific weights to allow anyone to recreate the datasets that were used in the competition, as is done for the is investigation documented in this note. Whilst the private scores are available, model comparison has so far only used a validation subset of the training data, on which the cut for the AMS is optimised.

Unfortunately, due to the small size of the training data available, the value of the AMS can fluctuate heavily. As described in Sec. 2.2.2, an attempt was made to find a generally optimal cut by bootstrapping the validation data, and finding the mean optimal cut. This was found to generally provide more optimal public scores than a single analysis of the validation data, over a variety of models.

Figure 8 summarises the AMS on both the validation and test data. "Overall Val. AMS" is the maximal AMS computed on the entire validation dataset. "Val.AMS at mean cut" is the AMS on the entire validation dataset at the mean optimal cut over 512 bootstrap resamples of the validation data. One can see that the value of "Val.AMS at mean cut" is consistently closer to the scores on the test data ("Public AMS" and "Private AMS").

The trend in the public AMS generally reflects that of the validation data, with respect to the improvements added to the model. Except in the case of the non-SWA data fixing/augmentation choice; Sec. 2.7 had shown that both choices gave approximately equal performance, however it would appear that the augmentation scheme offers greater generalisation to unseen data.

It is also interesting to note the scores of the SWA model. The version with data augmentation had shown superior performance on both the validation and public data, however the private score is slightly worse than the non-SWA data augmentation model, which slightly improves on the winning score.

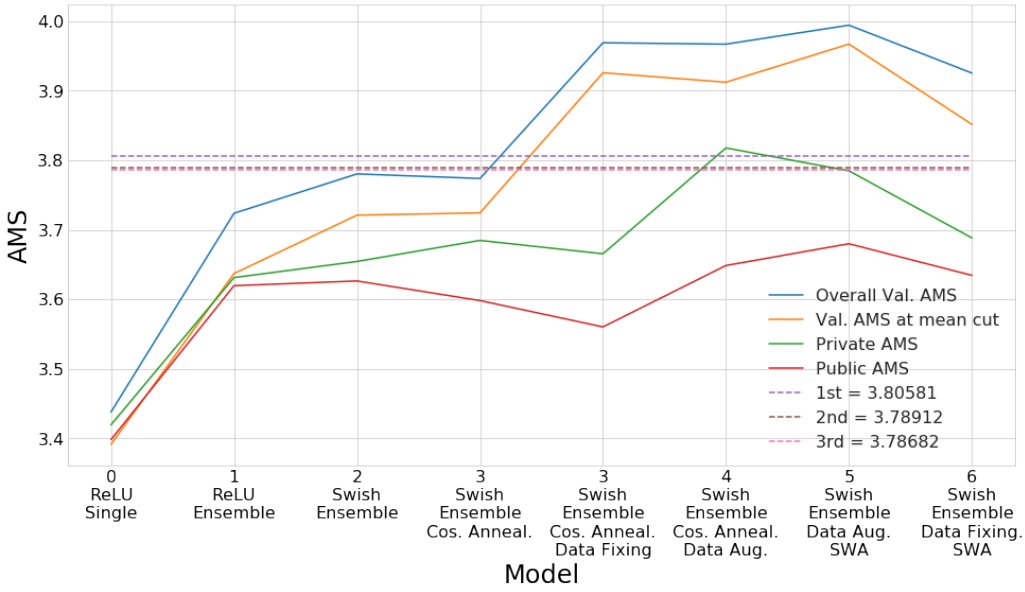


Figure 8: AMS values computed on validation and test data.

	Ours	1st	2nd	3rd
Method	10 DNNs	70 DNNs	Many BDTs	108 DNNs
Train time	1.5 h	24 h	48 h	3 h
Inference time	40 min	1 h	Unknown	20 min
Private AMS	3.818	3.806	3.789	3.787
Hardware	`Intel i7-6500U` < 8 GB RAM (2016 laptop)	`Nvidia Titan` < 24 GB RAM	>= 8-core CPU > 64 GB RAM (AWS m2.4.xlarge)	4-core CPU RAM Unknown (2012 laptop)

Table 13: Extended comparison between solutions

2.10 Solution comparison

Section 2.9 showed that the Swish ensemble using cosine annealing and data-augmentation outperformed the winning solution of the original competition. It is perhaps more useful to instead compare the solutions in terms of train and inference time, and the hardware required to get their scores. From Tab. 13 we can see that the solution developed here (labelled 'Ours') trains in less than 10 % of the winning solution's time, and requires much less specialised hardware (a system with a top-of-the-range GPU compared to a €700 laptop with a third of the RAM). Of course this comparison does not account for any advances in software or hardware that have occurred between the competition (2014) and the time of writing (2018).

Details of the top solutions may be found here: 1st - Melis (`https://github.com/melisgl/higgsml`), 2nd - Salimans (`https://github.com/TimSalimans/HiggsML`), and 3rd - Pierre (`https://www.kaggle.com/c/higgs-boson/discussion/10481`).

3 INFERNO: Inference-aware Neural Optimisation

Complex computer simulations are commonly required for accurate data modelling in many scientific disciplines, making statistical inference challenging due to the intractability of the likelihood evaluation for the observed data. Furthermore, sometimes one is interested on inference drawn over a subset of the generative model parameters while taking into account model uncertainty or mis-specification on the remaining nuisance parameters. In this work, we show how non-linear summary statistics can be constructed by minimising inference-motivated losses via stochastic gradient descent such they provided the smallest uncertainty for the parameters of interest. As a use case, the problem of confidence interval estimation for the mixture coefficient in a multi-dimensional two-component mixture model (i.e. signal vs background) is considered in this section, where we show that the proposed technique clearly outperforms summary statistics based on probabilistic classification, which are a commonly used alternative but do not account for the presence of nuisance parameters.

3.1 Background

Simulator-based inference is currently at the core of many scientific fields, such as population genetics, epidemiology, and experimental particle physics. In many cases the implicit generative procedure defined in the simulation is stochastic and/or lacks a tractable probability density $p(\boldsymbol{x}|\boldsymbol{\theta})$, where $\boldsymbol{\theta} \in \times$ is the vector of model parameters. Given some experimental observations $D = \{\boldsymbol{x}_0, ..., \boldsymbol{x}_n\}$, a problem of special relevance for these disciplines is statistical inference on a subset of model parameters $\boldsymbol{\omega} \in \otimes \subseteq \times$. This can be approached via likelihood-free inference algorithms such as Approximate Bayesian Computation (ABC) [39], simplified synthetic likelihoods [40] or density estimation-by-comparison approaches [41].

Because the relation between the parameters of the model and the data is only available via forward simulation, most likelihood-free inference algorithms tend to be computationally expensive due to the need of repeated simulations to cover the parameter space. When data are high-dimensional, likelihood-free inference can rapidly become inefficient, so low-dimensional summary statistics $\boldsymbol{s}(D)$ are used instead of the raw data for tractability. The choice of summary statistics for such cases becomes critical, given that naive choices might cause loss of relevant information and a corresponding degradation of the power of resulting statistical inference.

In the contex of data analyses at the Large Hadron Collider (LHC), the ultimate aim is to extract information about Nature from the large amounts of high-dimensional data on the subatomic particles produced by energetic collision of protons, and acquired by highly complex detectors built around the collision point. Accurate data modelling is only available via stochastic simulation of a complicated chain of physical processes, from the underlying fundamental interaction to the subsequent particle interactions with the detector elements and their readout. As a result, the density $p(\boldsymbol{x}|\boldsymbol{\theta})$ cannot be analytically computed.

The inference problem in particle physics is commonly posed as hypothesis testing based on the acquired data. An alternate hypothesis H_1 (e.g. a new theory that predicts the existence of a new fundamental particle) is tested against a null hypothesis H_0 (e.g. an existing theory, which explains previous observed phenomena). The aim is to check whether the null hypothesis can be rejected in favour of the alternate hypothesis at a certain confidence level surpassing $1 - \alpha$, where α, known as the Type I error rate, is commonly set to $\alpha = 3 \times 10^{-7}$ for discovery claims. Because α is fixed, the sensitivity of an analysis is determined by the power $1 - \beta$ of the test, where β is the probability of rejecting a false null hypothesis, also known as Type II error rate.

Due to the high dimensionality of the observed data, a low-dimensional summary statistic

has to be constructed in order to perform inference. A well-known result of classical statistics, the Neyman-Pearson lemma[42], establishes that the likelihood-ratio $\Lambda(\boldsymbol{x}) = p(\boldsymbol{x}|H_0)/p(\boldsymbol{x}|H_1)$ is the most powerful test when two simple hypotheses are considered. As $p(\boldsymbol{x}|H_0)$ and $p(\boldsymbol{x}|H_1)$ are not available, simulated samples are used in practice to obtain an approximation of the likelihood ratio by casting the problem as supervised learning classification.

In many cases, the nature of the generative model (a mixture of different processes) allows the treatment of the problem as signal (S) vs background (B) classification [43], when the task becomes one of effectively estimating an approximation of $p_S(\boldsymbol{x})/p_B(\boldsymbol{x})$ which will vary monotonically with the likelihood ratio. While the use of classifiers to learn a summary statistic can be effective and increase the discovery sensitivity, the simulations used to generate the samples which are needed to train the classifier often depend on additional uncertain parameters (commonly referred to as nuisance parameters). These nuisance parameters are not of immediate interest but have to be accounted for in order to make quantitative statements about the model parameters based on the available data. Classification-based summary statistics cannot easily account for those effects, so their inference power is degraded when nuisance parameters are finally taken into account.

In this section, we present a new machine learning method to construct non-linear sample summary statistics that directly optimises the expected amount of information about the subset of parameters of interest using simulated samples, by explicitly and directly taking into account the effect of nuisance parameters. This new method, dubbed Infernce-aware Neural Optimisation, or simply INFERNO for short, produces summary statistics can be used to build synthetic sample-based likelihoods and perform robust and efficient classical or Bayesian inference from the observed data, so they can be readily applied in place of current classification-based or domain-motivated summary statistics in current scientific data analysis workflows.

3.2 Problem Statement

Let us consider a set of n independent and identically distributed (i.i.d.) observations $D = \{\boldsymbol{x}_0, ..., \boldsymbol{x}_n\}$ where $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^d$, and a generative model which implicitly defines a probability density $p(\boldsymbol{x}|\boldsymbol{\theta})$ used to model the data. The generative model is a function of the vector of parameters $\boldsymbol{\theta} \in \times \subseteq \mathbb{R}^p$, which includes both relevant and nuisance parameters. We want to learn a function $\boldsymbol{s} : \mathcal{D} \subseteq \mathbb{R}^{d \times n} \to \mathcal{S} \subseteq \mathbb{R}^b$ that computes a summary statistic of the dataset and reduces its dimensionality so likelihood-free inference methods can be applied effectively. From here onwards, b will be used to denote the dimensionality of the summary statistic $\boldsymbol{s}(D)$.

While there might be infinite ways to construct a summary statistic $\boldsymbol{s}(D)$, we are only interested in those that are informative about the subset of interest $\boldsymbol{\omega} \in \otimes \subseteq \times$ of the model parameters. The concept of statistical sufficiency is especially useful to evaluate whether summary statistics are informative. In the absence of nuisance parameters, classical sufficiency can be characterised by means of the factorisation criterion:

$$p(D|\boldsymbol{\omega}) = h(D)g(\boldsymbol{s}(D)|\boldsymbol{\omega}) \tag{3}$$

where h and g are non-negative functions. If $p(D|\boldsymbol{\omega})$ can be factorised as indicated, the summary statistic $\boldsymbol{s}(D)$ will yield the same inference about the parameters $\boldsymbol{\omega}$ as the full set of observations D. When nuisance parameters have to be accounted in the inference procedure, alternate notions of sufficiency are commonly used such as partial or marginal sufficiency [44, 45]. Nonetheless, for the problems of relevance in this work, the probability density is not available in closed form so the general task of finding a sufficient summary statistic cannot be tackled directly. Hence, alternative methods to build summary statistics have to be followed.

For simplicity, let us consider a problem where we are only interested on statistical inference on a single one-dimensional model parameter $\boldsymbol{\omega} = \{\omega_0\}$ given some observed data. Be given

a summary statistic $\boldsymbol{s}$ and a statistical procedure to obtain an unbiased interval estimate of the parameter of interest which accounts for the effect of nuisance parameters. The resulting interval can be characterised by its width $\Delta\omega_0 = \hat{\omega}_0^+ - \hat{\omega}_0^-$, defined by some criterion so as to contain on average, upon repeated samping, a given fraction of the probability density, e.g. a central 68.3% interval. The expected size of the interval depends on the summary statistic $\boldsymbol{s}$ chosen: in general, summary statistics that are more informative about the parameters of interest will provide narrower confidence or credible intervals on their value. Under this figure of merit, the problem of choosing an optimal summary statistic can be formally expressed as finding a summary statistic $\boldsymbol{s}^*$ that minimises the interval width:

$$\boldsymbol{s}^* = \text{argmin}_{\boldsymbol{s}\in S}\Delta\omega_0 \tag{4}$$

where the minimisation $\text{argmin}_{\boldsymbol{s}\in S}$ refers to finding the summary statistic $\boldsymbol{s}^*$ within the set of possible summary statistics S, that minimises the interval width $\Delta\omega_0$. The above construction can be extended to several parameters of interest by considering the interval volume or any other function of the resulting confidence or credible regions.

3.3 Method

In this section, the INFERNO machine learning technique, that can used to learn non-linear sample summary statistics is described in detail. The method seeks to minimise the expected variance of the parameters of interest obtained via a non-parametric simulation-based synthetic likelihood. A graphical description of the technique is depicted on Fig. 9. The parameters of a neural network are optimised by stochastic gradient descent within an automatic differentiation framework, where the considered loss function accounts for the details of the statistical model as well as the expected effect of nuisance parameters.

The family of summary statistics $\boldsymbol{s}(D)$ considered in this work is composed by a neural network model applied to each dataset observation $\boldsymbol{f}(\boldsymbol{x};\boldsymbol{\phi}) : \mathcal{X} \subseteq \mathbb{R}^d \to \mathcal{Y} \subseteq \mathbb{R}^b$, whose parameters $\boldsymbol{\phi}$ will be learned during training by means of stochastic gradient descent, as will be discussed later. Therefore, using set-builder notation the family of summary statistics considered can be denoted as:

$$\boldsymbol{s}(D,\boldsymbol{\phi}) = \boldsymbol{s}\left(\left\{\, \boldsymbol{f}(\boldsymbol{x}_i;\boldsymbol{\phi}) \mid \forall\, \boldsymbol{x}_i \in D \,\right\}\right) \tag{5}$$

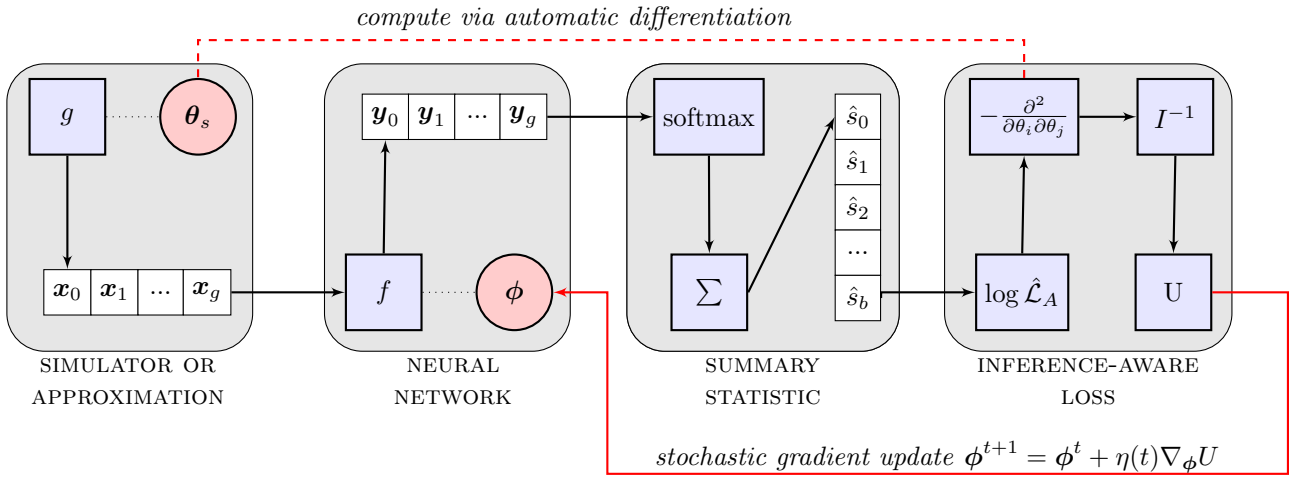


Figure 9: Learning inference-aware summary statistics (see text for details).

where $\boldsymbol{f}(\boldsymbol{x}_i;\boldsymbol{\phi})$ will reduce the dimensionality from the input observations space $\mathcal{X}$ to a lower-dimensional space $\mathcal{Y}$. The next step is to map observation outputs to a dataset summary statistic, which will in turn be calibrated and optimised via a non-parametric likelihood $\mathcal{L}(D;\boldsymbol{\theta},\boldsymbol{\phi})$ created using a set of simulated observations $G_s = \{\boldsymbol{x}_0, ..., \boldsymbol{x}_g\}$, generated at a certain instantiation of the simulator parameters $\boldsymbol{\theta}_s$.

In experimental high energy physics experiments, histograms of observation counts are the most commonly used non-parametric density estimator because the resulting likelihoods can be expressed as the product of Poisson factors, one for each of the considered bins. A naive sample summary statistic can be built from the output of the neural network by simply assigning each observation $\boldsymbol{x}$ to a bin corresponding to the cardinality of the maximum element of $\boldsymbol{f}(\boldsymbol{x};\boldsymbol{\phi})$, so each element of the sample summary will correspond to the following sum:

$$s_i(D;\boldsymbol{\phi}) = \sum_{\boldsymbol{x}\in D} \begin{cases} 1 & i = argmax_{j=\{0,...,b\}}(f_j(\boldsymbol{x};\boldsymbol{\phi})) \\ 0 & i \neq argmax_{j=\{0,...,b\}}(f_j(\boldsymbol{x};\boldsymbol{\phi})) \end{cases} \tag{6}$$

which can in turn be used to build the following likelihood, where the expectation for each bin is taken from the simulated sample G_s:

$$\mathcal{L}(D;\boldsymbol{\theta},\boldsymbol{\phi}) = \prod_{i=0}^{b} \text{Pois}\left(s_i(D;\boldsymbol{\phi}) \mid \left(\frac{n}{g}\right) s_i(G_s;\boldsymbol{\phi})\right) \tag{7}$$

where the n/g factor accounts for the different number of observations in the simulated samples. In cases where the number of observations is itself a random variable providing information about the parameters of interest, or where the simulated observation are weighted, the choice of normalisation of $\mathcal{L}$ may be slightly more involved and problem specific, but nevertheless amenable.

In the above construction, the chosen family of summary statistics is non-differentiable due to the $argmax$ operator, so gradient-based updates for the parameters cannot be computed. To work around this problem, a differentiable approximation $\hat{\boldsymbol{s}}(D;\boldsymbol{\phi})$ is considered. This function is defined by means of a $softmax$ operator, which is a generalisation of the logistic function returns a transformation so all operands add up to on, thus can be used to represent a categorical probability distribution:

$$\hat{s}_i(D;\boldsymbol{\phi}) = \sum_{x\in D} \frac{e^{f_i(\boldsymbol{x};\boldsymbol{\phi})/\tau}}{\sum_{j=0}^{b} e^{f_j(\boldsymbol{x};\boldsymbol{\phi})/\tau}} \tag{8}$$

where the temperature hyper-parameter τ will regulate the softness of the operator. In the limit of $\tau \to 0^+$, the probability of the largest component will tend to 1 while others to 0, and therefore $\hat{\boldsymbol{s}}(D;\boldsymbol{\phi}) \to \boldsymbol{s}(D;\boldsymbol{\phi})$. Similarly, let us denote by $\hat{\mathcal{L}}(D;\boldsymbol{\theta},\boldsymbol{\phi})$ the differentiable approximation of the non-parametric likelihood obtained by substituting $\boldsymbol{s}(D;\boldsymbol{\phi})$ with $\hat{\boldsymbol{s}}(D;\boldsymbol{\phi})$. Instead of using the observed data D, the value of $\hat{\mathcal{L}}$ may be computed when the observation for each bin is equal to its corresponding expectation based on the simulated sample G_s, which is commonly denoted as the Asimov likelihood [46] $\hat{\mathcal{L}}_A$:

$$\hat{\mathcal{L}}_A(\boldsymbol{\theta};\boldsymbol{\phi}) = \prod_{i=0}^{b} \text{Pois}\left(\left(\frac{n}{g}\right) \hat{s}_i(G_s;\boldsymbol{\phi}) \mid \left(\frac{n}{g}\right) \hat{s}_i(G_s;\boldsymbol{\phi})\right) \tag{9}$$

for which it can be easily proven that $argmax_{\boldsymbol{\theta}\in\theta}(\hat{\mathcal{L}}_A(\boldsymbol{\theta};\boldsymbol{\phi})) = \boldsymbol{\theta}_s$, i.e. the parameter vector $\boldsymbol{\theta} \in \theta$ that minimises the likelihood is the one used for generating the data. Therefore, the maximum likelihood estimator (MLE) for the Asimov likelihood is the parameter vector $\boldsymbol{\theta}_s$ used to generate the simulated dataset G_s. In Bayesian terms, if the prior over the parameters is flat

in the chosen metric, then $\boldsymbol{\theta}_s$ is also the maximum a posteriori (MAP) estimator. By taking the negative logarithm and expanding in $\boldsymbol{\theta}$ around $\boldsymbol{\theta}_s$, we can obtain the Fisher information matrix [47] for the Asimov likelihood:

$$\boldsymbol{I}(\boldsymbol{\theta})_{ij} = \frac{\partial^2}{\partial\theta_i \partial\theta_j} \Big(-\log \hat{\mathcal{L}}_A(\boldsymbol{\theta}; \boldsymbol{\phi}) \Big) \tag{10}$$

which can be computed via automatic differentiation if the simulation is differentiable and included in the computation graph or if the effect of varying $\boldsymbol{\theta}$ over the simulated dataset G_s can be effectively approximated. While this requirement does constrain the applicability of the proposed technique to a subset of likelihood-free inference problems, it is quite common for e.g. physical sciences that the effect of the parameters of interest and the main nuisance parameters over a sample can be approximated by the changes of mixture coefficients of mixture models, translations of a subset of features, or conditional density ratio re-weighting.

If $\hat{\boldsymbol{\theta}}$ is an unbiased estimator of the values of $\boldsymbol{\theta}$, the covariance matrix fulfils the Cramér-Rao lower bound [48, 49]:

$$\mathrm{cov}_{\boldsymbol{\theta}}(\hat{\boldsymbol{\theta}}) \geq I(\boldsymbol{\theta})^{-1} \tag{11}$$

and the inverse of the Fisher information can be used as an approximate estimator of the expected variance, given that the bound would become an equality in the asymptotic limit for MLE. If some of the parameters $\boldsymbol{\theta}$ are constrained by independent measurements characterised by their likelihoods $\{\mathcal{L}^0_C(\boldsymbol{\theta}), ..., \mathcal{L}^c_C(\boldsymbol{\theta})\}$, those constraints can also be easily included in the covariance estimation, simply by considering the augmented likelihood $\hat{\mathcal{L}}'_A$ instead of $\hat{\mathcal{L}}_A$ in Eq. 10:

$$\hat{\mathcal{L}}'_A(\boldsymbol{\theta}; \boldsymbol{\phi}) = \hat{\mathcal{L}}_A(\boldsymbol{\theta}; \boldsymbol{\phi}) \prod_{i=0}^{c} \mathcal{L}^i_C(\boldsymbol{\theta}). \tag{12}$$

In Bayesian terminology, this approach is referred to as the Laplace approximation [50] where the logarithm of the joint density (including the priors) is expanded around the MAP to a multi-dimensional normal approximation of the posterior density:

$$p(\boldsymbol{\theta}|D) \approx \mathrm{Normal}(\boldsymbol{\theta}; \hat{\boldsymbol{\theta}}, I(\hat{\boldsymbol{\theta}})^{-1}) \tag{13}$$

which has already been approached by automatic differentiation in probabilistic programming frameworks [51]. While a histogram has been used to construct a Poisson count sample likelihood, non-parametric density estimation techniques can be used in its place to construct a product of observation likelihoods based on the neural network output $\boldsymbol{f}(\boldsymbol{x}; \boldsymbol{\phi})$ instead. For example, an extension of this technique to use kernel density estimation (KDE) should be straightforward, given its intrinsic differentiability.

The loss function used for stochastic optimisation of the neural network parameters $\boldsymbol{\phi}$ can be any function of the inverse of the Fisher information matrix at $\boldsymbol{\theta}_s$, depending on the ultimate inference aim. The diagonal elements $I^{-1}_{ii}(\boldsymbol{\theta}_s)$ correspond to the expected variance of each of the ϕ_i under the normal approximation mentioned before, so if the aim is efficient inference about one of the parameters $\omega_0 = \theta_k$ a candidate loss function is:

$$U = I^{-1}_{kk}(\boldsymbol{\theta}_s) \tag{14}$$

which corresponds to the expected width of the confidence interval for ω_0 accounting also for the effect of the other nuisance parameters in $\boldsymbol{\theta}$. This approach can also be extended when the goal is inference over several parameters of interest $\boldsymbol{\omega} \subseteq \boldsymbol{\theta}$ (e.g. when considering a weighted sum of the relevant variances). A simple version of the approach described above to learn a

neural-network based summary statistic employing an inference-aware loss is summarised in Algorithm 1.

Algorithm 1 Inference-Aware Neural Optimisation (INFERNO).

Input 1: differentiable simulator or variational approximation $g(\boldsymbol{\theta})$.
Input 2: initial parameter values $\boldsymbol{\theta}_s$.
Input 3: parameter of interest $\omega_0 = \theta_k$.
Output: learned summary statistic $\boldsymbol{s}(D; \boldsymbol{\phi})$.

1: **for** $i = 1$ to N **do**
2: Sample a representative mini-batch G_s from $g(\boldsymbol{\theta}_s)$.
3: Compute differentiable summary statistic $\hat{\boldsymbol{s}}(G_s; \boldsymbol{\phi})$.
4: Construct Asimov likelihood $\mathcal{L}_A(\boldsymbol{\theta}, \boldsymbol{\phi})$.
5: Get information matrix inverse $I(\boldsymbol{\theta})^{-1} = \boldsymbol{H}_{\boldsymbol{\theta}}^{-1}(\log \mathcal{L}_A(\boldsymbol{\theta}, \boldsymbol{\phi}))$.
6: Obtain loss $U = I_{kk}^{-1}(\boldsymbol{\theta}_s)$.
7: Update network parameters $\boldsymbol{\phi} \rightarrow \mathrm{SGD}(\nabla_{\boldsymbol{\phi}} U)$.
8: **end for**

3.4 Related Work

Classification or regression models have been implicitly used to construct summary statistics for inference in several scientific disciplines. For example, in experimental particle physics, the mixture model structure of the problem makes it amenable to supervised classification based on simulated datasets [52, 53]. While a classification objective can be used to learn powerful feature representations and increase the sensitivity of an analysis, it does not take into account the details of the inference procedure or the effect of nuisance parameters like the solution proposed in this work.

The first known effort to include the effect of nuisance parameters in classification and explain the relation between classification and the likelihood ratio was by Neal [54]. In the mentioned work, Neal proposes training of classifier including a function of nuisance parameter as additional input together with a per-observation regression model of the expectation value for inference. Cranmer et al. [41] improved on this concept by using a parametrised classifier to approximate the likelihood ratio which is then calibrated to perform statistical inference. At variance with the mentioned works, we do not consider a classification objective at all and the neural network is directly optimised based on an inference-aware loss. Additionally, once the summary statistic has been learnt the likelihood can be trivially constructed and used for classical or Bayesian inference without a dedicated calibration step. Furthermore, the approach presented in this work can also be extended, as done by Baldi et al. [55], by considering a subset of the inference parameters to obtain a parametrised family of summary statistics with a single model.

Recently, Brehmer et al. [56, 57, 58] further extended the approach of parametrised classifiers to better exploit the latent-space space structure of generative models from complex scientific simulators. Additionally they propose a family of approaches that include a direct regression of the likelihood ratio and/or likelihood score in the training losses. While extremely promising, the most performing solutions are designed for a subset of the inference problems at the LHC and they require considerable changes in the way the inference is carried out. The aim of this work is different, as we try to learn sample summary statistics that may act as a plug-in replacement of classifier-based dimensionality reduction and can be applied to general likelihood-free problems where the effect of the parameters can be modelled or approximated.

Within the field of Approximate Bayesian Computation (ABC), there have been some attempts to use neural network as a dimensionality reduction step to generate summary statistics. For example, Jiang et al. [59] successfully employ a summary statistic by directly regressing the parameters of interest and therefore approximating the posterior mean given the data, which then can be used directly as a summary statistic.

A different path is taken by Louppe et al. [60], where the authors present a adversarial training procedure to enforce a pivotal property on a predictive model. The main concern of this approach is that a classifier which is pivotal with respect to nuisance parameters might not be optimal, neither for classification nor for statistical inference. Instead of aiming for being pivotal, the summary statistics learnt by our algorithm attempt to find a transformation that directly reduces the expected effect of nuisance parameters over the parameters of interest.

3.5 Experiments

In this section, we first study the effectiveness of the INFERNO technique in a synthetic mixture problem where the likelihood is known. We then compare our results with those obtained by standard classification-based summary statistics. All the code needed to reproduce the results presented the results presented here is available in an online repository [61], extensively using TensorFlow [62] and TensorFlow Probability [51, 63] software libraries.

3.5.1 3D Synthetic Mixture

In order to exemplify the usage of the proposed approach, evaluate its viability and test its performance by comparing to the use of a classification model proxy, a three-dimensional mixture example with two components is considered. One component will be referred to as background $f_b(\boldsymbol{x}|\lambda)$ and the other as signal $f_s(\boldsymbol{x})$; their probability density functions are taken to correspond respectively to:

$$f_b(\boldsymbol{x}|r,\lambda) = \mathcal{N}\left((x_0,x_1)\,\middle|\,(2+r,0),\begin{bmatrix}5 & 0\\ 0 & 9\end{bmatrix}\right) Exp(x_2|\lambda) \tag{15}$$

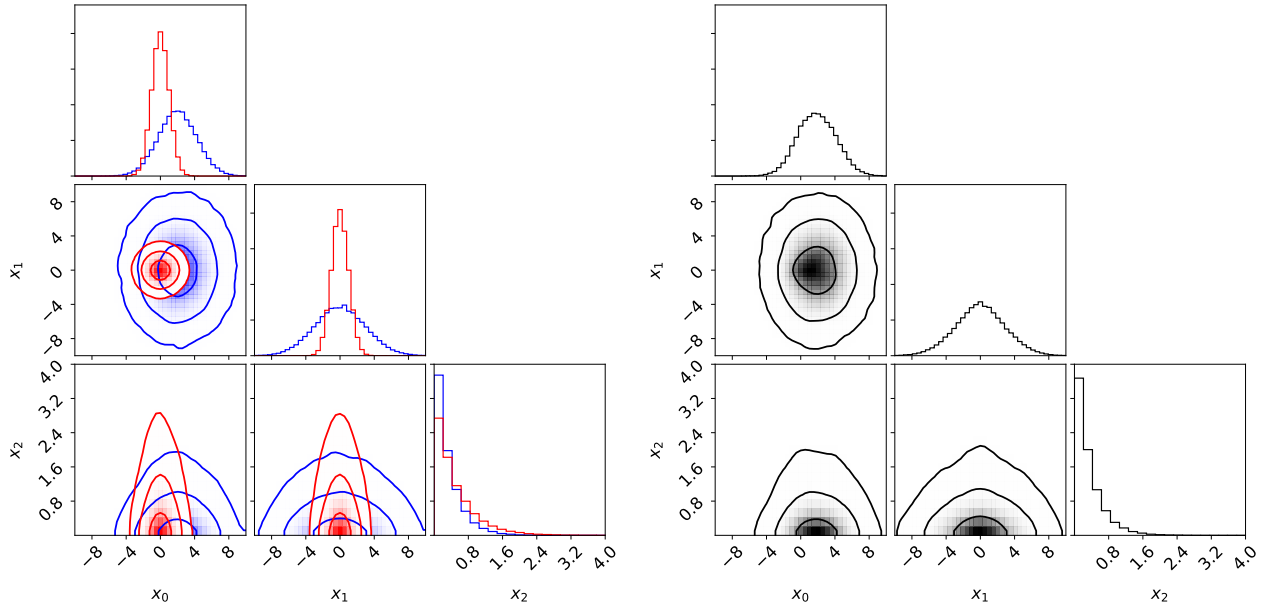
$$f_s(\boldsymbol{x}) = \mathcal{N}\left((x_0,x_1)\,\middle|\,(1,1),\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\right) Exp(x_2|2) \tag{16}$$

so that (x_0, x_1) are distributed according to a multivariate normal distribution while x_2 follows an independent exponential distribution both for background and signal, as shown in Fig. **??**. The signal distribution is fully specified while the background distribution depends on r, a parameter which shifts the mean of the background density, and a parameter λ which specifies the exponential rate in the third dimension x_2 of the background distribution. These parameters will be the treated as nuisance parameters when benchmarking different methods. Hence, the probability density function of observations has the following mixture structure:

$$p(\boldsymbol{x}|\mu,r,\lambda) = (1-\mu)f_b(\boldsymbol{x}|r,\lambda) + \mu f_s(\boldsymbol{x}) \tag{17}$$

where μ is the parameter corresponding to the mixture weight for the signal and consequently $(1-\mu)$ is the mixture weight for the background. The low-dimensional projections from samples from the mixture distribution for a small $\mu = 50/1050$ is shown in Fig. **??**.

Let us assume that we want to carry out inference based on n i.i.d. observations, such that $\mathbb{E}[n_s] = \mu n$ observations of signal and $\mathbb{E}[n_b] = (1-\mu)n$ observations of background are expected, respectively. While the mixture model parametrisation shown in Eq. 17 is correct as it is, the underlying model could also give information on the expected number of observations as a function of the model parameters. In this toy problem, we consider a case where the



(a) signal (red) and background (blue) *(b) mixture distribution (black)*

Figure 10: Projection in 1D and 2D dimensions of 50000 samples from the synthetic problem considered. Each mixture component is shown separately (left, a) while samples from the mixture are also shown (right, b). The background distribution nuisance parameters used for generating data correspond to $r = 0$ *and* $\lambda = 3$*. For samples of the mixture distribution, the coefficients* $s = 50$ *and* $b = 1000$ *were used, hence the mixture coefficient is* $\mu = 50/1050$*.*

underlying model predicts that the total number of observations are Poisson-distributed with a mean $s+b$, where s and b are the expected number of signal and background observations. Thus the following parametrisation will be more convenient for building sample-based likelihoods:

$$p(\boldsymbol{x}|s, r, \lambda, b) = \frac{b}{s+b} f_b(\boldsymbol{x}|r, \lambda) + \frac{s}{s+b} f_s(\boldsymbol{x}). \tag{18}$$

This parametrisation is common for physics analyses at the LHC, because theoretical calculations provide information about the expected number of observations. If the probability density is known, but the expectation for the number of observed events depends on the model parameters, the likelihood can be extended [64] with a Poisson count term as:

$$\mathcal{L}(s, r, \lambda, b) = \text{Pois}(n|s+b) \prod^{n} p(\boldsymbol{x}|s, r, \lambda, b) \tag{19}$$

which will be used to provide an optimal inference baseline when benchmarking the different approaches. Another quantity of relevance is the conditional density ratio, which would correspond to the optimal classifier (in the Bayes risk sense) separating signal and background events in a balanced dataset (equal priors):

$$s^*(\boldsymbol{x}|r, \lambda) = \frac{f_s(\boldsymbol{x})}{f_s(\boldsymbol{x}) + f_b(\boldsymbol{x}|r, \lambda)} \tag{20}$$

noting that this quantity depends on the parameters that define the background distribution r and λ, but not on s or b that are a function of the mixture coefficients. It can be proven (see Appendix A) that $s^*(\boldsymbol{x})$ is a sufficient summary statistic with respect to an arbitrary two-component mixture model if the only unknown parameter is the signal mixture fraction μ (or alternatively s in the chosen parametrisation). In practise, the probability density functions

of signal and background are not known analytically, and only forward samples are available through simulation, so alternative approaches are required.

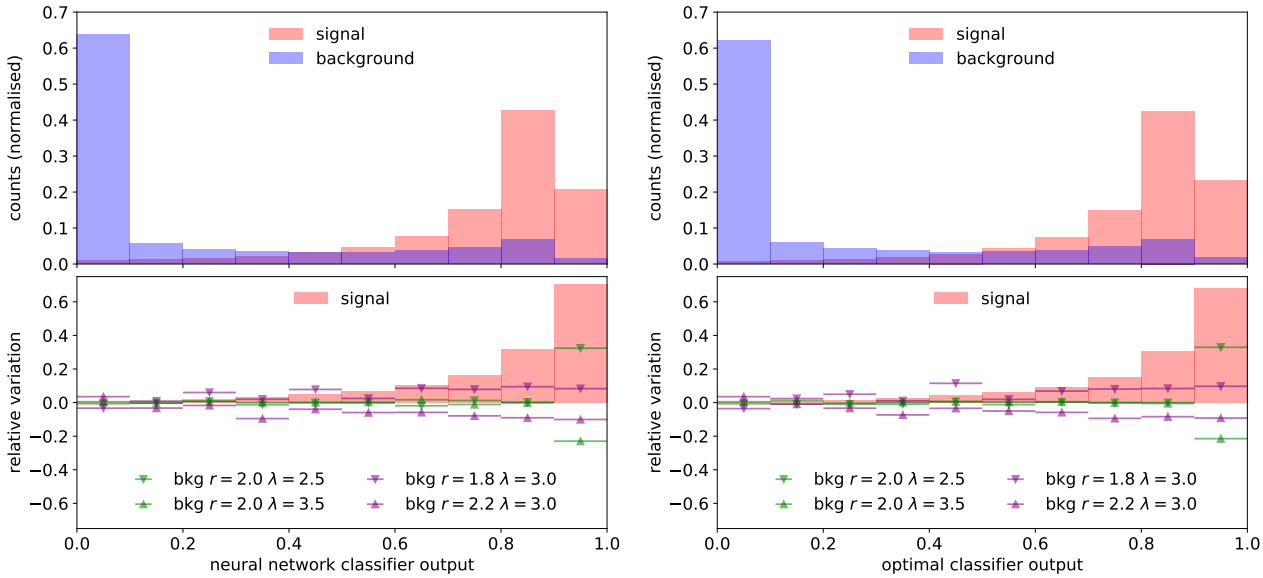
While the synthetic nature of this example allows to rapidly generate training data on demand, a training dataset of 200,000 simulated observations has been considered, in order to study how the proposed method performs when training data are limited. Half of the simulated observations correspond to the signal component and half to the background component. The latter has been generated using $r = 0.0$ and $\lambda = 3.0$. A validation holdout from the training dataset of 200,000 observations is only used for computing relevant metrics during training and to control over-fitting. The final figures of merit that allow to compare different approaches are computed using a larger dataset of 1,000,000 observations. For simplicity, mini-batches for each training step are balanced so the same number of events from each component is taken both when using standard classification or inference-aware losses.

An option is to pose the problem as one of classification based on a simulated dataset. A supervised machine learning model such a neural network can be trained to discriminate signal and background observations, considering a fixed parameters r and λ. The output of such a model typically consist in class probabilities c_s and c_b given an observation $\boldsymbol{x}$, which will tend asymptotically to the optimal classifier from Eq. 20 given enough data, a flexible enough model and a powerful learning rule. The conditional class probabilities (or alternatively the likelihood ratio $f_s(\boldsymbol{x})/f_b(\boldsymbol{x})$) are powerful learned features that can be used as summary statistic; however their construction ignores the effect of the nuisance parameters r and λ on the background distribution. Furthermore, some kind of non-parametric density estimation (e.g. a histogram) has to be considered in order to build a calibrated statistical model using the classification-based learned features, which will in turn smooth and reduce the information available for inference.

To exemplify the use of this family of classification-based summary statistics, a histogram of a deep neural network classifier output trained on simulated data and its variation computed for different values of r and λ are shown in Fig. 11a. The details of the training procedure will be provided later in this document. The classifier output can be directly compared with $s(\boldsymbol{x}|r = 0.0, \lambda = 3.0)$ evaluated using the analytical distribution function of signal and background according to Eq. 20, which is shown in Fig. 11b and corresponds to the optimal classifier. The trained classifier approximates very well the optimal classifier. The summary statistic distribution for background depends considerably on the value of the nuisance parameters both for the trained and the optimal classifier, which will in turn cause an important degradation on the subsequent statistical inference.

The statistical model described above has up to four unknown parameters: the expected number of signal observations s, the background mean shift r, the background exponential rate in the third dimension λ, and the expected number of background observations. The effect of the expected number of signal and background observations s and b can be easily included in the computation graph by weighting the signal and background observations. This is equivalent to scaling the resulting vector of Poisson counts (or its differentiable approximation) if a non-parametric counting model as the one described in Sec. 3.3 is used. Instead the effect of r and λ, both nuisance parameters that will define the background distribution, is more easily modelled as a transformation of the input data $\boldsymbol{x}$. In particular, r is a nuisance parameter that causes a shift on the background along the first dimension and its effect can accounted for in the computation graph by simply adding $(r, 0.0, 0.0)$ to each observation in the mini-batch generated from the background distribution. Similarly, the effect of λ can be modelled by multiplying x_2 by the ratio between the λ_0 used for generation and the one being modelled. These transformations are specific for this example, but alternative transformations depending on parameters could also be accounted for as long as they are differentiable or substituted by a differentiable approximation.

For this problem, we are interested in carrying out statistical inference on the parameter of



(a) classifier trained on simulated samples *(b) optimal classifier* $s(\boldsymbol{x}|r = 0.0, \lambda = 3.0)$

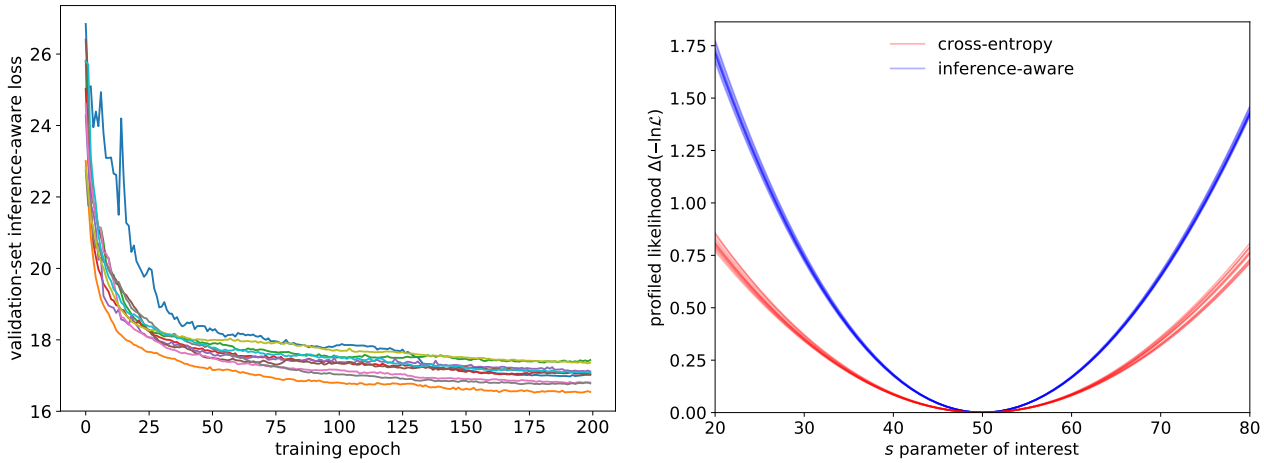
Figure 11: Histograms of summary statistics for signal and background (top) and variation for different values of nuisance parameters compared with the expected signal relative to the nominal background magniture (bottom). The classifier was trained using signal and background samples generated for $r = 0.0$ *and* $\lambda = 3.0$.

interest s. In fact, the performance of inference-aware optimisation as described in Sec. 3.3 will be compared with classification-based summary statistics for a series of inference benchmarks based on the synthetic problem described above that vary in the number of nuisance parameters considered and their constraints:

- **Benchmark 0:** no nuisance parameters are considered, both signal and background distributions are taken as fully specified ($r = 0.0$, $\lambda = 3.0$ and $b = 1000.$).
- **Benchmark 1:** r is considered as an unconstrained nuisance parameter, while $\lambda = 3.0$ and $b = 1000$ are fixed.
- **Benchmark 2:** r and λ are considered as unconstrained nuisance parameters, while $b = 1000$ is fixed.
- **Benchmark 3:** r and λ are considered as nuisance parameters but with the following constraints: $\mathcal{N}(r|0.0, 0.4)$ and $\mathcal{N}(\lambda|3.0, 1.0)$, while $b = 1000$ is fixed.
- **Benchmark 4:** all r, λ and b are all considered as nuisance parameters with the following constraints: $\mathcal{N}(r|0.0, 0.4)$, $\mathcal{N}(\lambda|3.0, 1.0)$ and $\mathcal{N}(b|1000., 100.)$.

When using classification-based summary statistics, the construction of a summary statistic does depend on the presence of nuisance parameters, so the same model is trained independently of the benchmark considered. In real-world inference scenarios, nuisance parameters have often to be accounted for and typically are constrained by prior information or auxiliary measurements. For the approach presented in this section, inference-aware neural optimisation, the effect of the nuisance parameters and their constraints can be taken into account during training. Hence, 5 different training procedures for INFERNO will be considered, one for each of the benchmarks, denoted by the same number.

The same basic network architecture is used both for cross-entropy and inference-aware training: two hidden layers of 100 nodes followed by ReLU activations. The number of nodes



(a) inference-aware training loss *(b) profile-likelihood comparison*

Figure 12: Dynamics and results of inference-aware optimisation: (a) square root of inference-loss (i.e. approximated standard deviation of the parameter of interest) as a function of the training step for 10 different random initialisations of the neural network parameters; (b) profiled likelihood around the expectation value for the parameter of interest of 10 trained inference-aware models and 10 trained cross-entropy loss based models. The latter are constructed by building a uniformly binned Poisson count likelihood of the conditional signal probability output. All results correspond to Benchmark 2.

on the output layer is two when classification proxies are used, matching the number of mixture classes in the problem considered. Instead, for inference-aware classification the number of output nodes can be arbitrary and will be denoted with b, corresponding to the dimensionality of the sample summary statistics. The final layer is followed by a *softmax* activation function and a temperature $\tau = 0.1$ (see Eq. 8 and corresponding explanation) for inference-aware learning to ensure that the differentiable approximations are closer to the true expectations. Standard mini-batch stochastic gradient descent (SGD) is used for training and the optimal learning rate is fixed and decided by means of a simple scan; the best choice found is specified together with the results.

In Fig. 12a, the dynamics of inference-aware optimisation are shown by the validation loss, which corresponds to the approximate expected variance of parameter s, as a function of the training step for 10 random-initialised instances of the INFERNO model corresponding to Benchmark 2. All inference-aware models were trained during 200 epochs with SGD using mini-batches of 2000 observations and a learning rate $\gamma = 10^{-6}$. All the model initialisations converge to summary statistics that provide low variance for the estimator of s when the nuisance parameters are accounted for.

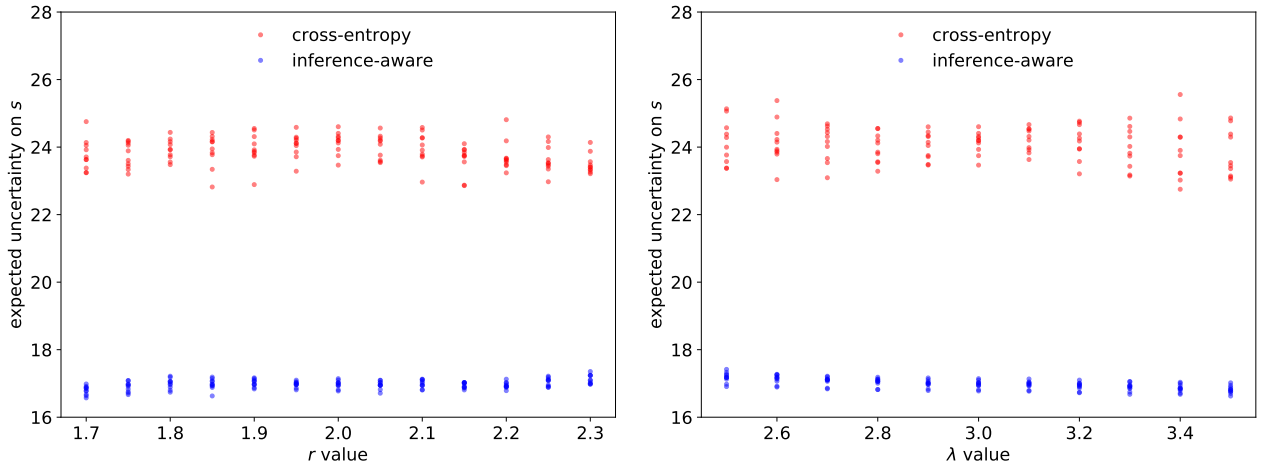
To compare with alternative approaches and verify the validity of the results, the profiled likelihoods obtained for each model are shown in Fig. 12b. The expected uncertainty if the trained models are used for subsequent inference on the value of s can be estimated from the profile width when $\Delta\mathcal{L} = 0.5$. Hence, the average width for the profile likelihood using inference-aware training, 16.97 ± 0.11, can be compared with the corresponding one obtained by uniformly binning the output of classification-based models in 10 bins, 24.01 ± 0.36. The models based on cross-entropy loss were trained during 200 epochs using a mini-batch size of 64 and a fixed learning rate of $\gamma = 0.001$.

A more complete study of the improvement provided by the different INFERNO training procedures is provided in Tab. 14, where the median and 1-sigma percentiles on the expected uncertainty on s are provided for 100 random-initialised instances of each model. In addi-

Table 14: Expected uncertainty on the parameter of interest s for each of the inference benchmarks considered using a cross-entropy trained neural network model, INFERNO customised for each problem as denoted by the succeeding ordinal index, the optimal classifier (see Eq. 20) and the results using the true analytical likelihood. The results for INFERNO matching each problem, i.e. where the set of nuisance parameter and their constraints for the specific benchmark are included in the procedure, are shown with bold characters.

	Benchmark 0	Benchmark 1	Benchmark 2	Benchmark 3	Benchmark 4
NN classifier	$14.99^{+0.02}_{-0.00}$	$18.94^{+0.11}_{-0.05}$	$23.94^{+0.52}_{-0.17}$	$21.54^{+0.27}_{-0.05}$	$26.71^{+0.56}_{-0.11}$
INFERNO 0	$\mathbf{15.51^{+0.09}_{-0.02}}$	$18.34^{+5.17}_{-0.51}$	$23.24^{+6.54}_{-1.22}$	$21.38^{+3.15}_{-0.69}$	$26.38^{+7.63}_{-1.36}$
INFERNO 1	$15.80^{+0.14}_{-0.04}$	$\mathbf{16.79^{+0.17}_{-0.05}}$	$21.41^{+2.00}_{-0.53}$	$20.29^{+1.20}_{-0.39}$	$24.26^{+2.35}_{-0.71}$
INFERNO 2	$15.71^{+0.15}_{-0.04}$	$16.87^{+0.19}_{-0.06}$	$\mathbf{16.95^{+0.18}_{-0.04}}$	$16.88^{+0.17}_{-0.03}$	$18.67^{+0.25}_{-0.05}$
INFERNO 3	$15.70^{+0.21}_{-0.04}$	$16.91^{+0.20}_{-0.05}$	$16.97^{+0.21}_{-0.04}$	$\mathbf{16.89^{+0.18}_{-0.03}}$	$18.69^{+0.27}_{-0.04}$
INFERNO 4	$15.71^{+0.32}_{-0.06}$	$16.89^{+0.30}_{-0.07}$	$16.95^{+0.38}_{-0.05}$	$16.88^{+0.40}_{-0.05}$	$\mathbf{18.68^{+0.58}_{-0.07}}$
Optimal classifier	14.97	19.12	24.93	22.13	27.98
Analytical likelihood	14.71	15.52	15.65	15.62	16.89

tion, results for 100 random-initialised cross-entropy trained models and the optimal classifier and likelihood-based inference are also included for comparison. The confidence intervals obtained using INFERNO-based summary statistics are considerably narrower than those using classification and tend to be much closer to those expected when using the true model likelihood for inference. Much smaller fluctuations between initialisations are also observed for the INFERNO-based cases. The improvement over classification increases when more nuisance parameters are considered. The results also seem to suggest the inclusion of additional information about the inference problem in the INFERNO technique leads to comparable or better results than its omission.



(a) different r value *(b) different λ value*

Figure 13: Expected uncertainty when the value of the nuisance parameters is different for 10 learnt summary statistics (different random initialisation) based on cross-entropy classification and inference-aware technique. Results correspond to Benchmark 2.

Given that a certain value of the parameters $\boldsymbol{\theta}_s$ has been used to learn the summary statistics as described in Algorithm 1 while their true value is unknown, the expected uncertainty on s has also been computed for cases when the true value of the parameters $\boldsymbol{\theta}_{\text{true}}$ differs. The variation of the expected uncertainty on s when either r or λ is varied for classification and

inference-aware summary statistics is shown in Fig. 13 for Benchmark 2. The inference-aware summary statistics learnt for $\boldsymbol{\theta}_s$ work well when $\boldsymbol{\theta}_{\text{true}} \neq \boldsymbol{\theta}_s$ in the range of variation explored.

This synthetic example demonstrates that the direct optimisation of inference-aware losses as those described in the Sec. 3.3 is effective. The summary statistics learnt accounting for the effect of nuisance parameters compare very favourably to those obtained by using a classification proxy to approximate the likelihood ratio. Of course, more experiments are needed to benchmark the usefulness of this technique for real-world inference problems as those found in High Energy Physics analyses at the LHC.

4 Matrix Element Method for $t\bar{t}H$

4.1 Introduction

As described in section 1.1, the $t\bar{t}H$ process is extremely interesting to study in detail with the data available from the LHC. It will be discussed in more detail in this section, including an overview of a powerful multivariate analysis technique to help identify it.

4.1.1 The $t\bar{t}H$ process

The two top quarks produced in $t\bar{t}H$ decay into a W boson and a bottom quark [1] (other top quark decays are so rare that we can safely ignore them here). The Higgs boson predominantly decays into a pair of bottom quarks, $b\bar{b}$ (58% of all Higgs decays, this decay mode was recently observed by both the ATLAS [65] and CMS [66] collaborations). There are two options for the further decay of the W bosons:

- *leptonic*: The W decays into leptons $W^- \to l^- \bar{\nu}_l$, where $l = e, \mu, \tau$
- *hadronic*: The W decays into a quark-antiquark pair $q\bar{q}'$

Leptonic decays happen roughly one third of the time, while the remaining decays are hadronic. The decay widths to different quark combinations are determined by the CKM matrix coefficients, and thus almost all hadronic decays are to "light" quarks u, d, c, s. Jets originating from these will be called light jets. In order to distinguish between these light jets and jets originating from bottom quarks, so-called b-tagging algorithms are used. These multivariate algorithms exploit characteristic features of b-jets. Due to the comparatively long lifetime of b-hadrons, b-jets tend to originate from vertices slightly displaced from the original proton-proton interaction. Other differences between b-jets and light jets include the jet shape, charged particle multiplicity and presence of low-energy leptons. Light jets are unlikely to be b-tagged. The b-tagging algorithm therefore can help discriminate between jets originating from W boson decays and jets from top and Higgs decays. Two W bosons are expected in a $t\bar{t}H$ collision event. Both preferentially decay hadronically, but the analysis of this topology is challenging [67] due to the overwhelming QCD background.

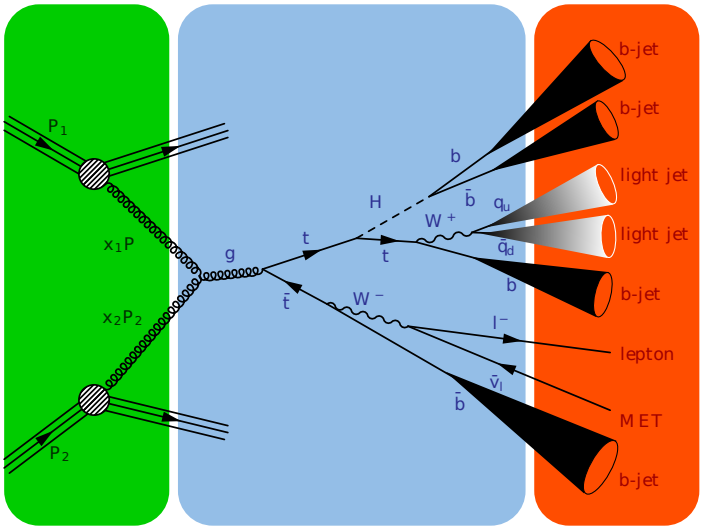


Figure 14: The $t\bar{t}H$ process [68] (edited)

[1] For simplicity, the anti-particle analogue will not always be explicitly given, here for example $t \to W^+ b$, $\bar{t} \to W^- \bar{b}$, compare also Figure 14.

The case where at least one W decays leptonically reduces the amount of jets expected in such $t\bar{t}H$ events in favor of leptons, which are more precisely measured by detectors and simplify the analysis. Both the dileptonic (both W decay leptonically) and the semileptonic (one W decays leptonically, the other hadronically) topology have been considered and analyzed with 36.1 fb^{-1} of data collected at the ATLAS detector at 13 TeV collisions [69] (an analysis was also performed at 8 TeV [70]). This document concentrates on the semileptonic case; a representative lowest order Feynman diagram is shown in Figure 14.

The $t\bar{t}H$ process has been observed for the first time at both the ATLAS and CMS experiments in 2018 ([71, 72]). The best-fit $t\bar{t}H$ signal strength $\mu_{t\bar{t}H}$ (which is the ratio of the observed cross-section to the theoretical prediction $\mu = \sigma_{obs.}/\sigma_{theo.}$) measured at ATLAS is [71]

$$\mu_{t\bar{t}H} = 1.32^{+0.28}_{-0.26} \tag{21}$$

This result takes additional analyses that examine Higgs decays besides $H \to b\bar{b}$ into account.

The measurement of $\mu_{t\bar{t}H}$ can be translated into a measurement of y_t. Determining y_t experimentally to high precision in order to verify agreement with the SM prediction requires improved sensitivity of the $t\bar{t}H$ analyses.

4.1.2 Analysis details and challenges: $t\bar{t}H$, $H \to b\bar{b}$

A major complication for $t\bar{t}H$ analyses where the Higgs decays to a bottom quark pair ($H \to b\bar{b}$) is the existence of another SM process producing identical final state objects with a much larger cross-section. This process is $t\,\bar{t}$ production with an additional gluon splitting into a bottom quark pair. The Feynman diagram looks almost identical to Figure 14, with just the Higgs boson H is replaced by a gluon g. Unfortunately, the kinematic differences between this $t\bar{t}+b\bar{b}$ "background" process and the $t\bar{t}H$ "signal" process are very small. The spin difference of H and g results in different angular distribution of the decay products $b\bar{b}$, and the invariant mass of the decay products is close to the Higgs mass m_H in the signal case (and more broadly distributed for the background). The QCD colour flow via the gluon in the background process can cause small differences as well.

Further background processes need to be considered in an analysis, but they are subdominant and will not be discussed here. Figure 15 shows various analysis regions considered in the recent ATLAS search for $t\bar{t}H$ with $H \to b\bar{b}$, using 36.1 fb^{-1} of data [69]. Each pie chart corresponds to one region in the semileptonic channel. SR denotes regions enriched in signal, and in particular the most powerful signal region is $\mathrm{SR}_1^{\geq 6j}$. This region is defined by requiring at least six jets, out of which at least four are b-tagged. The pie charts show the relative contributions of various background processes to the region, and $\mathrm{SR}_1^{\geq 6j}$ is dominated by $t\bar{t}+b\bar{b}$ production, with only minor non-$t\,\bar{t}$ backgrounds.

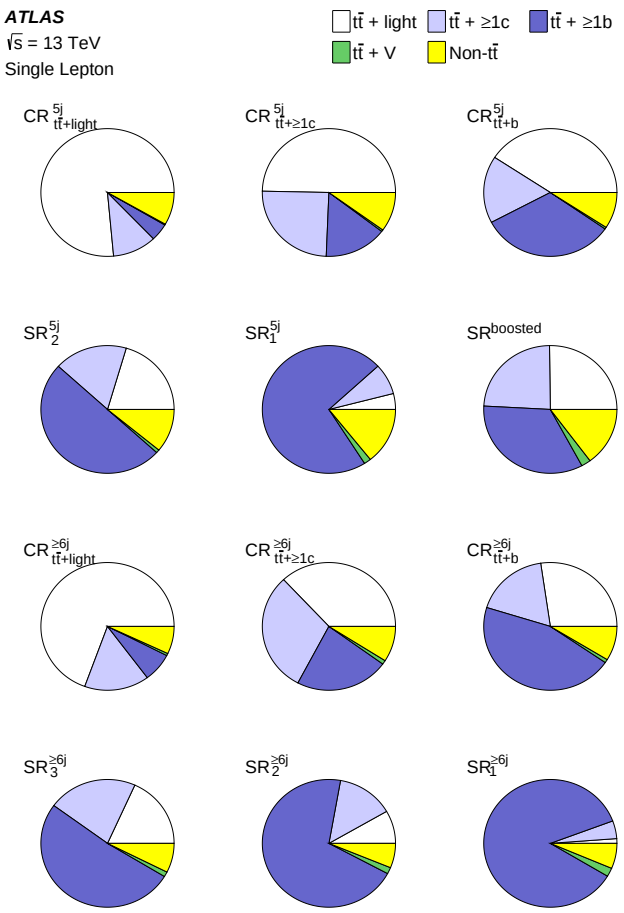


Figure 15: Breakdown of background contributions to the analysis regions considered in the semileptonic channel of the $t\bar{t}H$, $H \to b\bar{b}$ ATLAS analysis[69]

Figure 16 presents the signal and background contributions to the various analysis regions considered. It underlines the power of the $\mathrm{SR}_1^{\geq 6j}$ region.

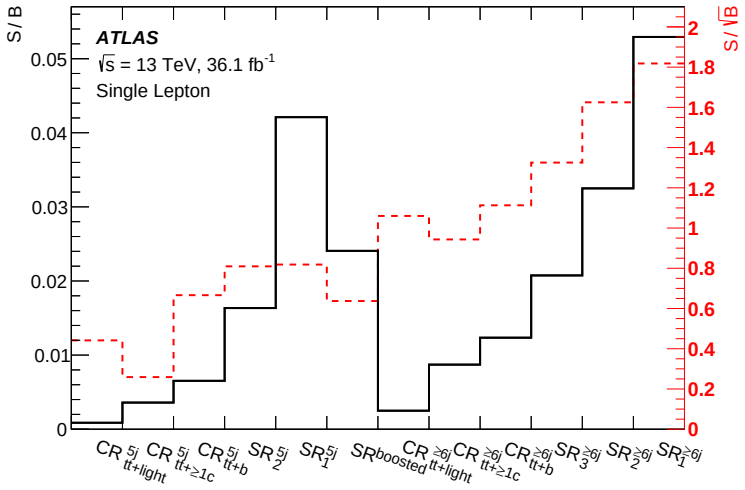


Figure 16: Signal contributions to regions in the semileptonic channel [69]

The determination of $\mu_{t\bar{t}H}$ requires the comparison of an expected number of identified signal-like collision events to a prediction made by the SM. Due to the similarity of the dominant $t\,\bar{t}$ background, it is difficult to select a region of high signal purity with a sufficient selection efficiency (a high selection efficiency is mandatory due to the low signal cross-section, so the usage of additional techniques to separate the signal from background contributions is mandatory). This is further complicated by the large uncertainties on the predicted background distribution shapes and yields. The approach chosen to remedy these issues is a simultaneous profile likelihood fit of the measured data to Monte Carlo generated predictions in multiple regions. The regions are defined by the number n of identified jets and b-tag requirements. Regions with many jets and tight b-tagging requirements contain most of the signal and as such are called "signal regions". Multivariate techniques are used in these regions to build discriminants, which aim to separate signal from background. The remaining "control regions" allow constraining backgrounds and systematic uncertainties when fitted together with the signal regions. A lot of the dominant systematic uncertainties concern the modelling of $t\,\bar{t}$ with additional jets, and the respective fractions of different jet flavors. The $t\,\bar{t}\,+c$ and $t\,\bar{t}\,+b$ fractions are used as unconstrained parameters in the fit; additional uncertainties due to the choice of Monte Carlo generator, parton shower, hadronization model, detector effects, etc., are implemented as Gaussian or log-normal priors.

Even though the signal process of interest is expected to have exactly 6 jets and 4 b-tags, several things can change this:

- jets with $|\eta| > 2.5$ or transverse momentum $p_T < 25$ GeV are not considered in the analysis due to detector limitations,
- additional interactions in a proton-proton collision (so-called "pileup") can result in more jets being reconstructed,
- the processes can happen at a higher order in QCD, producing initial and final state radiation, which can also result in more jets being reconstructed,
- the b-tagging algorithm can mistakenly identify a light jet as originating from a b-hadron decay, or can fail to b-tag a jet originating from a b-hadron,
- top quarks at very high momenta can appear as a single, large jet in the detector instead of being resolved into multiple objects.

4.2 The Matrix Element Method

The Matrix Element Method (MEM) provides a powerful way of discriminating $t\bar{t}H$ from the $t\bar{t} + b\bar{b}$ background, and had already been used in the Run-1 ATLAS analysis of $t\bar{t}H$ [70]. It provides a way to calculate the likelihood of an event originating from a given production mechanism, by calculating probability densities via Fermi's golden rule and making use of the SM predictions for these processes. Of particular interest here is distinguishing between $t\bar{t}H$ and $t\bar{t} + b\bar{b}$. The MEM is thus used to calculate the two likelihoods L_S and L_B:

- L_S: signal likelihood, likelihood of the event having been produced via a $t\bar{t}H$ Feynman diagram
- L_B: background likelihood, likelihood of the event having been produced via a $t\bar{t} + b\bar{b}$ Feynman diagram

These likelihoods are defined below:

$$L_i = \sum \int \frac{f_1\left(x_1, Q^2\right) f_2\left(x_2, Q^2\right)}{|\vec{q}_1||\vec{q}_2|} \left|\mathcal{M}_i\left(\mathbf{Y}\right)\right|^2 T\left(\mathbf{X};\mathbf{Y}\right) \mathrm{d}\mathbf{\Phi}_n\left(\mathbf{Y}\right) \tag{22}$$

The likelihood is a sum over different possible initial states (quarks or gluons from within the colliding protons) and over multiple jet-parton assignments. It contains a product of parton distribution functions $f_1 f_2$, which are probability distribution functions for a certain parton with momentum $\vec{q_j}$ to carry energy fraction x_j of the proton in a collision at scale Q (for the two initial states $j = 1, 2$). The matrix element (ME) $\mathcal{M}_i$ is calculated for a phase space configuration $\mathbf{Y}$ at parton level, where the index i refers to either signal or background Feynman diagrams (which typically are evaluated only at leading order). The transfer function T represents the probability for a given jet measured on reconstruction level $\mathbf{X}$ to be originating from a certain parton level configuration $\mathbf{Y}$. In an analysis, only the reconstruction level information $\mathbf{X}$ is available (provided by a detector measurement of the collision remnants). All unknown parameters needed to fully specify the event are integrated out via the phase space factor $\mathrm{d}\mathbf{\Phi}_n$. This integration includes the neutrino expected in semileptonic $t\bar{t}H$ events, which escapes the detector unmeasured. The transfer function T is used to constrain the size of the phase space significantly contributing to the likelihood L_i, and therefore limits the integration range.

The most powerful test statistic to discriminate between signal- and background-like events is given according to the Neyman-Pearson lemma by the likelihood ratio L_S/L_B, which will be referred to as $\mathrm{MEM_{D1}}$:

$$\mathrm{MEM_{D1}} = \log_{10}\left(L_S\right) - \log_{10}\left(L_B\right) \tag{23}$$

4.2.1 Challenges

Before going into the MEM implementation in more detail, here is a quick description of some challenges in practice.

Even though the MEM can make use of a complete physics description of the dominant signal and background processes at leading order (LO), the discrimination power achieved is limited by various complications.

Detector-related Challenges:

- Imperfect object reconstruction in the detector can lead to jets being mis-measured or not identified at all,
- When there are more jets present in a collision event than the six expected from the LO Feynman diagram, a selection has to be made to calculate the likelihoods, this selection may be wrong,
- It is unclear from which quark on the Feynman diagram level each reconstructed jet originated from, and hence multiple assignments (including wrong ones) need to be considered.

Computing-related Challenges:

- The MEM likelihood calculations take extremely long, and consequently significant approximations need to be used,
- Detailed sampling of the full integration phase space is not possible in a reasonable amount of time, hence peaks in the matrix elements may be missed during integration.

4.3 Approximations and integration strategy

4.3.1 Integration strategy

A full integration over the phase space $\mathbf{\Phi}_n$ is not feasible in practice. The most important assumption made is that the measured jet direction in the detector corresponds exactly to the direction of the quarks on parton (Feynman diagram) level $\mathbf{Y}$. This reduces the amount of integration dimensions from 24 (8 final state particles with three degrees of freedom each, they are assumed to be on-shell) to 12 (6 jets with one degree of freedom, 2 leptons with three degrees of freedom). The charged lepton is assumed to be well-measured by the ATLAS detector and not integrated over (reducing the integration dimensions by 3 degrees of freedom). Lastly, the incoming proton beams are assumed to be exactly aligned with the beam axis, such that the total momentum transverse to this axis of all final state particles should vanish ($\sum p_T = 0$) by momentum conservation. With these assumptions, the neutrino has only one degree of freedom remaining, which can be chosen as its momentum along the z-axis. Seven degrees of freedom then remain to be integrated over.

Additional simplifications are possible:

- The Higgs boson propagator largely suppresses any configuration where the Higgs boson is far off-shell. This means that large parts of the $\mathbf{\Phi}_n$ phase space contain negligible contributions to the full integral in (22), and only configurations where the invariant mass of the quarks from the Higgs decay is close to m_H need to be considered in the evaluation of the $t\bar{t}H$ likelihood L_S. It is then possible to treat the Higgs decay like a Dirac δ and reduce the amount of integration dimensions by one. The Dirac δ is a good approximation of the very narrow Higgs width, which is significantly below the detector resolution.

- A similar argument can be made for the decays of the W boson. When forcing the leptonically decaying W boson to be exactly on mass-shell in the matrix element (ME) calculation, the resulting second order polynomial can be solved for the neutrino momentum p_z. Instead of integrating over this neutrino momentum, both solutions of the resulting second order polynomial may be summed in the evaluation of L_S and L_B. It should be noted that this approximation is worse than in the Higgs case, as the width of the W boson is significantly larger.

Both of these additional approximations were tested, but found to not improve the performance of the method in practice and are thus not used. Simplification or approximations do not render the method invalid, but rather imply that the information in the data is not used in the most optimal way, which limits the overall performance (while in turn drastically speeding up the processing).

4.3.2 Integration variables

The phase space integrals are performed by integrating over the following 7 variables:

- energy of bottom quark from leptonic top decay
- energy of bottom quark from hadronic top decay
- energy of bottom quark (leading in p_T) from Higgs decay (if signal hypothesis) or gluon splitting (if background)
- energy of bottom quark (subleading in p_T) from Higgs decay (if signal hypothesis) or gluon splitting (if background)

- energy of light quark (leading in p_T) from hadronic W decay
- energy of light quark (subleading in p_T) from hadronic W decay
- z momentum p_z of the neutrino from the leptonic W decay

The integration ranges for jets are centered around the measured jet energy E_j, and determined from transfer function distributions. See the section on transfer functions for details. The neutrino p_z integration is done over the interval [-1 TeV, 1 TeV].

4.3.3 Jet-parton assigments and permutations

The MEM is only used in an analysis region with at least 6 jets and 4 b-tags. 6 quarks are expected for the LO ME evaluation, so for events with $n > 6$ jets, two methods are possible:

- consider all possible 6-jet subsets of the whole n jet set, sum up the corresponding likelihoods
- select only one subset to evaluate the likelihood and ignore the other possible assignments

The first approach is computationally expensive and thus not chosen here. Instead, the subset choice to select the 6 jets describing the configuration $\mathbf{X}$ used in the likelihood evaluation is done in two steps:

- Consider all jets that were b-tagged at the highest criteria, and select the top 4 ordered by their p_T. These will be referred to as "b-jets".
- Evaluate the invariant masses m_{jj} of all possible 2-jet pairs, not including the 4 jets selected as b-jets in the first step. Pick the pair that minimizes $|m_W - m_{jj}|$ (for W mass $m_W = 80.4$ GeV), these jets will be referred to as "light jets".

This algorithm selects two light jets, which will always be assigned to the hadronic W decay in the MEM calculation. From the four b-jets, two need to be assigned to the Higgs decay, one to the leptonic top decay, and one to the hadronic top decay. This results in $\frac{4!}{2} = 12$ different jet-parton assignements, also called "permutations", which are being considered.

4.4 Technical implementation

The likelihood calculation is implemented in a framework based on the results of [73]. The VEGAS algorithm [74] is used to carry out the phase space integration. An interface to LHAPDF [75] provides the parton distribution function information, while MadGraph5_aMC@NLO [76] is used for the matrix element evaluation.

4.4.1 Framework

The MEM calculations are steered by a python framework. In order to speed up the computationally costly integration process, the integrand is implemented to be usable with CUDA [77] or OpenCL [78]. The framework can thus run on many CPU cores or GPUs in parallel. When running on a Intel Xeon E5-2650 CPU, one event is processed in 1.8 seconds.

4.4.2 Integration process

The integration is performed using importance sampling with the VEGAS algorithm. The process is tuned for performance, while trying to limit the cost to separation power as much as possible. Each permutation is integrated separately, and the evaluation of signal and background likelihoods are done separately as well. The approach is described below, and consists of multiple integration rounds. At each integration round, VEGAS evaluates the integrand at up to 1024 phase space points. After each integration round, VEGAS updates the grid used to sample the phase space and to evaluate the phase space integral.

- adaption phase: The first round of integration is used for VEGAS to very roughly map out the phase space, and adjust its integration grid. Integration results from this round are not included in the MEM calculation, but the adjusted grid is used for the consecutive rounds.

- main integration: After the initial grid adaption, three rounds of integration are performed. The grid is updated after every round, and the integral estimate is calculated combining all integration rounds in this main integration step. In case that the reported uncertainty on the integral is already below 1%, the process stops here and the calculated result is reported.

- pruning: After the main integration round, the likelihoods calculated for all permutations are compared. No further integration refinement is performed for permutations that are smaller than 1% of the likelihood in the largest permutation. The total likelihood at the end will be a sum over the likelihoods of all permutations, so it is most important to correctly evaluate the permutations that contribute most to this sum. Typically the likelihoods of different permutations range over multiple orders of magnitude, and thus several permutations contribute a negligible amount to the final result (and do not need to be calculated to high accuracy).

- refinement: Another set of three integration rounds is performed for all permutations unaffected by the previous step. The integration grid is still allowed to adjust after each round, and the integration is stopped if the reported uncertainty drops below 1%.

After this process, the signal and background likelihoods are calculated as a sum over all 12 permutations.

4.4.3 Parton distribution functions

The CT10 pdf set is used and obtained by interfacing to LHAPDF. The pdf information is saved to a grid, enabling fast look-up. The chosen factorization scale is dynamic:

$$Q^2 = \left(\sum_i E_i\right)^2 - \left(\sum_i p_{z,i}\right)^2 \tag{24}$$

where i runs over all 8 final state partons.

4.4.4 Matrix Elements

The ME evaluation is done via standalone code exported from MadGraph5_aMC@NLO. For the signal hypothesis, leading order top quark pair production with a Higgs boson decaying to a bottom quark pair is considered. The top quarks are each forced to decay to W boson and b quark, one W is forced to decay leptonically, while the other one decays hadronically. This can be generated with the following string in MadGraph5_aMC@NLO:

- generate p p > t t~h, h > b b~, (t > w+ b, w+ > l+ vl), (t~> w- b~, w- > p p)

For the background hypothesis, only top quark pair production with an additional bottom quark pair is considered. The system of top quarks is forced to decay the same way as for the signal hypothesis. The process can be generated with the following string:

- generate p p > t t~b b~, (t > w+ b, w+ > l+ vl), (t~> w- b~, w- > p p)

All (non-physical) helicity combinations with no contributions to the ME are removed from the calculation by hand in order to speed up the evaluation. The Feynman diagrams contributing to the ME calculation can be split up into two classes, those describing gg interactions and those produced by $q\bar{q}$ interactions:

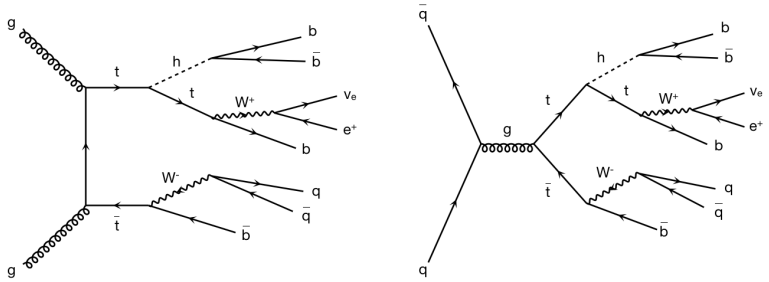


Figure 17: Representative ttH Feynman diagrams of the gg (left) and $q\bar{q}$ kind (right)

The gg diagrams dominate in the cross-section calculation, especially at low Q^2. By default, $q\bar{q}$ diagrams are thus not considered, resulting in a moderate speed improvement of roughly 30%, while no impact on the separation power of the MEM was observed.

4.4.5 Transfer functions

Transfer functions serve to constrain the phase space integration volume, and at the same time provide an approximate model for the reconstruction level energy E_j dependence on the parton level configuration E_p. Two functional forms are used to describe light and b-jets. Light jets are described by a double Gaussian:

$$W_{jet}^{light}\left(E_j, E_p\right) = \frac{1}{\sqrt{2\pi}\left(\sigma_1 + A\sigma_2\right)}\left[\exp\left(\frac{-\left(E_p - E_j - \mu_1\right)^2}{2\sigma_1^2}\right) + A\exp\left(\frac{-\left(E_p - E_j - \mu_2\right)^2}{2\sigma_2^2}\right)\right] \tag{25}$$

An example of double Gaussian transfer functions for various jet energies is shown in Figure 18. The transfer functions become increasingly wide for higher jet energies.

The b-jet parametrization is done via a crystal ball function [79, 80]:

$$W_{jet}^{b}\left(E_j, E_p\right) = \begin{cases} N \cdot \exp\left(-\frac{(E_p - E_j - \mu)^2}{2\sigma^2}\right), \frac{E_p - E_j - \mu}{\sigma} < \alpha \\ N \cdot A \cdot \left(B + \frac{E_p - E_j - \mu}{\sigma}\right)^{-n}, \frac{E_p - E_j - \mu}{\sigma} \geq \alpha \end{cases} \tag{26}$$

which has parameters as given by

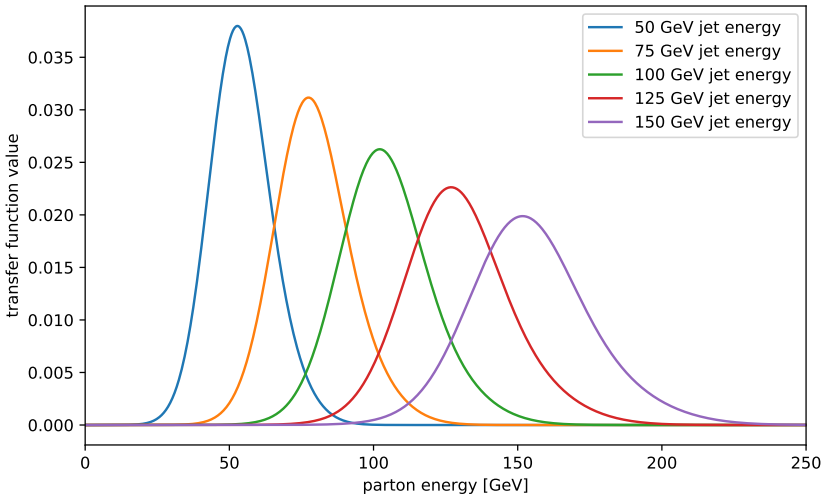


Figure 18: Exemplary transfer function distributions using the double Gaussian functions, showing the parton energy dependence for various measured jet energies

$$A = \left(\frac{n}{|\alpha|}\right)^n \cdot \exp\left(-\frac{(|\alpha|)^2}{2}\right) \tag{27}$$

$$B = \frac{n}{|\alpha|} - |\alpha| \tag{28}$$

$$C = \frac{n}{|\alpha|} \cdot \frac{1}{n-1} \cdot \exp\left(-\frac{(|\alpha|)^2}{2}\right) \tag{29}$$

$$D = \sqrt{\frac{\pi}{2}} \cdot \left(1 + \operatorname{erf}\left(\frac{|\alpha|}{\sqrt{2}}\right)\right) \tag{30}$$

$$N = \frac{1}{\sigma\,(C+D)} \tag{31}$$

An example for crystal ball transfer functions is shown below in Figure 19. Compared to the double Gaussian case, the power law tail behavior at high parton energy values is visible for the crystal ball functions here.

In order to limit the phase space volume for the MEM integration, tails in the TF distributions of possible E_p values (given a measured E_j) are truncated. This allows for quicker integral convergence, without compromising separation power. These regions in the TF tails are drastically suppressed by the TF and thus do not contribute significantly to the overall likelihood. The jet integration ranges are defined symmetrically like

$$E_p \in [E_j \cdot (1-R)\,, E_j \cdot (1+R)] \tag{32}$$

for integration range parameter R. These parameters can be derived separately for light and b-jets and depend on the measured jet energy.

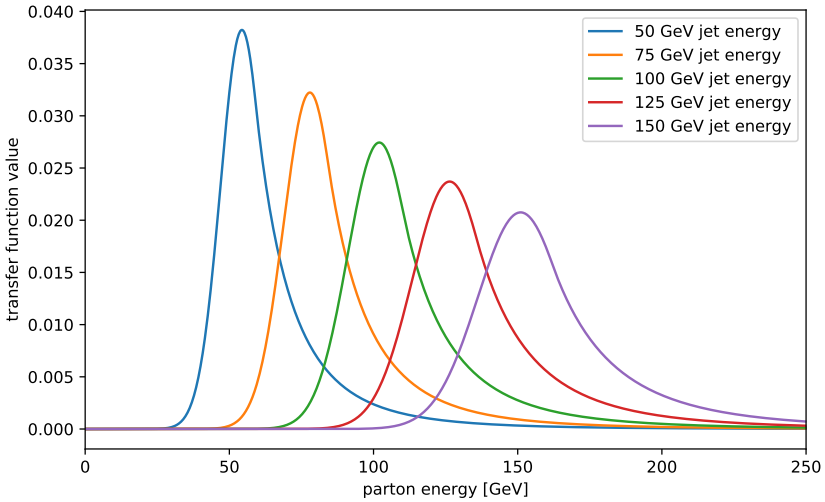


Figure 19: Exemplary transfer function distributions using the crystal ball functions, showing the parton energy dependence for various measured jet energies

4.5 Results

The final results achieved with the MEM implementation described in this document are presented here. They are taken from an ATLAS analysis of 36.1 fb^{-1} of data collected at $\sqrt{s} = 13$ TeV at the LHC [69]. The MEM discriminant, calculated according to equation 23, is shown below in Figure 20 for the $\mathrm{SR}_1^{\geq 6j}$ region. A sigmoid function is used to map it into the interval $[0, 1]$ for presentation purposes: $[1 + \exp(-\mathrm{MEM}_{\mathrm{D1}} - 4)]^{-1}$.

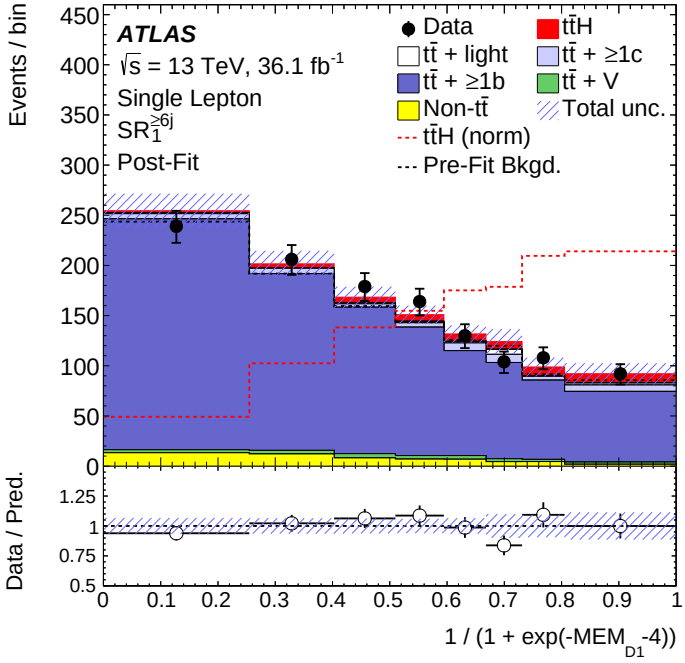


Figure 20: Final MEM discriminant used in the analysis in the $SR_1^{\geq 6j}$ region, distribution after profile likelihood fit to data [69]

In the upper panel, the stacked filled histograms show the predicted distribution of simu-

lated data, after a profile likelihood fit to the data measured by the ATLAS detector (shown with the black points) has been performed. The prediction is in good agreement with the measured data, and the MEM distribution is modelled well. The dashed red line shows the relative distribution of the $t\bar{t}H$ signal, normalized to the total background contribution. The signal distribution peaks towards high values of $\mathrm{MEM_{D1}}$ as expected, while the background is predominantly located at the other end of the distribution. The MEM calculation thus provides strong separation between the $t\bar{t}H$ signal and the background processes, which are dominated by $t\bar{t}+b\bar{b}$ shown in dark blue. The lower panel of the plot shows the ratio of data and simulation, and confirms the good agreement within the associated uncertainties.

For the final signal extraction in this $t\bar{t}H$ analysis, the Matrix Element Method likelihood ratio is combined with additional techniques in a boosted decision tree classifier. The output of this classifier is ultimately used in the statistical analysis. Figure 21 shows the distribution of this classifier in the $\mathrm{SR}_1^{\geq 6j}$ region. When comparing the normalized $t\bar{t}H$ distribution (dashed red line) to the background shape, this classifier performs better than the MEM likelihood ratio by itself. The additional inputs to the classifier improve performance lost due to some of the approximations that had to be done in the MEM calculation due to the associated computational cost.

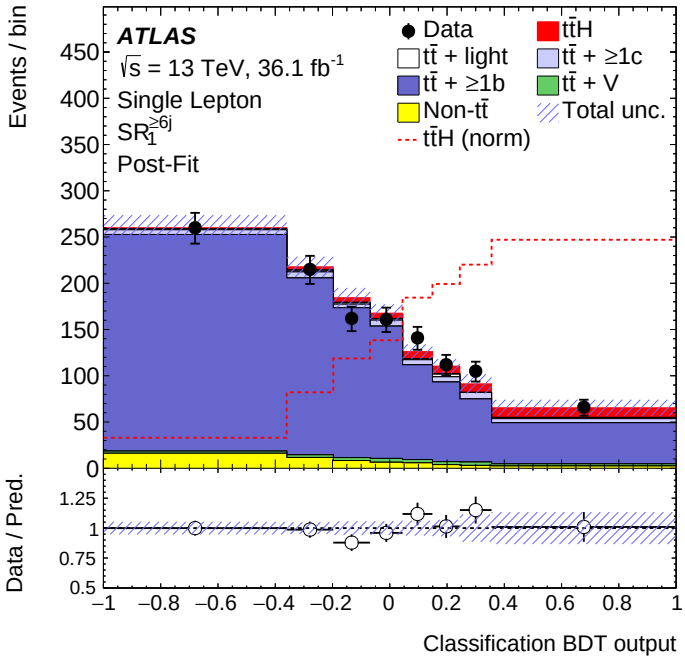


Figure 21: Boosted decision tree distribution in most sensitive signal region, post-fit [69]

5 Conclusions

In the present document we describe the construction of three different ML-driven solutions to specific problems of data analysis in HEP arising in the search for the Higgs boson and the measurement of its properties.

5.1 $h \to \tau\,\bar{\tau}$ classification

In Sec. 2 we examined the potential improvements of several new methods in deep learning. Whilst these methods had been developed outside of HEP, they were still found to bring benefits when applied to a well established example of a common HEP problem; the Higgs ML Kaggle challenge.

Starting from a single, simple DNN model which achieves a score of 3.419 - 886$^{\text{th}}$ out of 1786, we found that ensembling brought a 6 % score improvement - 507$^{\text{th}}$. Moving to a more modern activation function brought a 0.6 % improvement - 304$^{\text{th}}$. Scheduling the learning rate improved further the score by 0.8 % - 140$^{\text{th}}$. Finally, by introducing data augmentation at both train and test time we improved by and additional 4 % - 1$^{\text{st}}$. Overall, these constitute a total improvement of 12 % over the baseline, however this is bought at the cost of a 450 % increase in training time, and a 25 000 % increase in inference time.

The final solution achieves effectively the same level of performance as the winning solution, but with the advantage that it can be trained (evaluated) in less than 10 % (70 %) of the time using consumer-grade hardware.

5.2 INFERNO: Inference-aware Neural Optimisation

Classification-based summary statistics for mixture models often suffer from the need of specifying a fixed model of the data, thus neglecting the effect of nuisance parameters in the learning process. The effect of nuisance parameters is only considered downstream of the learning phase, resulting in sub-optimal inference on the parameters of interest.

In Section 3 of the present document we have described a new approach for building non-linear summary statistics for likelihood-free inference that directly minimises the expected variance of the parameters of interest, which is considerably more effective than the use of classification surrogates when nuisance parameters are present.

The results obtained for the synthetic experiment considered clearly demonstrate that machine learning techniques, in particular neural networks, can be adapted for learning summary statistics that match the particularities of the inference problem at hand, greatly increasing the information available for subsequent inference. The application of INFERNO to non-synthetic examples where nuisance parameters are relevant, such as the systematic-extended Higgs dataset [81], are left for future studies.

The technique presented can be applied to arbitrary likelihood-free problems as long as the effect of parameters over the simulated data can be implemented as a differentiable transformations. As a possible extension, alternative non-parametric density estimation techniques such as kernel density could very well be used in place Poisson count models.

5.3 Matrix Element Method

The Matrix Element Method offers itself as a powerful tool to distinguish between different physics production processes. By design, it makes use of the whole event kinematics on parton level, including all correlations between particles. It is an extremely useful tool in complex analysis like the $t\bar{t}H$ case described in this document. At the same time, the method comes at a high computational cost. Section 4 of this report summarized steps taken to mitigate

this cost, with the aim of minimal impact to the MEM separation power. Nevertheless, some approximations certainly are overly simple, and with more computation power available in the future, the method is likely to continue to shine. Three promising avenues are:

- extending the description to next-to-leading order matrix element (extreme computational cost, only slowly starting to become possible),
- finding better ways of jet-parton matching, and relax some assumptions on the transfer functions,
- enhancing the method with machine learning techniques.

The use of advanced machine learning developments combined with the ideas behind the MEM are likely to ensure that this analysis technique will retain its relevance in the future.

A Sufficient Statistics for Mixture Models

Let us consider the general problem of inference for a two-component mixture problem, which is very common in scientific disciplines such as High Energy Physics. While their functional form will not be explicitly specified to keep the formulation general, one of the components will be denoted as signal $f_s(\boldsymbol{x}|\boldsymbol{\theta})$ and the other as background $f_b(\boldsymbol{x}|\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is are of all parameters the distributions might depend on. The probability distribution function of the mixture can then be expressed as:

$$p(\boldsymbol{x}|\mu, \boldsymbol{\theta}) = (1-\mu) f_b(\boldsymbol{x}|\boldsymbol{\theta}) + \mu f_s(\boldsymbol{x}|\boldsymbol{\theta}) \tag{33}$$

where μ is a parameter corresponding to the signal mixture fraction. Dividing and multiplying by $f_b(\boldsymbol{x}|\boldsymbol{\theta})$ we have:

$$p(\boldsymbol{x}|\mu, \boldsymbol{\theta}) = f_b(\boldsymbol{x}|\boldsymbol{\theta}) \left(1 - \mu + \mu \frac{f_s(\boldsymbol{x}|\boldsymbol{\theta})}{f_b(\boldsymbol{x}|\boldsymbol{\theta})}\right) \tag{34}$$

from which we can already prove that the density ratio $s_{s/b} = f_s(\boldsymbol{x}|\boldsymbol{\theta})/f_b(\boldsymbol{x}|\boldsymbol{\theta})$ (or alternatively its inverse) is a sufficient summary statistic for the mixture coefficient parameter μ. This would also be the case for the parametrization using s and b if the alternative $\mu = s/(s+b)$ formulation presented for the synthetic problem in Sec. 3.5.1.

However, previously in this work (as well as for most studies using classifiers to construct summary statistics) we have been using the summary statistic $s_{s/(s+b)} = f_s(\boldsymbol{x}|\boldsymbol{\theta})/(f_s(\boldsymbol{x}|\boldsymbol{\theta}) + f_b(\boldsymbol{x}|\boldsymbol{\theta}))$ instead of $s_{s/b}$. The advantage of $s_{s/(s+b)}$ is that it represents the conditional probability of one observation $\boldsymbol{x}$ coming from the signal assuming a balanced mixture, and hence is bounded between zero and one. This greatly simplifies its visualisation and non-parametric likelihood estimation. Taking Eq. 34 and manipulating the subexpression depending on μ by adding and subtracting 2μ we have:

$$p(\boldsymbol{x}|\mu, \boldsymbol{\theta}) = f_b(\boldsymbol{x}|\boldsymbol{\theta}) \left(1 - 3\mu + \mu \frac{f_s(\boldsymbol{x}|\boldsymbol{\theta}) + f_b(\boldsymbol{x}|\boldsymbol{\theta})}{f_b(\boldsymbol{x}|\boldsymbol{\theta})}\right) \tag{35}$$

which can in turn can be expressed as:

$$p(\boldsymbol{x}|\mu, \boldsymbol{\theta}) = f_b(\boldsymbol{x}|\boldsymbol{\theta}) \left(1 - 3\mu + \mu \left(1 - \frac{f_s(\boldsymbol{x}|\boldsymbol{\theta})}{f_s(\boldsymbol{x}|\boldsymbol{\theta}) + f_b(\boldsymbol{x}|\boldsymbol{\theta})}\right)^{-1}\right) \tag{36}$$

hence proving that $s_{s/(s+b)}$ is also a sufficient statistic and theoretically justifying its use for inference about μ. The advantage of both $s_{s/(s+b)}$ and $s_{s/b}$ is they are one-dimensional and do not depend on the dimensionality of $\boldsymbol{x}$ hence allowing much more efficient non-parametric density estimation from simulated samples. Note that we have been only discussing sufficiency with respect to the mixture coefficients and not the additional distribution parameters $\boldsymbol{\theta}$. In fact, if a subset of $\boldsymbol{\theta}$ parameters are also relevant for inference (e.g. they are nuisance parameters) then $s_{s/(s+b)}$ and $s_{s/b}$ are not sufficient statistics unless the $f_s(\boldsymbol{x}|\boldsymbol{\theta})$ and $f_b(\boldsymbol{x}|\boldsymbol{\theta})$ have very specific functional form that allows a similar factorisation.

B Software details

The investigation performed in Sec. 2 made use of several packages, which are detailed in Tab. 15. The framework and notebook experiments are made available at `https://github.com/GilesStrong/QCHS-2018`.

Software	Version	References	Use/Notes
Keras	2.1.6	[82]	Implementing neural networks
Tensorflow	1.8.0	[83]	Keras back-end
Scikit-Learn	0.19.1	[84]	Cross-validation and pre-processing
Matplotlib	2.1.2	[85]	Plot production
Seaborn	0.8.1	[86]	Plot production
NumPy	1.14.0	[87]	Data analysis and computation
Pandas	0.22.0	[88]	Data analysis and computation

Table 15: Software used for the investigation performed in Sec. 2.

References

[1] *A search using multivariate techniques for a standard model Higgs boson decaying into two photons*. Tech. rep. CMS-PAS-HIG-12-001. Geneva: CERN, 2012. URL: https://cds.cern.ch/record/1429931.

[2] A Graves and J Schmidhuber. "Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks". In: *Advances in Neural Information Processing Systems* 22 (2009), pp. 545–552. URL: http://people.idsia.ch/~juergen/nips2009.pdf.

[3] George E. Dahl et al. "Large vocabulary continuous speech recognition with context-dependent DBN-HMMS". In: *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2011), pp. 4688–4691.

[4] M. Andrews et al. "End-to-End Physics Event Classification with the CMS Open Data: Applying Image-based Deep Learning on Detector Data to Directly Classify Collision Events at the LHC". In: (2018). arXiv: 1807.11916 [hep-ex].

[5] Michela Paganini, Luke de Oliveira, and Benjamin Nachman. "CaloGAN : Simulating 3D high energy particle showers in multilayer electromagnetic calorimeters with generative adversarial networks". In: *Phys. Rev.* D97.1 (2018), p. 014021. DOI: 10.1103/PhysRevD.97.014021. arXiv: 1712.10321 [hep-ex].

[6] "Heavy flavor identification at CMS with deep neural networks". In: *CMS-DP-2017-005* (2017). URL: https://cds.cern.ch/record/2255736.

[7] J. Deng et al. "ImageNet: A Large-Scale Hierarchical Image Database". In: *CVPR09*. 2009.

[8] K. A. Olive et al. "Review of Particle Physics". In: *Chin. Phys.* C38 (2014), p. 090001. DOI: 10.1088/1674-1137/38/9/090001.

[9] Lyndon Evans and Philip Bryant. "LHC Machine". In: *JINST* 3 (2008), S08001. DOI: 10.1088/1748-0221/3/08/S08001.

[10] Andy Buckley et al. "General-purpose event generators for LHC physics". In: *Phys. Rept.* 504 (2011), pp. 145–233. DOI: 10.1016/j.physrep.2011.03.005. arXiv: 1101.2599 [hep-ph].

[11] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". In: *CoRR* abs/1603.02754 (2016). URL: http://arxiv.org/abs/1603.02754.

[12] J. Allison et al. "Geant4 developments and applications". In: *IEEE Transactions on Nuclear Science* 53.1 (2006), pp. 270–278. ISSN: 0018-9499. DOI: 10.1109/TNS.2006.869826.

[13] S. Agostinelli et al. "Geant4—a simulation toolkit". In: *Nucl. Instr.Meth. A* 506.3 (2003), pp. 250 –303. ISSN: 0168-9002. DOI: http://dx.doi.org/10.1016/S0168-9002(03)01368-8. URL: http://www.sciencedirect.com/science/article/pii/S0168900203013688.

[14] G Aad et al. "The ATLAS Experiment at the CERN Large Hadron Collider". In: *J. Instrum.* 3 (2008). Also published by CERN Geneva in 2010, S08003. 437.

[15] R. Johnson and T. Zhang. "Learning Nonlinear Functions Using Regularized Greedy Forest". In: *ArXiv e-prints* (Sept. 2011). arXiv: 1109.0887 [stat.ML].

[16] Glen Cowan et al. "Asymptotic formulae for likelihood-based tests of new physics". In: *Eur. Phys. J.* C71 (2011). [Erratum: Eur. Phys. J.C73,2501(2013)], p. 1554. DOI: 10.1140/epjc/s10052-011-1554-0,10.1140/epjc/s10052-013-2501-z. arXiv: 1007.1727 [physics.data-an].

[17] "Dataset from the ATLAS Higgs Boson Machine Learning Challenge 2014". In: *CERN Open Data Portal* (2014). DOI: 10.7483/OPENDATA.ATLAS.ZBP2.M5T8.

[18] Claire Adam-Bourdarios et al. "Dataset from the ATLAS Higgs Boson Machine Learning Challenge 2014". In: *CERN Open Data Portal* (2014). DOI: 10.7483/OPENDATA.ATLAS.ZBP2.M5T8.

[19] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Neurocomputing: Foundations of Research". In: (1988). Ed. by James A. Anderson and Edward Rosenfeld, pp. 696–699. URL: http://dl.acm.org/citation.cfm?id=65669.104451.

[20] Raman Arora et al. "Understanding Deep Neural Networks with Rectified Linear Units". In: *CoRR* abs/1611.01491 (2016). URL: http://arxiv.org/abs/1611.01491.

[21] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *JMLR Workshop and Conference Proceedings* 9 (2010). URL: http://jmlr.org/proceedings/papers/v9/glorot10a/glorot10a.pdf.

[22] Kaiming He et al. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". In: *CoRR* abs/1502.01852 (2015). URL: http://arxiv.org/abs/1502.01852.

[23] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *CoRR* abs/1412.6980 (2014). arXiv: 1412.6980. URL: http://arxiv.org/abs/1412.6980.

[24] Yoshua Bengio. "Practical recommendations for gradient-based training of deep architectures". In: *CoRR* abs/1206.5533 (2012). arXiv: 1206.5533. URL: http://arxiv.org/abs/1206.5533.

[25] Leslie N. Smith. "No More Pesky Learning Rate Guessing Games". In: *CoRR* abs/1506.01186 (2015). arXiv: 1506.01186. URL: http://arxiv.org/abs/1506.01186.

[26] Leslie N. Smith. "A disciplined approach to neural network hyper-parameters: Part 1 - learning rate, batch size, momentum, and weight decay". In: *CoRR* abs/1803.09820 (2018). arXiv: 1803.09820. URL: http://arxiv.org/abs/1803.09820.

[27] Ilya Loshchilov and Frank Hutter. "SGDR: Stochastic Gradient Descent with Restarts". In: *CoRR* abs/1608.03983 (2016). arXiv: 1608.03983. URL: http://arxiv.org/abs/1608.03983.

[28] Leslie N. Smith and Nicholay Topin. "Super-Convergence: Very Fast Training of Residual Networks Using Large Learning Rates". In: *CoRR* abs/1708.07120 (2017). arXiv: 1708.07120. URL: http://arxiv.org/abs/1708.07120.

[29] Günter Klambauer et al. "Self-Normalizing Neural Networks". In: *CoRR* abs/1706.02515 (2017). URL: http://arxiv.org/abs/1706.02515.

[30] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Stanford University, Tech. Rep.* (2015). URL: http://jmlr.org/proceedings/papers/v37/ioffe15.pdf.

[31] Yann LeCun et al. "Efficient Backprop". In: (1998). URL: http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf.

[32] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. "Searching for Activation Functions". In: *CoRR* abs/1710.05941 (2017). arXiv: 1710.05941. URL: http://arxiv.org/abs/1710.05941.

[33] Gao Huang et al. "Snapshot Ensembles: Train 1, get M for free". In: *CoRR* abs/1704.00109 (2017). arXiv: 1704.00109. URL: http://arxiv.org/abs/1704.00109.

[34] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.* 2012, pp. 1106–1114. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.

[35] S. Chatrchyan et al. "The CMS Experiment at the CERN LHC". In: *JINST* 3 (2008), S08004. DOI: 10.1088/1748-0221/3/08/S08004.

[36] T. Garipov et al. "Loss Surfaces, Mode Connectivity, and Fast Ensembling of DNNs". In: *ArXiv e-prints* (Feb. 2018). arXiv: 1802.10026 [stat.ML].

[37] F. Draxler et al. "Essentially No Barriers in Neural Network Energy Landscape". In: *ArXiv e-prints* (Mar. 2018). arXiv: 1803.00885 [stat.ML].

[38] Pavel Izmailov et al. "Averaging Weights Leads to Wider Optima and Better Generalization". In: *CoRR* abs/1803.05407 (2018). arXiv: 1803.05407. URL: http://arxiv.org/abs/1803.05407.

[39] Mark A. Beaumont, Wenyang Zhang, and David J. Balding. "Approximate Bayesian computation in population genetics". In: *Genetics* 162.4 (2002), pp. 2025–2035.

[40] Simon N. Wood. "Statistical inference for noisy nonlinear ecological dynamic systems". In: *Nature* 466.7310 (2010), p. 1102.

[41] Kyle Cranmer, Juan Pavez, and Gilles Louppe. "Approximating likelihood ratios with calibrated discriminative classifiers". In: *arXiv preprint arXiv:1506.02169* (2015).

[42] J. Neyman and E. S. Pearson. "On the Problem of the Most Efficient Tests of Statistical Hypotheses". In: *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character* 231 (1933), pp. 289–337. ISSN: 02643952. URL: http://www.jstor.org/stable/91247.

[43] Claire Adam-Bourdarios et al. "The Higgs boson machine learning challenge". In: *NIPS 2014 Workshop on High-energy Physics and Machine Learning*. 2015, pp. 19–55.

[44] D Basu. "On partial sufficiency: A review". In: *Selected Works of Debabrata Basu*. Springer, 2011, pp. 291–303.

[45] David A Sprott. "Marginal and conditional sufficiency". In: *Biometrika* 62.3 (1975), pp. 599–605.

[46] Glen Cowan et al. "Asymptotic formulae for likelihood-based tests of new physics". In: *The European Physical Journal C* 71.2 (2011), p. 1554.

[47] R. A. Fisher. "Theory of Statistical Estimation". In: *Mathematical Proceedings of the Cambridge Philosophical Society* 22.5 (1925), 700–725. DOI: 10.1017/S0305004100009580.

[48] Harald Cramér. *Mathematical methods of statistics (PMS-9)*. Vol. 9. Princeton university press, 2016.

[49] C. Radhakrishna Rao. "Information and the accuracy attainable in the estimation of statistical parameters". In: *Breakthroughs in statistics*. Springer, 1992, pp. 235–247.

[50] Pierre Simon Laplace. "Memoir on the probability of the causes of events". In: *Statistical Science* 1.3 (1986), pp. 364–378.

[51] Dustin Tran et al. "Edward: A library for probabilistic modeling, inference, and criticism". In: *arXiv preprint arXiv:1610.09787* (2016).

[52] A Hocker et al. "TMVA—Toolkit for Multivariate Data Analysis, in proceedings of 11th International Workshop on Advanced Computing and Analysis Techniques in Physics Research". In: *Amsterdam, The Netherlands* (2007).

[53] Pierre Baldi, Peter Sadowski, and Daniel Whiteson. "Searching for exotic particles in high-energy physics with deep learning". In: *Nature communications* 5 (2014), p. 4308.

[54] Radford Neal et al. "Computing likelihood functions for high-energy physics experiments when distributions are defined by simulators with nuisance parameters". In: (2008).

[55] Pierre Baldi et al. "Parameterized neural networks for high-energy physics". In: *The European Physical Journal C* 76.5 (2016), p. 235.

[56] Johann Brehmer et al. "Mining gold from implicit models to improve likelihood-free inference". In: (2018). arXiv: 1805.12244 [stat.ML].

[57] Johann Brehmer et al. "Constraining Effective Field Theories with Machine Learning". In: *arXiv preprint arXiv:1805.00013* (2018).

[58] Johann Brehmer et al. "A Guide to Constraining Effective Field Theories with Machine Learning". In: *arXiv preprint arXiv:1805.00020* (2018).

[59] Bai Jiang et al. "Learning summary statistic for approximate Bayesian computation via deep neural network". In: *arXiv preprint arXiv:1510.02175* (2015).

[60] Gilles Louppe, Michael Kagan, and Kyle Cranmer. "Learning to Pivot with Adversarial Networks". In: *Advances in Neural Information Processing Systems*. 2017, pp. 982–991.

[61] Pablo de Castro. *Code and manuscript for the paper "INFERNO: Inference-Aware Neural Optimisation"*. https://github.com/pablodecm/paper-inferno. 2018.

[62] Martin Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[63] Joshua V Dillon et al. "TensorFlow Distributions". In: *arXiv preprint arXiv:1711.10604* (2017).

[64] Roger Barlow. "Extended maximum likelihood". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 297.3 (1990), pp. 496–506.

[65] Morad Aaboud et al. "Observation of $H \to b\bar{b}$ decays and VH production with the ATLAS detector". In: *Phys. Lett.* B786 (2018), pp. 59–86. DOI: 10.1016/j.physletb.2018.09.013. arXiv: 1808.08238 [hep-ex].

[66] A. M. Sirunyan et al. "Observation of Higgs boson decay to bottom quarks". In: *Phys. Rev. Lett.* 121.12 (2018), p. 121801. DOI: 10.1103/PhysRevLett.121.121801. arXiv: 1808.08242 [hep-ex].

[67] Georges Aad et al. "Search for the Standard Model Higgs boson decaying into $b\bar{b}$ produced in association with top quarks decaying hadronically in pp collisions at $\sqrt{s} = 8$ TeV with the ATLAS detector". In: *JHEP* 05 (2016), p. 160. DOI: 10.1007/JHEP05(2016)160. arXiv: 1604.03812 [hep-ex].

[68] Olaf Nackenhorst et al. "Search for the Standard Model Higgs boson produced in association with tt and decaying into bb at 8 TeV with the ATLAS detector using the Matrix Element Method". Presented 08 Jun 2015. PhD thesis. Gottingen U., 2015. URL: https://cds.cern.ch/record/2063972.

[69] Morad Aaboud et al. "Search for the standard model Higgs boson produced in association with top quarks and decaying into a $b\bar{b}$ pair in pp collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector". In: *Phys. Rev.* D97.7 (2018), p. 072016. DOI: 10.1103/PhysRevD.97.072016. arXiv: 1712.08895 [hep-ex].

[70] Georges Aad et al. "Search for the Standard Model Higgs boson produced in association with top quarks and decaying into $b\bar{b}$ in pp collisions at $\sqrt{s} = 8$ TeV with the ATLAS detector". In: *Eur. Phys. J.* C75.7 (2015), p. 349. DOI: 10.1140/epjc/s10052-015-3543-1. arXiv: 1503.05066 [hep-ex].

[71] M. Aaboud et al. "Observation of Higgs boson production in association with a top quark pair at the LHC with the ATLAS detector". In: *Phys. Lett.* B784 (2018), pp. 173–191. DOI: 10.1016/j.physletb.2018.07.035. arXiv: 1806.00425 [hep-ex].

[72] Albert M Sirunyan et al. "Observation of $t\bar{t}H$ production". In: *Phys. Rev. Lett.* 120.23 (2018), p. 231801. DOI: 10.1103/PhysRevLett.120.231801,10.1130/PhysRevLett.120.231801. arXiv: 1804.02610 [hep-ex].

[73] Doug Schouten, Adam DeAbreu, and Bernd Stelzer. "Accelerated Matrix Element Method with Parallel Computing". In: *Comput. Phys. Commun.* 192 (2015), pp. 54–59. DOI: 10.1016/j.cpc.2015.02.020. arXiv: 1407.7595 [physics.comp-ph].

[74] G. P. Lepage. "A New Algorithm for Adaptive Multidimensional Integration". In: *Journal of Computational Physics* 27 (May 1978), p. 192. DOI: 10.1016/0021-9991(78)90004-9.

[75] Andy Buckley et al. "LHAPDF6: parton density access in the LHC precision era". In: *Eur. Phys. J.* C75 (2015), p. 132. DOI: 10.1140/epjc/s10052-015-3318-8. arXiv: 1412.7420 [hep-ph].

[76] J. Alwall et al. "The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations". In: *JHEP* 07 (2014), p. 079. DOI: 10.1007/JHEP07(2014)079. arXiv: 1405.0301 [hep-ph].

[77] John Nickolls et al. "Scalable Parallel Programming with CUDA". In: *Queue* 6.2 (Mar. 2008), pp. 40–53. ISSN: 1542-7730. DOI: 10.1145/1365490.1365500. URL: http://doi.acm.org/10.1145/1365490.1365500.

[78] John E. Stone, David Gohara, and Guochun Shi. "OpenCL: A Parallel Programming Standard for Heterogeneous Computing Systems". In: *IEEE Des. Test* 12.3 (May 2010), pp. 66–73. ISSN: 0740-7475. DOI: 10.1109/MCSE.2010.69. URL: http://dx.doi.org/10.1109/MCSE.2010.69.

[79] M. Oreglia. "A Study of the Reactions $\psi' \to \gamma\gamma\psi$". PhD thesis. SLAC, 1980. URL: http://www-public.slac.stanford.edu/sciDoc/docMeta.aspx?slacPubNumber=slac-r-236.html.

[80] Tomasz Skwarnicki. "A study of the radiative CASCADE transitions between the Upsilon-Prime and Upsilon resonances". PhD thesis. Cracow, INP, 1986. URL: http://www-library.desy.de/cgi-bin/showprep.pl?DESY-F31-86-02.

[81] Victor Estrade et al. "Adversarial learning to eliminate systematic errors: a case study in High Energy Physics". In: *Deep Learning for Physical Sciences workshop@ NIPS*. 2017.

[82] François Chollet. "Keras". In: *GitHub* (2016). URL: https://github.com/fchollet/keras.

[83] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: http://tensorflow.org/.

[84] M. Brucher et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[85] J. D. Hunter. “Matplotlib: A 2D graphics environment”. In: *Computing In Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[86] Tom Augspurger et al. *seaborn: v0.7.1 (June 2016)*. June 2016. DOI: 10.5281/zenodo.54844. URL: https://doi.org/10.5281/zenodo.54844.

[87] S. van der Walt, S. C. Colbert, and G. Varoquaux. “The NumPy Array: A Structure for Efficient Numerical Computation”. In: *Computing in Science Engineering* 13.2 (2011), pp. 22–30. ISSN: 1521-9615. DOI: 10.1109/MCSE.2011.37.

[88] Wes McKinney. “Data Structures for Statistical Computing in Python”. In: *Proceedings of the 9th Python in Science Conference*. Ed. by Stéfan van der Walt and Jarrod Millman. 2010, pp. 51 –56.