Improving hadronic tau decay mode reconstruction at ATLAS using neural networks

Hoang Nguyen

Masterarbeit in Physik
angefertigt im Physikalischen Institut

vorgelegt der
Mathematisch-Naturwissenschaftlichen Fakultät
der
Rheinischen Friedrich-Wilhelms-Universität
Bonn

September 2020

I hereby declare that this thesis was formulated by myself and that no sources or tools other than those cited were used.

Bonn, ...
Date Signature

1. Gutachter: Priv.-Doz. Dr. Philip Bechtle
2. Gutachterin: Prof. Dr. Jochen Dingfelder

Contents

CHAPTER 1

Introduction

Physics is science which means that one tries to understand the laws of nature by experimental means. One big, fundamental question which is attempted to be answered is: What is the world made of?
To achieve this, the Large Hadron Collider (LHC), the world's largest particle accelerator and collider, was built and it started operating in 2009. With a center-of-mass energy of about 7 TeV in Run I, the four main experiments ATLAS, CMS, ALICE and LHCb could obtain data to probe and maybe confirm the Standard Model of Particle Physics, the theory which summarizes the current knowledge and understanding of Particle Physics. In the year 2012, a major achievement was the discovery of a particle predicted by the Standard Model, the Higgs boson. With the second round of data taking in Run II, the center-of-mass energy could be improved to 13 TeV, resulting in an increase of data to be analyzed.
The tau lepton plays an important role in physics analyses at the LHC. Because of its large mass, it is an effective probe of the fermionic coupling of the Higgs boson. Moreover, it is essential in finding evidence for new physics effects which cannot be explained by the Standard Model. However, the identification of the tau lepton is not a trivial task. While its leptonic decay channel can be recognized with no problems, its hadronic decays, producing collimated jets of particles, are covered by other jets from other sources which can therefore mimic hadronic tau decays. Furthermore, the tau lepton can only be detected by said decay products which requires a good understanding of the substructure of hadronic tau decays. For this, the ATLAS experiment already uses an algorithm called `PanTau` which is able to classify the various decay modes based on the detected particles. However, misclassification still occurs which means that improvements are still possible. Latest studies indicate that a novel approach with recurrent neural networks (RNN), algorithms which are inspired by human neural networks, yields better results than `PanTau`. Although these algorithms are not new, they could not be used to great success until now since the required computational power, which is essential for these algorithms, was not available.
This thesis attempts to describe how such a neural network can be used for tau decay mode classification. Furthermore, it is investigated if the aforementioned RNN can be optimized so that its efficiency increases even more. Last, but not least, it is examined if said network can also be used for practical use cases.
The structure of the thesis is as follows:
The theoretical background and motivation is presented in Chapter 2. In the following chapter, the ATLAS detector and the method of data-taking is described. Chapter 4 introduces the concept of

artificial neural networks and Chapter 5 concludes this by presenting and discussing the structure of the recurrent neural network studied in this work. Chapter 6 shows how it was attempted to improve tau decay mode classification, done by the RNN, while Chapter 7 and Chapter 8 address the used data and the questions regarding the real usability of this algorithm for this task. In Chapter 9, the thesis is concluded and an future prospect is given.

CHAPTER 2

Theoretical foundations

2.1 The Standard Model of Particle Physics

The Standard Model of Particle Physics (SM) is the current theory which includes all knowledge about the known elementary particles and their interactions among each other. Since its formulation, it has been tested by a large number of experiments and, until now, all results did not deviate significantly from the predictions. The main components are summarized in Figure 2.1.

According to it, matter is built from particles with half-spin, so-called fermions. These are further divided into quarks, the constituents of hadrons like protons or neutrons, and leptons. They obey Fermi-Dirac statistics as well as the Pauli exclusion principle. Fermions are categorized into three generations. All generations are identical except their masses which increase with each generation. There is always a quark with a fractional charge of +2/3 and -1/3 respectively, charged leptons with a charge of +1 and neutral leptons. The first generation of quarks and leptons, i.e. up quark, down quark, electron and electron neutrino, build all known matter whereas the other ones can only be observed in high-energy interactions. These particles eventually decay into first generation particles. Additionally, for each fermion, there exists a corresponding antiparticle. These antiparticles possess the same properties except for their opposite conserved quantum numbers.

The Standard Model also includes four gauge bosons. All of them have spin 0, obey Bose-Einstein statistics and act as exchange particles for the fundamental forces.
The electromagnetic force is mediated by the photon. It is electrically neutral and massless. The latter also explains the long-range effect of this force.
Gluons act as mediators for the strong force. Like the photon, they are electrically neutral and massless. Furthermore, they carry an additional (anti-)colour charge. With this, they can interact with quarks and also among each other, explaining the non-infinite range of the force.
In contrast to the two mentioned bosons, the remaining gauge bosons, the $W^{\pm}$ and the Z^0 boson, are massive, which means they possess a non-zero mass. Their rest mass is (80.379 ± 0.012)GeV and (91.1876 ± 0.0021) GeV respectively [2]. Both act as mediators for the weak interaction. As indicated by the superscript, $W^{\pm}$ bosons are electrically charged with $Q(W^+) = +1e$ and $Q(W^-) = -1e$. They specifically mediate weak charged current interactions which also allows flavour changes for quarks. Additionally, they couple to chiral left-handed particles and chiral right-handed antiparticles, maximally violating parity symmetry. The Z^0 boson has no electric charge and acts as a mediator for

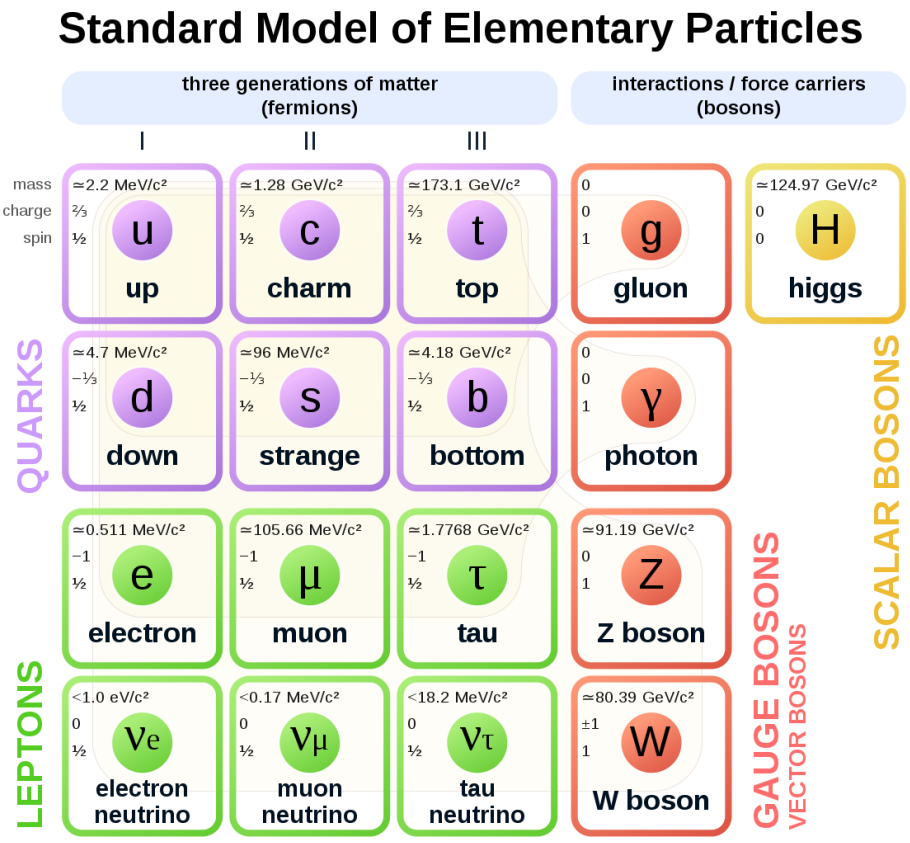


Figure 2.1: Particles of the Standard Model with their respective properties [1]

weak neutral current interactions. For this work, their behaviour to couple to $\tau^+\tau^-$ pairs is important and has been exploited during the studies.

The last elementary particle which is included in the Standard Model is the Higgs boson H which is a scalar boson. It is created by spontaneous symmetry breaking in the electroweak theory (see 2.1.1). The coupling of the Higgs boson to fermions is called Yukawa coupling. Its strength is proportional to the mass of the fermion. This boson has a rest mass of (125.10 ± 0.14) GeV [2].

2.1.1 Unification of the electromagnetic and weak interaction

The theory of weak interaction is based on the $SU(2)_L$ group. However, this theory is not self-consistent since requiring gauge invariance under $SU(2)_L$ transformation results in two charged and one neutral gauge bosons. Contrary to this, the observed weak neutral current also has a right-handed component. Glashow, Salam and Weinberg developed a theory which solved this problem by unifying electromagnetic and weak interaction to the electroweak interaction [3]. It is based on the $SU(2)_L \times U(1)_Y$ gauge group (with $Y = 2(Q - T_3)$ and Q = electric charge and T_3 = third component of weak isopin). The local gauge invariance leads to the existence of four gauge bosons: W_1, W_2, W_3 from $SU(2)_L$ and B from $U(1)_Y$. In contrast to the reality, these bosons would be massless. To give

the gauge bosons mass, an scalar isospin doublet Φ is introduced as

$$\Phi = \begin{pmatrix} \Phi^+ \\ \Phi^0 \end{pmatrix} \tag{2.1}$$

with the field potential $V(\Phi) = \mu^2 \Phi^+ \Phi + \lambda(\Phi^+ \Phi)^2$.
With $\mu < 0$ and $\lambda > 0$, this potential is minimized by a set of infinitely many possible solutions for Φ with non-vanishing expectation value. The necessity to choose one unique solution Φ_0 breaks the group $SU(2)_L \times U(1)_Y$ down to $U(1)_Q$. This is called *sponatenous symmetry breaking* and leads to the $W^\pm$ and the Z boson having mass while the photon stays massless. Furthermore, this Higgs mechanism predicts a neutral, massive boson with zero spin which is the Higgs boson H. Similarly, the Higgs mechanism can be used to explain the masses of fermions.

2.1.2 Problems with the Standard Model

Although the Standard Model has enourmous success in describing microscopic processes, it still has some shortcomings which will be discussed now. The following list shall not be seen as complete by any means though.

First, only three of the four fundamental forces are included in the Standard Model. Although often attempted, an extension so that gravitational interactions between massive particles can be described has not been found until today. The current theory describing this interaction, the theory of General Relativity, does not seem to be mathematically compatible to the Standard Model, which is based on Quantum Field Theory (QFT), either.

Moreover, it is implied by cosmological observations that there is a large fraction of matter in our universe which only interacts gravitationally. For this *dark matter*, the Standard Model does not contain particles having the right properties to be candidates. In addition to this, it is theorized that the presence of *dark energy* is needed to explain effects like, for example, the accelerated expansion of the universe. For this, no candidate particles have been found either.

Another shortcoming is called the *hierarchy problem*. It describes how the huge discrepancy between the gravitational and the electroweak force, which is of the order of 32 orders of magnitude, can lead to a problem in the theoretical treatment of the Higgs mass. For high energy scales, the Higgs boson mass would reach high values due to radiative corrections from higher order contribution in perturbative theory. Since the observed mass is about 125 GeV, fine-tuning the contributions to the Higgs mass would be needed to keep it at electroweak scale.

Besides, neutrinos in the Standard Model are said to be massless. However, neutrino oscillation experiments have proven that they do have mass. This cannot be explained by the Standard Model.

One last phenomena which cannot be explained is the asymmetry of matter and antimatter. The Standard Model predicts that both should have been created in equal amounts. However, this would contradict the fact that the universe is made out of mostly matter.

2.2 The tau lepton

Since the tau lepton is the study object of this work, it will be introduced in this section.
Although already anticipated in 1960, the tau lepton was discovered in 1975 at the Stanford Linear Accelerator Center (SLAC)[4]. It has a mass of (1776.86 ± 0.12)MeV [2] and is the heaviest lepton in

the Standard Model. Because of this mass, it is rather shortlived with an average lifetime of about 290 ps.

2.2.1 Decay

The tau lepton decays via the weak charged current interaction. Due to its high mass, it is able to decay leptonically and hadronically which distinguishes it as being the only lepton which can decay like this. The main decay modes can be seen in Figure 2.2.

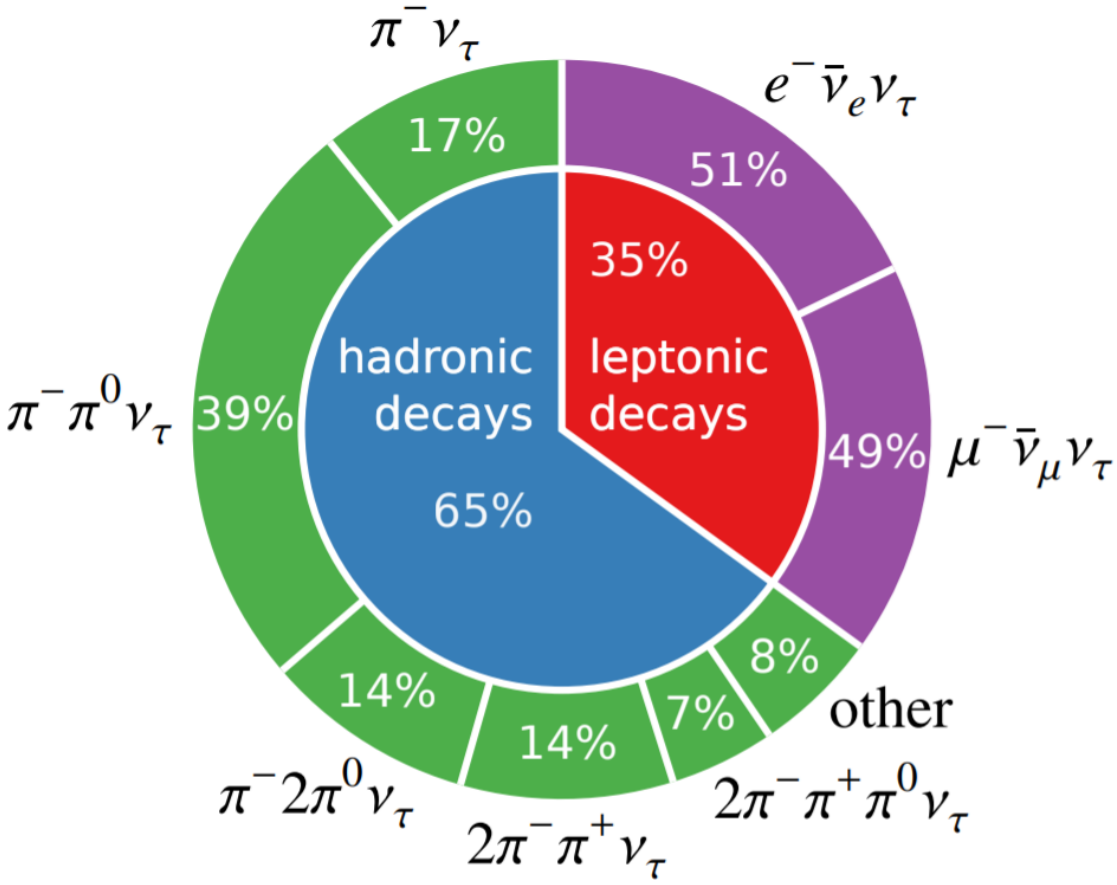


Figure 2.2: Main decay modes of the tau lepton; the specific decay mode percentages are given with respect to all hadronic (leptonic) decay modes

The leptonic decay modes make up about 35 % of all tau decays. Due to lepton universality, the branching ratio of both leptonic decays are almost the same. For this work, the hadronic decays which make up approximately 65 % of all decays are more important. The Feynman diagram, describing this physical process, can be seen in Figure 2.3.

As can be seen, strange quarks can occur during this decay process as well resulting in a kaon. However, this decay is suppressed so that kaons will not be produced frequently. At last, the abbreviations regarding a tau lepton decay are introduced. They can be seen in Figure 2.1.

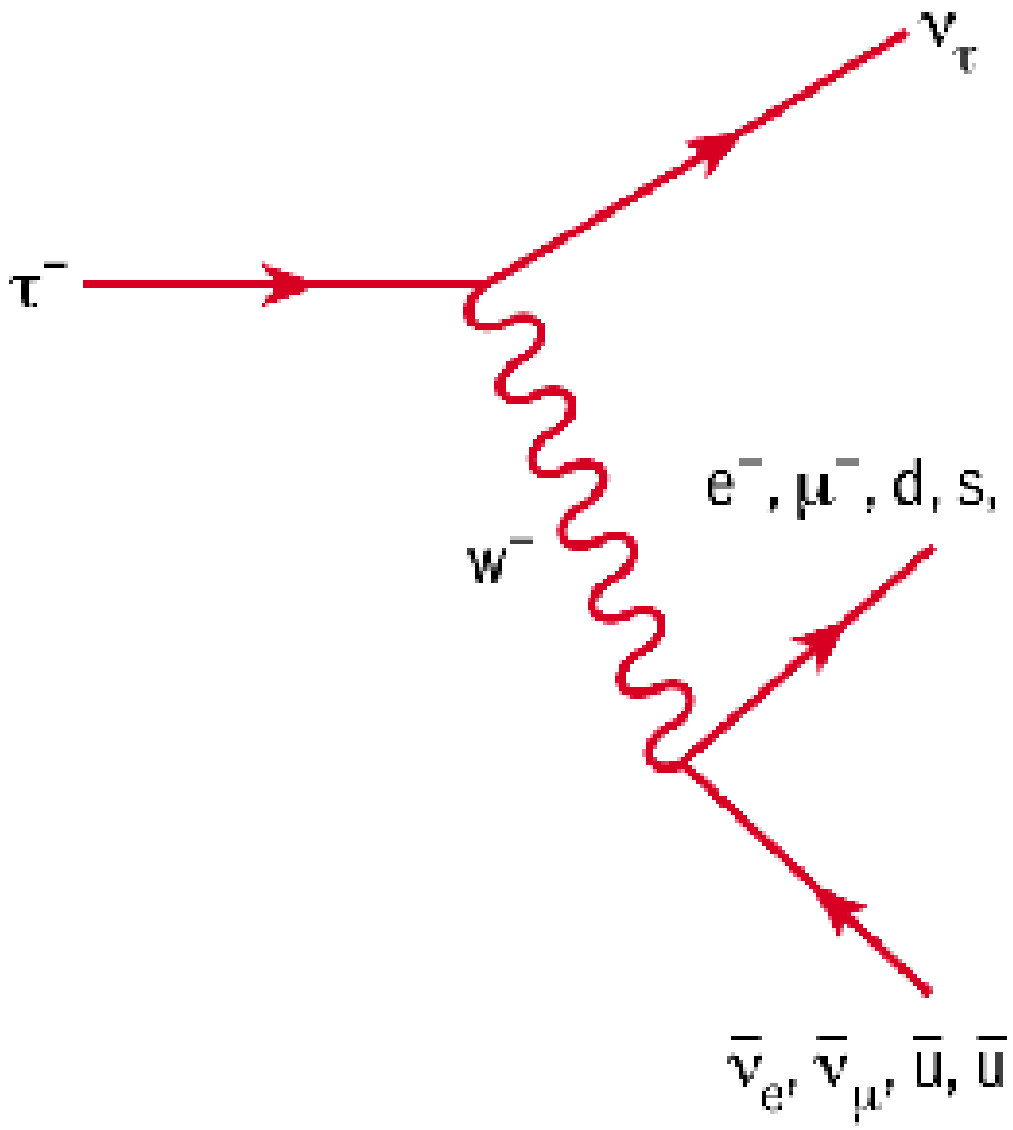


Figure 2.3: Feynman diagram of the tau decay [5]

Decay mode	also denoted as
$\tau^- \rightarrow h^- \nu_\tau$	1p0n
$\tau^- \rightarrow h^- \pi^0 \nu_\tau$	1p1n
$\tau^- \rightarrow h^- \geq 2\pi^0 \nu_\tau$	1pXn
$\tau^- \rightarrow h^- h^+ h^- \nu_\tau$	3p0n
$\tau^- \rightarrow h^- h^+ h^- \geq 1\pi^0 \nu_\tau$	3pXn

Table 2.1: List of the five most important hadronic tau decay modes and their shorthand

2.2.2 Tau decay mode research

The decay mode of tau leptons has an important role for many physics analyses at the LHC. It is obvious that with better understanding of this topic, the identification of tau leptons itself can be improved. Furthermore, with this, jet background can be suppressed. However, the knowledge of the decay mode can also contribute to possible breakthroughs in other areas. One example is using the tau decay as a spin-analyzer in $Z \rightarrow \tau\tau$ processes. Moreover, the polarisation of the tau lepton could be exploited to reduce the $Z \rightarrow \tau\tau$ background in measurements and searches of neutral spin-0 resonances in the ditau channel.
Tau final states also play a role in the search of new physics. An example is the search for Physics Beyond Standard Model (BSM) Higgs bosons. In a Standard Model extension called Minimal Supersymmetric Standard Model (MSSM), three neutral Higgs bosons are predicted which decay into two taus with a chance of 10 % on a mass range up to the TeV scale. Here, tau leptons can serve as an important probe to constrain the parameter space of Supersymmetry models.

CHAPTER 3

The ATLAS experiment at the LHC

The data which is used in this work is obtained by the ATLAS detector at CERN. Furthermore, also Monte Carlo simulated data will be used which also mimics the just mentioned experimental data. Because of this, this chapter will give an short overview about the LHC and the ATLAS experiment.

3.1 The Large Hadron Collider

The Large Hadron Collider (LHC) is a circular hadron collider which can be found at CERN, short for *Conseil européen pour la recherche nucléaire*, near Geneva. It is currently the largest particle physics experiment in the world with a circumference of approximately 27 km. It is designed to collide proton beams with a center-of-mass energy of 14 TeV and a peak luminosity of 10^{34}cm^{-2} s as well as lead ions with an energy of 2.8 TeV per nucleon and a peak luminosity of 10^{27}cm^{-2} s.
Before the proton beams can enter the LHC, they need to run through a pre-accelerator system which is depicted in Figure 3.1. After the extraction by ionisation from a hydrogen source, the protons are accelerated to 50 MeV by the linear accelerator LINAC2. After that they enter the *Proton Synchrotron Booster* (PSB) where they gain an energy of 1.4 GeV. Next, they are injected in the *Proton Synchroton* (PS) which the protons leave with an energy of 25 GeV. Afterwards, the protons are fed in as packages in the *Super Proton Synchrotron* (SPS) where they are ramped up to 450 GeV. Finally they are split into two beams and enter the LHC via two transfer tunnels.

In the LHC, four interactions points are defined where the particle beams collide. At these points, a particle detector is built respectively: ALICE, ATLAS, CMS and LHCb. Each of those is specialized for different purposes. ALICE mainly studies the collision of heavy ions to widen the knowledge of fundamental properties of the strong force and to probe conditions comparable to the early universe. The LHCb experiment is specialized to detect small asymmetries between matter and antimatter in the decay of B mesons. At last, ATLAS and CMS are multipurpose detectors to study a broad range of high-energy processes of particles.
A very high amount of particle collisions is needed for these experiments which can be calculated with $N_{\text{event}} = \mathscr{L}\sigma_{\text{event}}$. The cross section σ_{event} can be interpreted as a probability for a process to happen while $\mathscr{L}$ denotes the luminosity which describes the amount of events per time and area and depends on machine parameters of the accelerator. During operation in 2018, the ATLAS Detector measured a peak instantaneous luminosity of $L = 21 \times 10^{33}\ \text{cm}^{-2}\ \text{s}$

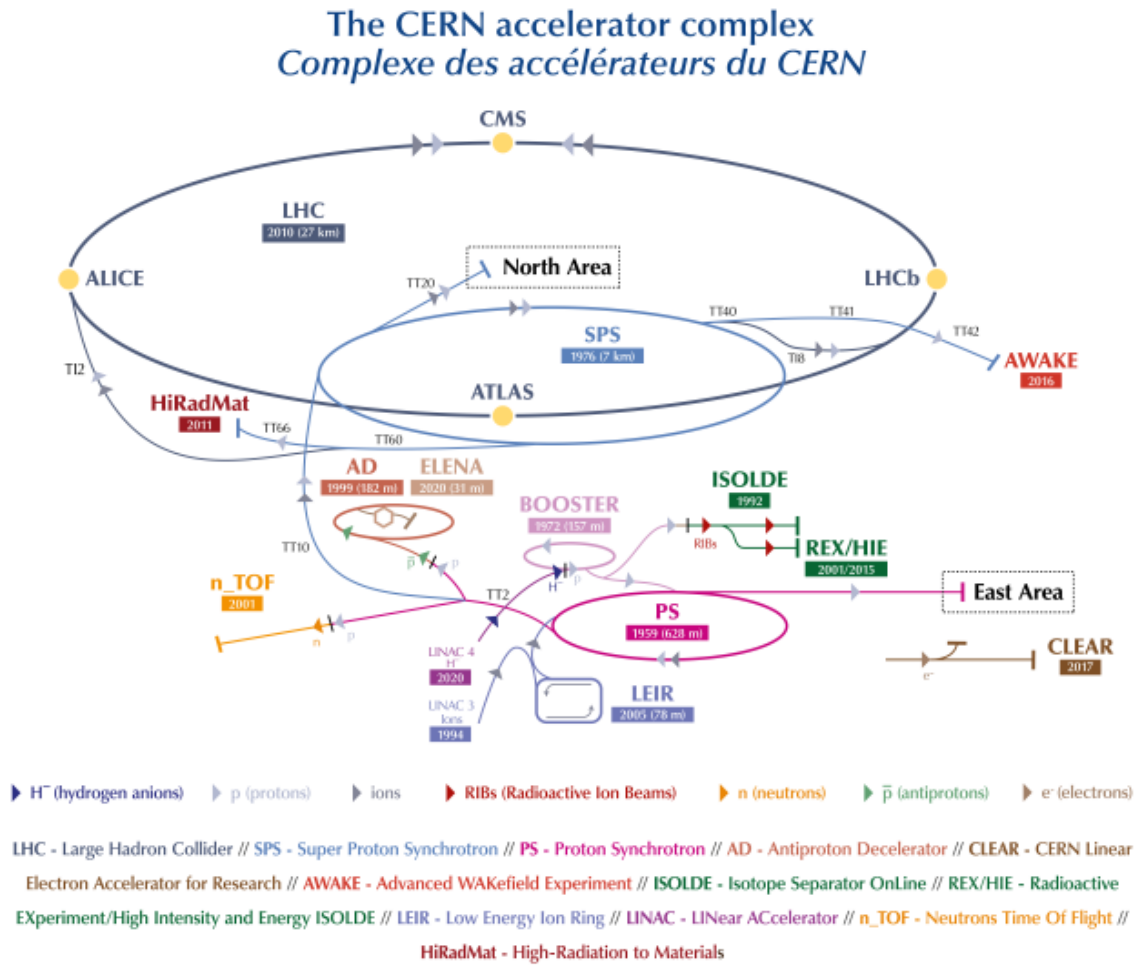


Figure 3.1: Schematic overview of the accelerator complex at CERN [6]

3.2 The ATLAS detector

ATLAS (A Toroidal LHC ApparatuS) is one of two general purpose experiments at the LHC. The high luminosity provided by the LHC and high cross sections due to the high energy enable further precision tests of the Standard Model as well as searches for physics beyond the Standard Model. Several benchmark searches have been used to establish the performance of the ATLAS detector. The search for the Standard Model Higgs boson is particularly important since a broad range of production and decay mechanisms is covered by ATLAS. Many theories beyond the Standard Model predict new particles with masses in the TeV-range and small production cross sections. These rare processes have to be distinguished from dominating inelastic proton-proton interactions and QCD jet production.
An overview of the detector can be seen in Figure 3.2. Because of its cylindrical geometry, the detector can almost cover 4π and has forward-backward symmetry with respect to the nominal interaction point. Around this interaction point, detector subsystems, each specialized to detect specific particles, are arranged in concentric layers around the beam axis.
A right-handed coordinate system is used to describe positions in the detector. The z-axis points along the beam axis. The x-axis towards the centre of the LHC ring and the y-axis points upwards, spanning the transverse plane. Spherical coordinates (r, ϕ, θ) can also be used to describe the positions. The azimuthal angle ϕ is measured in the transverse plane while the polar angle θ with respect to the beam axis.

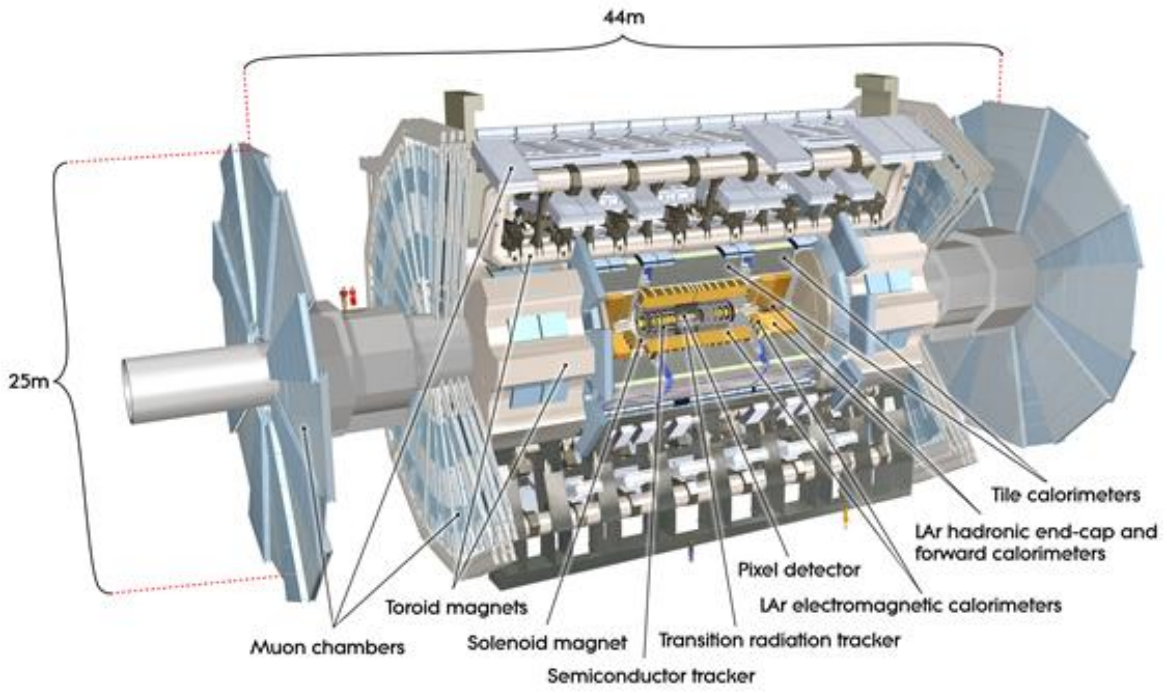


Figure 3.2: Overview of the ATLAS detector [7]

The momentum and energy can only be quantified in the transverse plane since the longitudinal

momentum of colliding partons is unknown. Therefore, they are defined as

$$p_t = |p|sin(\theta) \qquad E_T = Esin(\theta) \tag{3.1}$$

In addition to this, the unknown boost along the x-axis motivates the definition of the pseudorapidity and from this the angular distance can be defined:

$$\eta = -ln\left(tan\left(\frac{\theta}{2}\right)\right) \qquad \Delta R = \sqrt{(\Delta\eta)^2 + (\Delta\phi)^2} \tag{3.2}$$

$\Delta\eta$ and $\Delta\phi$ are the differences in pseudorapidity and azimuthal angle respectively.
In the following, a short description of the detector subsystems will be given. Further, more detailed information is provided in [8].

3.2.1 The Inner Detector

The Inner Detector is approximately 6.2 m long and has a diameter of about 2.1 m. It is situated the closest around the interaction point. It consists of three parts starting from the interaction points: the Pixel Detector, the Silicon Tracker and the Transition Radiation Tracker.

Pixel Detector The Pixel Detector is built such that a particle traverses three silicon sensors independent of its pseudorapidity. This is realized by a barrel and an end-cap region. In the barrel region three layers are built cylindrically around the beam axis while in the end-cap regions the detector modules are three discs perpendicular to the beam axis. It reaches a resolution of about 10 µm in the r-ϕ-plane and covers the range $|\eta| < 2.5$.

Silicon tracker The Silicon Tracker is built around the Pixel Detector. It is made of silicon strips allowing a resolution of 17 µm in the r-ϕ-plane and 580 µm in the z-direction. The range covered is $|\eta| < 2.5$.

Transition Radiation Tracker Track measurement at large radial ranges are provided by the TRT which consists of drift tubes with a diameter of 4 mm aligned with the beam axis. They only provide azimuthal measurements with a resolution of 130 µm and the pseudorapidity range which is covered is $|\eta| < 2.0$.

3.2.2 The Calorimeter system

The purpose of the calorimeters is to measure the energies of the particles. Therefore, it is required that the particles deposit their full energy within the calorimeters. The calorimeters are designed in a way that an incoming particle induces a shower within the calorimeter material. There are two types of calorimeters in the inner detector: electromagnetic and hadronic calorimeters.

Electromagnetic calorimeter (ECAL) The Electromagnetic calorimeter is divided into a barrel and two endcaps, covering the regions $|\eta| < 1.475$ and $1.375 < |\eta| < 3.2$. It consists of a presampling layers which measures the energy loss before the ECAL and three longitudinal samplings called EM1, EM2 and EM3. EM1 has a high granularity in order to make high precision measurements of photons. In EM2, the main energy of the electromagnetic showers

are deposited while EM3 measures the rest energy of very high energetc showers. In here, two important quantities called *photon shots* and *conversion tracks* are measured. Photon shots denote highly energized photons which were created from a neutral pion decay and which could be reconstructed individually. Conversion tracks are tracks of photons undergoing pair production.

Hadronic Calorimeter (HCAL) The hadronic calorimeter measures the energy of hadrons which passed the ECAL. It is divided into a barrel ($|\eta| < 1.7$ and an end-cap region ($1.5 < |\eta| < 3.2$). Both use different detector setups to measure the energy of particles.

3.2.3 The Muon System

Muons do not interact by the strong force and due to their high mass, they do not lose as much energy as, for example, electrons via bremsstrahlung. That is why they are often able to pass the ECAL and the HCAL. To detect them, an additional muon chamber as the outermost part completes the ATLAS detector. It covers the range $|\eta| < 2.7$ and is basically a tracking detector which measures the bending of muons in a magnetic field to get information about their momenta.

3.2.4 The Trigger System

Given the event rate of 40 MHz and, therefore, the high data amount which the collisions provide, an efficient trigger system is required to reduce it to a storable size. The trigger system of ATLAS consists of two levels using both, hardware and software, algorithms to only preserve interesting events for physics analyses. A schematic overview can be seen in Figure 3.3.
The *Level 1 Trigger* is fully hardware based and works on information provided from the calorimeters and the muon detectors. The data rate is reduced to 100 kHz with a latency of 2.5 µs. Furthermore, region of interests are determined.
The chosen region of interests are sent to the *High Level Trigger* (HLT) which uses selection algorithms using full granularity detector information. Thus, the data rate is further reduced to 1 kHz with an average processing time of 200 ms.

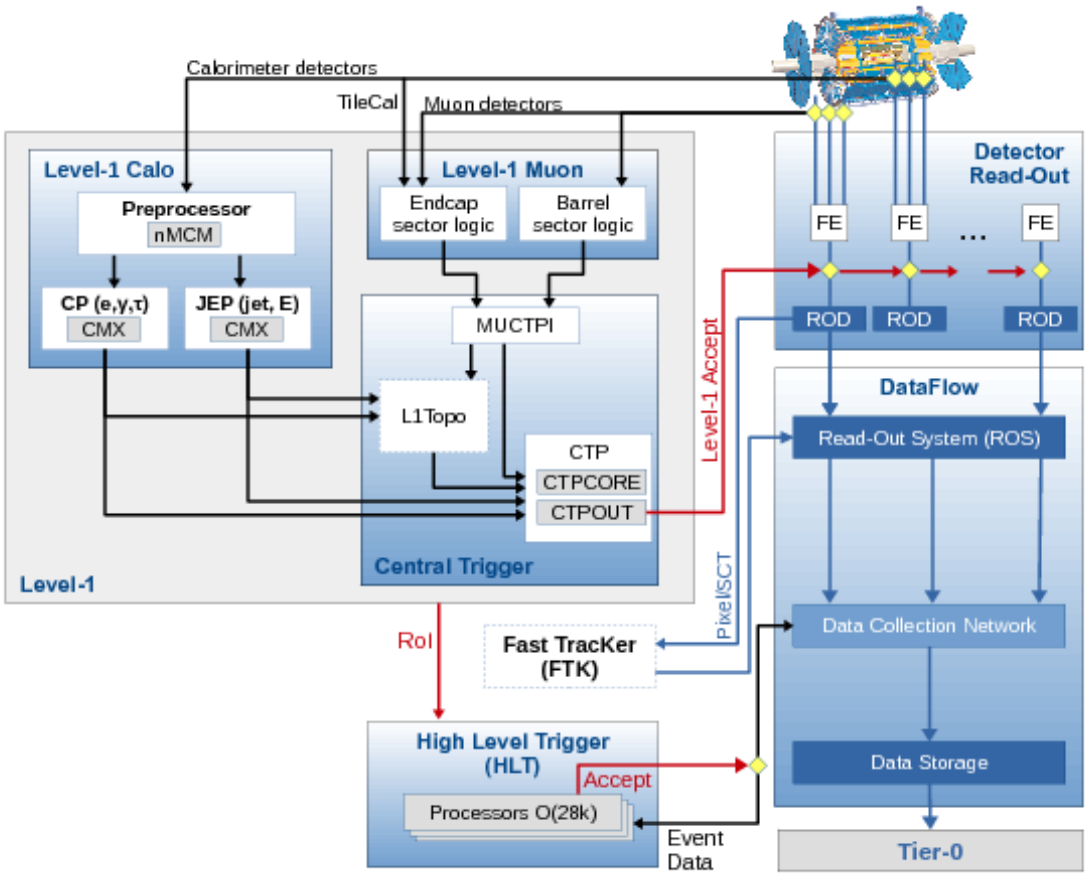


Figure 3.3: Overview of the trigger system used in the ATLAS experiment [9]

CHAPTER 4

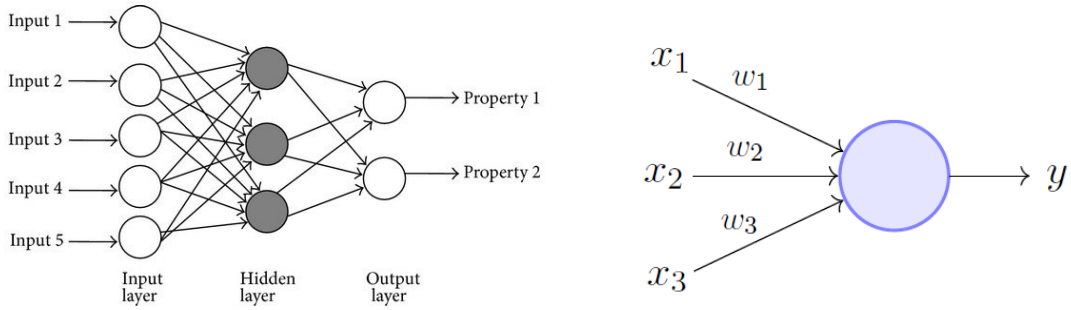
Deep learning

Even before the expression *artificial intelligence* was coined in 1956, classical philosophers speculated about inanimate machines that could mimic human intelligence. Although today, the definition of artificial intelligence is much broader, but also elusive, the original idea is manifested in machine learning. In this field, algorithms and models which enable machines to improve at tasks with experience, i.e. learning from information of any kind and form, are studied. The aim is to program computer systems to execute a task successfully even though not being explicitly programmed for that task. A subfield of machine learning, Deep Learning, tries to achieve this by using artificial neural networks (ANN) which are inspired by biological neural networks.
Although not fully understood, Deep Learning is often used for various classification problems like speech and image recognition or translation tasks with great success. Their performance equals and often surpasses conventional methods to date. Therefore, deep learning algorithms are also used in particle physics and the results are, at least in the case of tau decay mode classification, promising and indicate better performance.
This chapter will introduce the general concept of neural networks and extend it specifically to recurrent neural networks (RNN). Furthermore, the choices, so-called hyperparameters, regarding the structure are presented, too. All of this will be discussed with regard to the fact that the provided data is labelled which means that the input and the targeted output are known a priori. From this, the algorithm can deduce a rule which, then, can be applied to unknown data. This is called *supervised learning*.

4.1 Feedforward neural network

Feedforward neural networks are the most basic type of artificial networks. An example is the multi-layered perceptron which can be seen in Figure 4.1(a).

As can be seen, they consist of a number of input nodes which just pass provided data to the nodes in the hidden layer. These process the information with computations and transfer the results to the next layer which is, in this case, the output layer. Here, the available information will be evaluated again and an output is given. As it is obvious, these processing nodes which are inspired by neurons in the human nervous system are the basic units of the network to imitate decision-making. They are arranged in layers and each of them receives an input $\vec{x}$ from a previous layer while each x_i is multiplied with a input-specific weight w_i. These weighted inputs are added with a node-specific bias. Finally, the sum is passed to an activation function f which results in an output y (see Figure 4.1(b)).



(a) Scheme of a multi-layered perceptron with one input layer, one hidden layer and one output layer, the arrows indicate the flow of information [10]

(b) Scheme of one neuron in a hidden layer with input x, weights w and output y [11]

Figure 4.1: Examples of feedforward neural networks

Thus, the processing of data of every artificial neuron can be summed up as

$$y = f\left(\sum_{i=0}^{N}\left(w_i x_i\right) + b\right) = f(a) \qquad f : \mathbb{R} \to \mathbb{R} \tag{4.1}$$

Mathematically, f is only used to map an unbounded input into an bounded output with a predictable form, but it also acts as a rule which information is passed on as input to subsequent nodes. The reason is that, in general, all activation functions are continuously differentiable and monotonic which actually enables gradient-based optimization methods (more in 4.1.1). The main reason for their implentation is, however, to introduce non-linearity into the network. That is why the selection of the function is often dependent on the given task. This will be discussed in 4.1.2.
An important property of feedforward neural networks is depicted by the arrows in Figure 4.1(a). They show the information flow starting from the input. Thus, in those kinds of networks, everything is processed in the forward direction with no feedback loops between subsequent layers. It should also be noted that the number of hidden layers in a model is variable. A model with more than one hidden layer is considered as a *deep neural network*.
All in all, all components, like the number of layers and their nodes and also the activation function, of a network model are arbitrary and they should be customized depending on the task the neural network is trained for. These components are also called *hyperparameters*.

4.1.1 Forward and backpropagation

As touched before, a combination of input data, weights and bias produce an output in each node of a network. In supervised learning, data is labelled so that a neural network has to learn how to conclude to the given output just by the provided input. Thus, only weights and bias are adjustable to change the outcome of a node which means that *learning* essentially consists of finding the right values to produce the desired decision of a node. The underlying main concepts of this task are called forward and back propagation and shall be shortly discussed. However, only a general picture will be provided so that the processes and motivations behind these algorithms can be understood. For a more

extensive elaboration, see [12].
The concept of forward propagation is basically summarized in equation 4.1. Every neuron after the input layer processes their input by this principle and the result of this computation consequently influences the output of the nodes of the next layer. Eventually, the output nodes predict an outcome. So, during forward propagation, a prediction is made with given input, weight and bias values. Ideally, it is correct, but if that is not the case, the weights and biases need to be adjusted. To quantify the prediction error, a metric is needed which is dependent on the predicted outcome $\vec{p}$ and the labelled outcome $\vec{l}$, the so-called loss function $L(\vec{p},\vec{l})$.
Backpropagation is an algorithm which aims to minimize the loss function by adjusting the network's weights and biases. The level of adjustment is determined by the gradients of the loss function with respect to those parameters. So, following the definitions in equation 4.1 and using the chain rule, the gradient with respect to the weight for the last layer n can be calculated with

$$\frac{\partial L}{\partial w^{(n)}} = \frac{\partial L}{\partial y^{(n)}} \frac{\partial y^{(n)}}{\partial a^{(n)}} \frac{\partial a^{(n)}}{\partial w^{(L)}} \tag{4.2}$$

With this, it is known how a change of the weight respectively the bias affects the loss function. So, adjusting the value iteratively with

$$w_{i+1}^{L} = w_{i}^{L} - \eta \frac{\partial L}{\partial w^{L}} \qquad \eta > 0$$

will result in decreasing loss. η represents the learning rate which determines the gradient's influence.
Another advantage of backpropagation is that it exploits the fact that neural networks mathematically consist of composite functions. So, using the chain rule repeatedly, the gradients of the loss function with respect to the weights can be calculated for all other layers while reusing former results. For example, one would get for the layer n-1:

$$\frac{\partial L}{\partial w^{(n-1)}} = \frac{\partial L}{\partial y^{(n)}} \frac{\partial y^{(n)}}{\partial a^{(n)}} \frac{\partial a^{(n)}}{\partial y^{(n-1)}} \frac{\partial y^{(n-1)}}{\partial a^{(n-1)}} \frac{\partial a^{(n-1)}}{\partial w^{(n-1)}}$$

The first two factors are already known from the calculation in equation 4.2 and can be reused. Analogous, all equations can be applied to adjust biases.
Performing forward and backpropagating in alternating order leads to a decreasing loss function until a minima is found. Since the supposedly optimal weights and biases are obtained by using gradients, this method is also called *gradient descent*.
It should be mentioned that although it is the aim to reach the global minima of the loss function, it cannot be guaranteed. If the adjusting steps to the parameters are too small, the possibility exists that the algorithm terminates due to a no longer decreasing loss function. There are various optimization algorithms which all try to achieve the best and fastest convergence to the global minimum. The most established ones and their underlying motivation is described in [13]. The optimizer used in this work is the Adam, derived from *adaptive moment estimation*, optimizer [14]. It stands out as offering quick convergence with little parameter tuning due to its adapting, ergo non-static, learning rates.

4.1.2 Activation functions

Since activations functions contribute greatly to the network's ability to predict relations which varies non-linearly with given variables, they shall be quickly discussed in the following. Since they actually function as a mathematical gate between the input feeding the current node and its output going to the next layer, every transformation can be considered as an activation function. However, to ensure some abilities of a neural network, there are some desirable properties for them.
First, it is important that a model is able to learn from tasks which are non-linear which means that the correct output cannot be reproduced from a linear combination of inputs. If the used activation function were a linear one, all hidden layers could be collapsed into just one hidden layer. In this case, the neural network wouldn't be able to learn from complex situations. Second, since most optimization algorithms follow the gradient descent algorithm, it is required that the activation is continuously differentiable. Only if that is fulfilled, gradients can be calculated. Furthermore, there shouldn't be a continuous domain which equals zero. Gradient-based methods would have problems to make progress, too, then. There are, of course, other properties like monotony for more stable training or a finite codomain, but they aren't strictly needed.

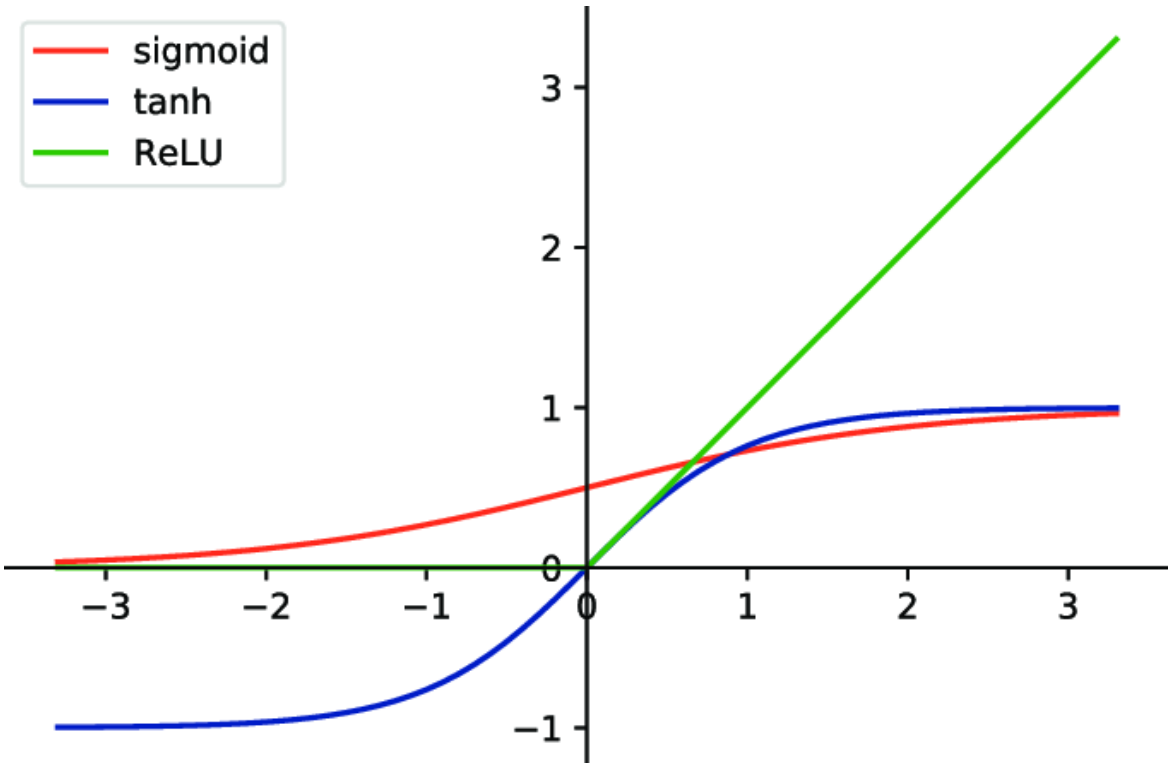


Figure 4.2: Comparison of three activation functions: sigmoid, tanh and ReLU [15]

There is a great number of activation functions to choose from and each has its advantages and disadvantages. In Figure 4.2, a graphical comparison of the three most popular ones can be seen. Since the neural network discussed in this work will use two activation functions, hyperbolic tangent (tanh) and softmax, they are shortly presented.

Tanh is defined as followed:

$$f(x) = tanh(x) = 1 - \frac{2}{e^{2x} + 1}$$

The output is bounded between -1 and 1, monotonous and since this function is also continuous differentiable, it is suited as an activation function. Furthermore, the output is zero centered which eases optimizing weights during backpropagation. However, the tail regions lead to a downside since the gradient approaches zero. This can lead to the network not being able to reach an accurate prediction or, at least a slow convergence. This problem is also called *vanishing gradient problem*.
Softmax is defined as followed:

$$f_i(x) = \frac{e^{x_i}}{\sum_k e^{x_k}}$$

It is used for multi-class problems and the output is a normalized probability distribution of all possible outcomes. Typically, this activation function is only used for the output layer to classify inputs into multiple categories.

4.2 Recurrent neural networks (RNN)

Although performing many tasks with great success, feedforward neural networks are still limited. They are not able to do predictions based on already provided informations from the past. One can say that past inputs are always forgotten and because of that they are not able to guess future events. Sequential data which, for example, can be obtained by tracking weather data or the stock market, but also occurs in particle physics since reconstructed events can contain reconstructed objects of the same type like clusters or tracks, cannot be used by feedforward networks to its full extent. RNNs provide a model structure which is able to process such data by using cyclical connections so that information does not strictly flows forward anymore. With this extension, RNNs can be said to have a memory to do future predictions.

4.2.1 Simple recurrent neural networks

Simple recurrent neural networks have the same basic structure like feedforward neural networks with the exception that nodes, now, can have loops to themselves. The left side of Figure 4.3 depicts such a neuron.

A unit s gets an input x and produces an output o. U and V are the respective weights. Additionally, a loop to iself with weight W exists as well. Now, if the network shall consider an arbitrary sequence length n, the loop is run n times, effectively memorizing n data points. Although not required, during every cycle, an output can be given, too. This can also be seen as a chain of nodes getting input and calculating an output. So, unrolling the loop leads to the right side of Figure 4.3 and it can be concluded that s acts as memory of the network. At every time step t, the state of s_t is calculated similar to a feedforward neural network by

$$s_t = f\left(Ux_t + Ws_{t-1}\right)$$

Normally, the node possesses a bias as well, but since this value is just a constant, it will not considered now. f is an activation function. s_{t-1} which is also needed to calculate the first state s_1 is set to zero at the beginning. To reduce the number of parameters to optimize, weights and biases are shared across

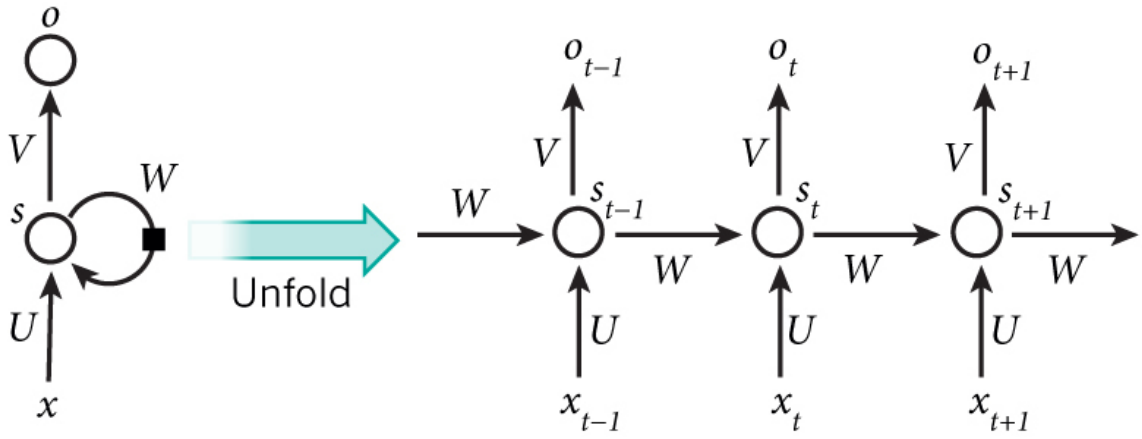


Figure 4.3: Left side: Structure of one node of an RNN with a direct feedback loop, right: unfolded node into three time steps [16]

all time steps. Adjusting these parameters happens, also similar to feedforward networks, through backpropagation and is called *backpropagation through time*. This algorithm functions like standard backpropagation except with the extension that nodes with loops have to be unrolled when calculating gradients with respect to a particular weight.
In theory, there should be no problems for an RNN anymore to consider previous data to perform a given task. For this, the network can always be constructed deeper to have a longer memory for long-term dependencies. However, it was proven that the wider the gap between a prediction and the needed information for it is, the more often exploding gradients or vanishing gradients tend to occur [17, 18]. While it is easy to find a countermeasure for the first problem, the solution to the second one had to be solved by extending simple RNNs.

4.2.2 Long short-term memory (LSTM)

LSTMs are designed to overcome the vanishing gradient problem and also allow to retain information for longer periods compared to traditional RNNs [19]. They use gated cells, namely an input, an output and a forget gate, to store information outside the regular flow of the RNN. With these cells, the network can manipulate the information in many ways, including storing information in the cells and reading from them. The cells are individually capable of making decisions regarding the information and can execute these decisions by opening or closing the gates.

In Figure 4.4, the unfolded structure of a basic unit is shown. Here, its external inputs are its previous cell state $c_{(t-1)}$, the previous hidden state $h_{(t-1)}$ and the current input vector x_t. Additionally, the inputs of the hidden state and the input vector are, although not sketched in the figure, weighted as well. As usual, the output of a gate can be (while disregarding biases) calculated as

$$f_t = \sigma\left(W_f x_t + W_f h_{t-1}\right) i_t = \sigma\left(W_i x_t + W_i h_{t-1}\right) o_t = \sigma\left(W_o x_t + W_o h_{t-1}\right)$$

σ is the activation function which maps the input into the interval [0,1]. Multiplying these output values elementwise with another vector, basically defines how much information is allowed to pass.

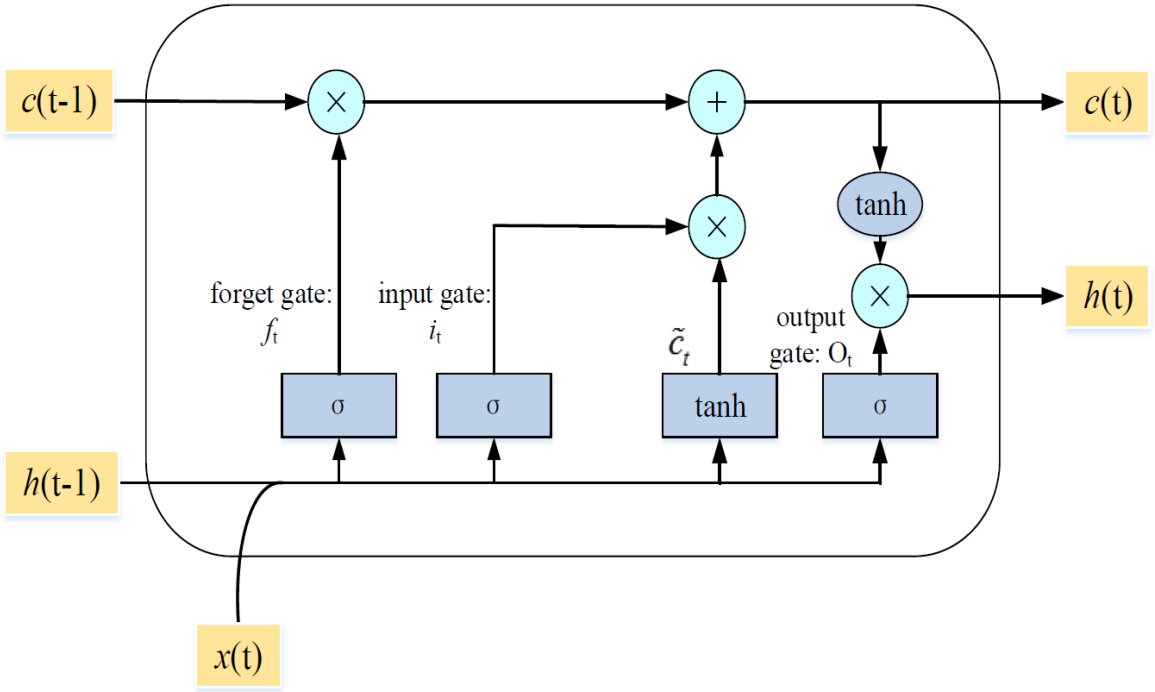


Figure 4.4: Structure of an LSTM unit [20]

The output gate removes information that is no longer necessary for the completion of the task, the input gate defines how much of the newly computed state shall be used in the LSTM unit and the output gate selects and outputs necessary information to the subsequent parts of the RNN. Following these just described roles for every gate, the cell state in a time step t is defined as

$$c_t = f_t \bullet c_{t-1} + i_t \bullet \tanh\left(W_c x_t + U_c h_{t-1}\right)$$

and the output h_t is calculated as

$$h_t = o_t \bullet \tanh\left(c_t\right),$$

where $\bullet$ denotes elementwise multiplication with a vector.

CHAPTER 5

RNN architecture and used data

There are many factors which can be adjusted to achieve a trained neural network with high performance. Not only is the general structure important and often dependent on the area of application, but also the choice of the hyperparameters and the provided data.
This chapter's aim is to give a description of the used RNN and what is needed to achieve a suitable environment for training. More precisely, the starting point based on latest studies in [21, 22], i.e. the input variables and, from there, the resulting structure of the RNN, are presented. Furthermore, the data sample which is used for the training and its preparation will be discussed.

5.1 Input variables

Similar to the `PanTau` algorithm, the RNN is mainly provided with kinematic information of reconstructed decay products. For this, Particle Flow Objects (PFO), which will be quickly discussed in 6.1, are suited and can be used to classify the hadronic decay modes. In addition to these, further kinematic quantities of the reconstructed tau axis are used to complement information provided by the PFOs.
The common input variables for both kinds of PFOs, charged and neutral, are the transverse momentum associated with the particular PFO $p_{\mathrm{T,\,PFO}}$ and the signed angular distances to the reconstructed tau axis in transverse and longitudinal direction, namely $\Delta\phi$ and $\Delta\eta$. As mentioned before, the visible transverse momentum $p_{\mathrm{T},\tau}$ and the angular directions ϕ_τ and η_τ of the reconstructed tau are added as well.
Since the identification of the number of neutral pions can be a hint for the decay mode, additional properties of neutral PFOs are included in the training. The full list of all input variables associated with neutral PFOs can be seen in Table 5.1. With these input variables, the amount of information is already comparable to the PanTau algorithm. To get even better results, conversion tracks and photon shots are added as well, the first being tracks which are created by long-lived secondary particles from converted photons or hadronic interactions while the second describes highly collimated photons from π^0 decays. The specific input here is defined in analogy to the common variables of the charged and neutral PFOs.
All in all, the number of input variables based on former studies in [21] amounts to 32 and can be seen in Table 5.1. However, this list is optimized later in further studies. These results are presented and discussed in 6.

Category	Variable	
Charged PFO	Φ_τ	transverse angular direction of reconstructed tau axis
	$(\Delta\Phi)_{\mathrm{PFO},\,\tau}$	angular distance between PFO and reconstructed tau axis
	η_τ	longitudinal angular direction of reconstructed tau axis
	$(\Delta\eta)_{\mathrm{PFO},\,\tau}$	angular distance between PFO and reconstructed tau axis
	$\log\left(p_{\mathrm{T,\,PFO}}\right)$	log-transformed momentum of PFO
	$\log\left(p_{\mathrm{T},\,\tau}\right)$	log-transformed momentum of reconstructed tau axis
Neutral PFO	Φ_τ	transverse angular direction of reconstructed tau axis
	$(\Delta\Phi)_{\mathrm{PFO},\,\tau}$	angular distance between PFO and reconstructed tau axis
	η_τ	longitudinal angular direction of reconstructed tau axis
	$(\Delta\eta)_{\mathrm{PFO},\,\tau}$	angular distance between PFO and reconstructed tau axis
	$\log\left(p_{\mathrm{T,\,PFO}}\right)$	log-transformed momentum of PFO
	$\log\left(p_{\mathrm{T},\,\tau}\right)$	log-transformed momentum of reconstructed tau axis
	π^0 score	discrimant quantifying likeness of PFO creation by a π^0
	NHitsInEM1	total number of photonsin all shots associated with a neutral PFO Cluster
	SECOND_R	second moment of the radial distance R between cluster cells and shower axis
	η-width in EM1	second moment of the pseudorapidity distance between cluster
	$\langle(\eta-\eta_{cluster})^2\rangle$	cells in EM1 and the energy barycentre of the cluster $\eta_{cluster}$
	NPosECells_EM1	number of cells with positive energy in EM1
	ENG_FRAC_CORE	fraction of the total cluster energy contained in the highest energetic cell in Presampler, EM1 and EM2
	energyfrac_EM2	energy fraction contained in EM2
	ptSubRatio	measure for overlap of neutral and charged PFOs
Photon Shots	Φ_τ	transverse angular direction of reconstructed tau axis
	$(\Delta\Phi)_{\mathrm{PFO},\,\tau}$	angular distance between shot reconstructed tau axis
	η_τ	longitudinal angular direction of reconstructed tau axis
	$(\Delta\eta)_{\mathrm{PFO},\,\tau}$	angular distance between shot and reconstructed tau axis
	$\log\left(p_{\mathrm{T,\,PFO}}\right)$	log-transformed momentum of shot
	$\log\left(p_{\mathrm{T},\,\tau}\right)$	log-transformed momentum of reconstructed tau axis
Conversion Tracks	Φ_τ	transverse angular direction of reconstructed tau axis
	$(\Delta\Phi)_{\mathrm{PFO},\,\tau}$	angular distance between track and reconstructed tau axis
	η_τ	longitudinal angular direction of reconstructed tau axis
	$(\Delta\eta)_{\mathrm{PFO},\,\tau}$	angular distance between track and reconstructed tau axis
	$\log\left(p_{\mathrm{T,\,PFO}}\right)$	log-transformed momentum of track
	$\log\left(p_{\mathrm{T},\,\tau}\right)$	log-transformed momentum of reconstructed tau axis

Table 5.1: Initial input variables for the neural net

5.2 RNN architecture

The RNN uses the KERAS framework with Tensorflow as backend. The optimizer is chosen to be the Adam optimizer. It is an algorithm based on stochastic gradient descent, but with the advantage of adaptive learning rates in constrast to classical algorithms.
The network is constructed in accordance to the main input variables categories: charged PFOs, neutral PFOS, photon shots and conversion tracks. Since the input is sequential, it should be noted that the time steps, which means the number objects given at once, are different for all input nodes. A maximum of three charged PFOs, ten neutral PFOs, six photon shots and four conversion tracks are passed per variable. All input variables which are associated with the respective category are given into a shared dense layer with 24 nodes for the PFOs and 16 nodes for the rest. They are called shared since they share their weights to reduce computational load. The results of these layers are directly connected to LSTM layers with the same amount of nodes. The activation function between these layers is selected to be a hyperbolic tangent one which is also used between all upcoming layers. This information gets merged after this and is passed through two more dense layers with 64, and 32 nodes respectively. After that, another dense layer is added with 5 nodes, one for every examined decay mode. With using the softmax activation function, the RNN is able to classify between the five tau decay modes and provides probability estimates for every outcome. A pictured summary of the whole model can be seen in Figure 5.1.

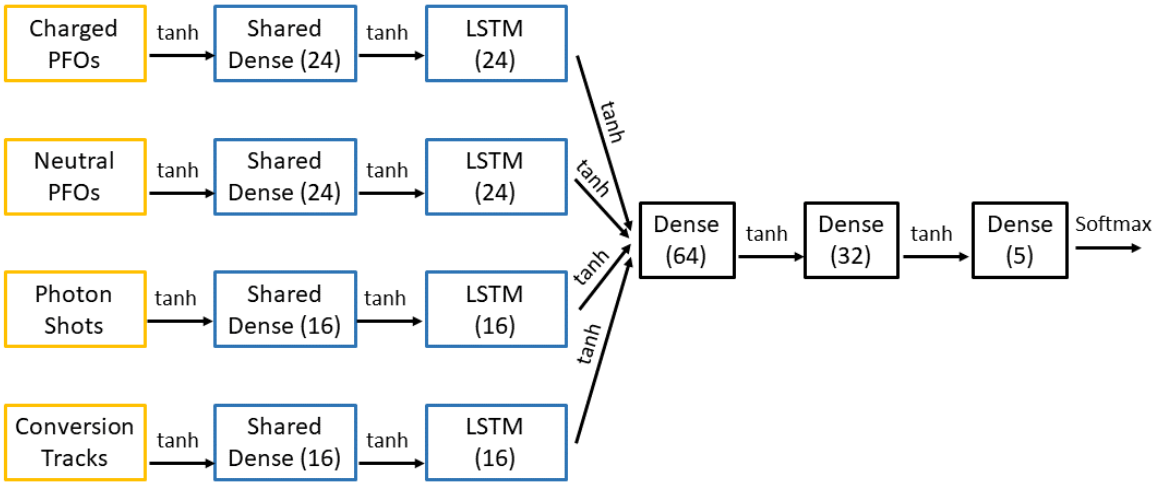


Figure 5.1: Architecture of the RNN with the number of nodes per layer denoted in brackets

5.3 Data

5.3.1 Data sample

The provisioning of enough data which also cover different statistics are essential for the training of a neural network. Since the net discussed in this thesis learns by supervision, the correct labelling of data is important, too.
For this, Monte Carlo generated data is used. The data sample contains events generated by the

unphysical process $\gamma^* \rightarrow \tau^{\text{had}}\tau^{\text{had}}$. The simulation of the events are done by the PYTHIA8 event generator.

5.3.2 Event selection

Before being able to train the RNN, the data sample needs to be reduced to suitable events which help achieving the task the neural network will be working on the most. This affects many important aspects of the training like the duration of the whole procedure because of the smaller data size. Furthermore, modelling errors can be reduced as well and, in general, the system gets simpler which supports interpretability. Therefore, cuts to separate unnecessary events are applied.
The preselection matches typical selections used in other analyses. The following cuts are applied:

- $p_{\text{T},\,\tau} > 20\,\text{GeV}$
- $|\eta| < 2.5$ to get candidates in the acceptance range of the tracking system
- $1.37 < |\eta| < 1.52$ to reject candidates between barrel and end-cap of the calorimeter
- only tau candidates with a number of charged tracks of 1 or 3 are accepted
- $p_{\text{T},\,\tau} < 100\,\text{GeV}$
- all mentioned cuts are also used for τ at truth level
- decay modes needs to be smaller than 5, i.e. `truthDecayMode < 5`

The cut which limits the upper momentum of tau candidates is used since the Particle Flow algorithm is rather optimized for the low-momentum regime due to the decreasing momentum resolution in the tracking system at high transverse momenta. However, studies regarding the training and use of the neural net for higher momentum regions should be considered in the future.

5.3.3 Data preparation

In order to evaluate the training process, the data sample is split into three sets: a training set, a validation set and a test set. The ratio between all sets is 4:5:1. To utilize the limited data the most, it is also possible to perform k-fold cross-validation to get different sets of training, validation and test data, but since it is computationally more expensive to do this and since the data sample was big enough, this simple split is used.
Furthermore, the RNN only takes sequences as input. Because of this, it is needed to portion the data into smaller pieces according to the chosen time steps as mentioned in section 5.1.

CHAPTER 6

Tau decay mode classification

An efficient reconstruction and classification of tau lepton decays is highly important for many physics analyses at the LHC. With this, the identification of the tau lepton can be improved, but understanding the tau decay itself can also contribute to other researches. Therefore, various algorithms were developed to improve the performance, mainly the accuracy of correct classification. In this work, the focus lies on the following decay modes:

Decay mode	also denoted as
$\tau^- \rightarrow h^- \nu_\tau$	1p0n
$\tau^- \rightarrow h^- \pi^0 \nu_\tau$	1p1n
$\tau^- \rightarrow h^- \geq 2\pi^0 \nu_\tau$	1pXn
$\tau^- \rightarrow h^- h^+ h^- \nu_\tau$	3p0n
$\tau^- \rightarrow h^- h^+ h^- \geq 1\pi^0 \nu_\tau$	3pXn

Table 6.1: List of to be discriminated decay modes and their shorthand

with $h^\pm$ being either charged pions or kaons. Decay modes containing intermediate K^+ are not considered.
In this chapter, a quick recap about the current method for tau decay mode classification called `PanTau` is done. Afterwards, performance improvements achieved by using an RNN described in Chapter 5 are presented and, following that, used methods to optimize said neural network to improve the efficiency of classification are described and evaluated.

6.1 PanTau

6.1.1 CellBased energy flow algorithm

Energy flow is a concept with which the identification of neutral clusters by substracting estimated energy deposits of charged particles from the calorimeters can be improved. The `CellBased` algorithm is one implementation of this concept and it was developed specifically for tau particles. It consists of two parts: First, hadronic showers produced by charged pions are removed and, after that, neutral pions are identified in the remaining calorimeter deposits.

Subtraction of hadronic showers This part is defined by the following algorithm. The clusters in the electromagnetic calorimeter produced by charged pions are located by using tracker information. Then, the total amount of energy deposited by the specific charged pion is estimated. After that, this total energy is distributed to each layer and substracted. The estimation is calculated by $E_{ECal}^{estimate} = p_{Track} - E_{HCal}$ [23].

Identification of neutral pions The remaining energy clusters are reclustered using the ATLAS topological clustering algorithm. Ideally, the remaining clusters are neutral pions, but since there can also be other sources for clusters, for example from pile-up, a BDT is used to separate clusters of neutral pions from those which originated due to other sources.

6.1.2 PanTau algorithm

PanTau is an algorithm which approaches the task of classifying decay modes by using decay product kinematics. For this, it uses the kinematics of all particles found by the Particle Flow algorithm, in this case from the Cellbased allgorithm, in combination with the estimated decay mode on counting the number of $\pi^{\pm}$ and π^{0}. Reconstruced taus which are classified to decay in a decay mode which is likely to be confused with another one will be tested again by a Boosted Decision Tree (BDT) which was trained to be able to distinguish between these two modes. With this, the overall classification can be improved. The rough workflow of this algorithm can be seen in Figure 6.1.

The performance metric for decay mode classifications are quantified in *migration matrices*. Depending on whether the columns or the rows are normalized, information about the efficiency or the purity can be obtained. In this work, only efficiency matrices will be considered and, thus, will be called *migration matrices* from now on. The migration matrix of PanTau can be seen in Figure 6.2. Based on this example, it will be explained how such a matrix is read.

As one can see, the true decay mode is shown on the x-axis and the reconstructed decay mode on the y-axis. Now, an entry on a column c is reconstructed as the mode belonging to r. So, for example, the efficiency of a true 1p0n decay mode to be reconstructed as a 1p1n is 16.6 %. What is important is the diagonal line because the fraction of correctly classified tau leptons per decay mode is indicated. The entries on this line can also be summarized into the diagonal efficiency, here 73.1 % which describes how many tau leptons are classified based on the number of all tau leptons. This value will be the metric for later performance tests in this work.

6.2 Classifying tau decay modes with an RNN

With their ability to perform classification tasks well, recurrent neural networks (RNN) are suited for tau decay mode classification. First studies, done by [21], show that the results are promising. As discussed in Chapter 5, the final model of the RNN do not only includes Particle Flow objects (PFOs), charged and neutral ones, like PanTau, but also photon shots and conversion tracks. The resulting migration matrix can be seen in Figure 6.3.

It should be emphasized that, already with this setup, misclassification seems to occur not as often as with PanTau, indicated by the higher diagonal efficiency. However, the question arises if this is the limit which can be achieved or if the result can be improved even further.

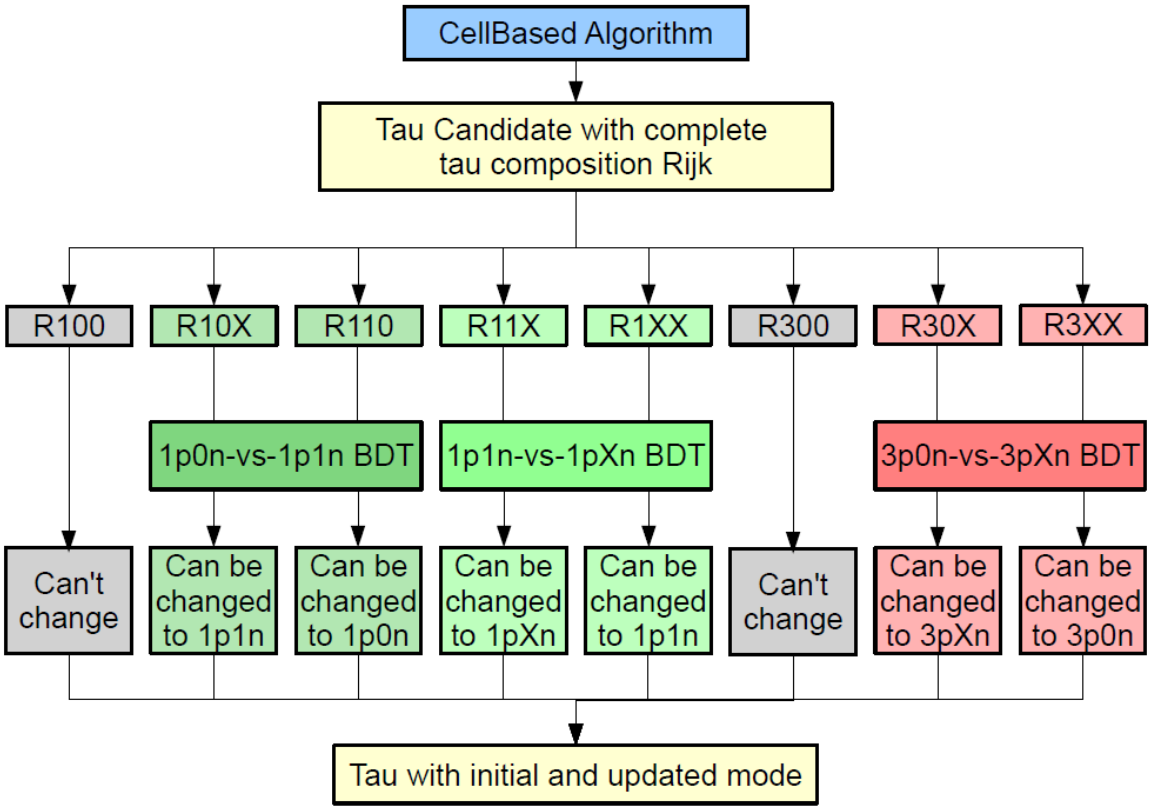


Figure 6.1: First step of the PanTau workflow. Reconstructed taus from a cellbased Particle Flow implementation are used as input and ordered different categories depending on their composition. Three BDTs test evaluate likely to be confused decay modes again and correct them if needed. Full workflow can be seen in appendix 9.3 which includes updating the 4-momentum of the tau.

6.2.1 Training a neural network with PanTau variables

It was observed that an RNN yields a better result than the current method, PanTau. However, it should be considered that to obtain this, more information was used, too, since, for example, photon shots and conversion tracks are included. That is why the question should be answered if this improvement is caused by the neural network itself, which means that it is really able to discriminate the decay modes better, or by the additional provided information. Therefore, an RNN was trained while the provided variables are exactly the same as used in PanTau. The variable list can be seen in Table 6.2.

After training and evaluating the neural network, the migration matrix seen in Figure 6.4 is obtained.

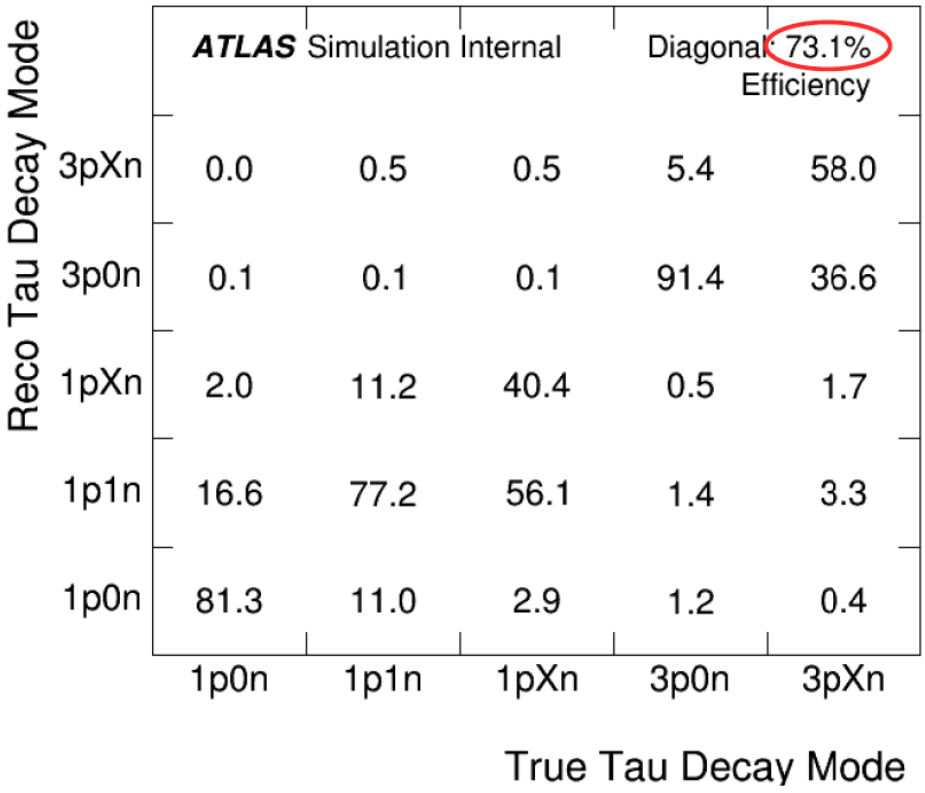


Figure 6.2: Migration matrix of the PanTau algorithm

discriminated decay modes	variable name
1p0n vs 1p1n	TauJets.PanTau_BDTVar_Combined_DeltaR1stNeutralTo1stCharged
	TauJets.PanTau_BDTVar_Charged_JetMoment_EtDRxTotalEt
	TauJets.PanTau_BDTVar_Neutral_PID_BDTValues_BDTSort_1
	TauJets.PanTau_BDTVar_Neutral_Shots_NPhotonsInSeed
	TauJets.PanTau_BDTVar_Neutral_Ratio_1stBDTEtOverEtAllConsts
1p1n vs 1pXn	TauJets.PanTau_BDTVar_Neutral_PID_BDTValues_BDTSort_2
	TauJets.PanTau_BDTVar_Neutral_Ratio_EtOverEtAllConsts
	TauJets.PanTau_BDTVar_Neutral_Shots_NphotonsInSeed
	TauJets.PanTau_BDTVar_Neutral_HLV_SumM
	TauJets.PanTau_BDTVar_Basic_NneutralConsts
3p0n vs 3pXn	TauJets.PanTau_BDTVar_Charged_StdDev_Et_WrtEtAllConsts
	TauJets.PanTau_BDTVar_Neutral_Ratio_EtOverEtAllConsts
	TauJets.PanTau_BDTVar_Neutral_PID_BDTValues_BDTSort_1
	TauJets.PanTau_BDTVar_Neutral_Shots_NphotonsInSeed
	TauJets.PanTau_BDTVar_Charged_HLV_SumM

Table 6.2: List of variables used in PanTau

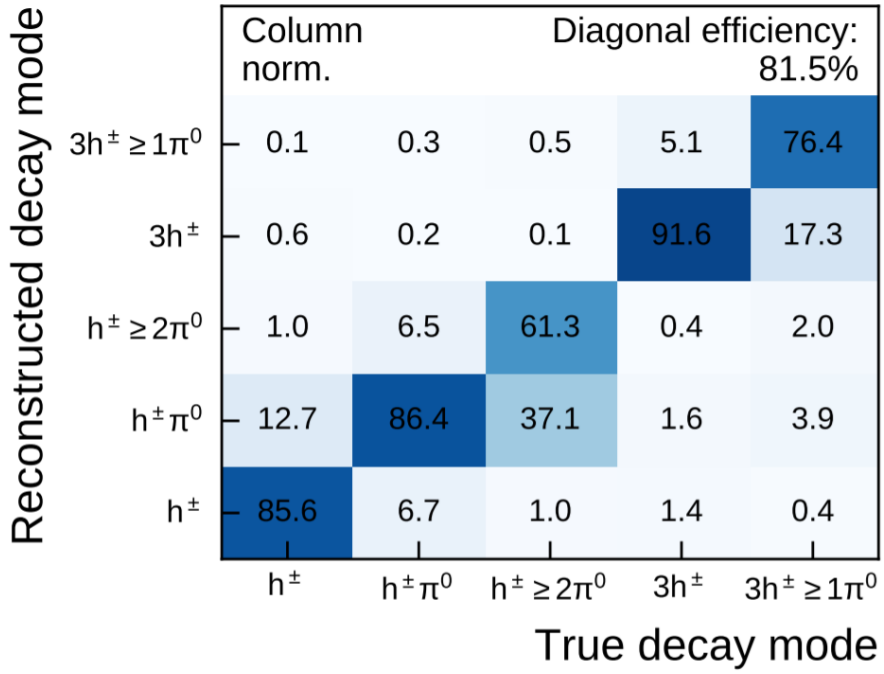


Figure 6.3: Migration matrix showing the classification by an RNN; evaluation done on an independent testing sample

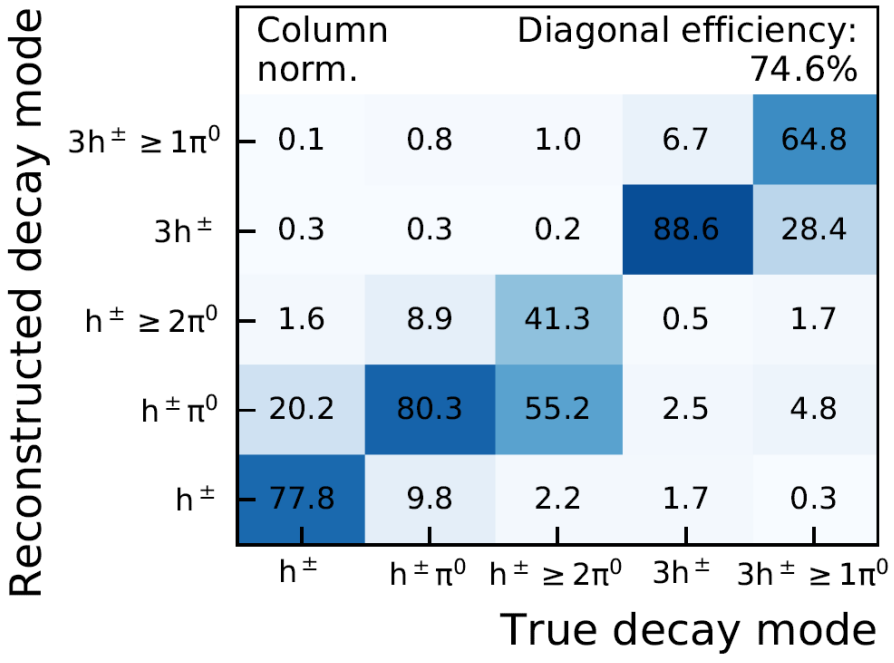


Figure 6.4: Migration matrix obtained by training an RNN with exclusively the variables which are also used in `PanTau`

It can be seen that with 74.6 % the resulting diagonal efficiency is still better than the overall efficiency of 73.1 % which was shown in Figure 6.2. This indicates that the fact that an RNN is used to process the provided data already leads to an improvement. However, the main improvement probably comes from the better integration of the cluster characteristics. Nevertheless, because of this, it seems worthwhile to continue to pursue the development of an RNN for tau decay mode classification.

6.3 Neural network optimization

There are various parameters of neural networks which can be tuned. Starting from deciding the general kind of network through the choice of details like which optimizer or loss function to use or like the value of batch size or learning rate, this should be done methodically. In this work, the attempt to improve the recurrent neural network used to classify tau decays is described. Here, the focus lies on the model-side optimization, which mainly affects the structure, and not on the optimization of the training process to achieve better convergence although those are certainly correlated. Thus, first, the importance of the used input variables and, after considering this, the structure of the net defined by its hyperparameters were examined.

6.3.1 Baseline

Since the performance of a net depends on the training data, the workflow described in [22] is followed with the dataset mentioned in 5.3.1. From this, a diagonal efficiency of 80.2 % was obtained as can be seen in Figure 6.5. This will be the starting point for further discussions about optimization results.

6.3.2 Input variable optimization

The number of initial input variables amounts to 32. The list can be seen in Table 5.1. Based on the amount of variables in comparison to `PanTau`, it could be assumed that the network is given redundant input. That is why the first step of optimizing the used selection was to find out which variables are important for the neural network, i.e. which ones carry crucial information regarding discriminating purposes. This can be done with a *variable ranking*. The objective of this method is to measure the usefulness of all variables with a fitting metric and ordering them to possibly find hints what kind of input leads to the best performance. If the set could be at least reduced, this would already lead to a less complex model which is generally desirable as long as the performance does not drop significantly. The to be minimized function, used by the RNN, is still the loss function and a simpler structure of the network also leads to a less chaotic loss landscape [24] which improves trainability. Furthermore, if other discriminating variables can be found, the redundant ones can be exchanged with these new variables which should likewise result in a better trained net.
The method for variable ranking will be quickly described in the following:
First, a starting point has to be determined. Generally, a neural network is trained with all input variables which shall be examined. Choosing a metric to be able to map the result to a value which can be ranked, a baseline estimator ϵ_{base} obtained. Afterwards, for each variable, the data of the respective variable is replaced with uniformly distributed data which practically omits it from the following training. The result is a value ϵ_n. So, if there are n variables, $n + 1$ different trainings would be required. Applying the metric, the variables can be ordered based on their value and further conclusions can be drawn.

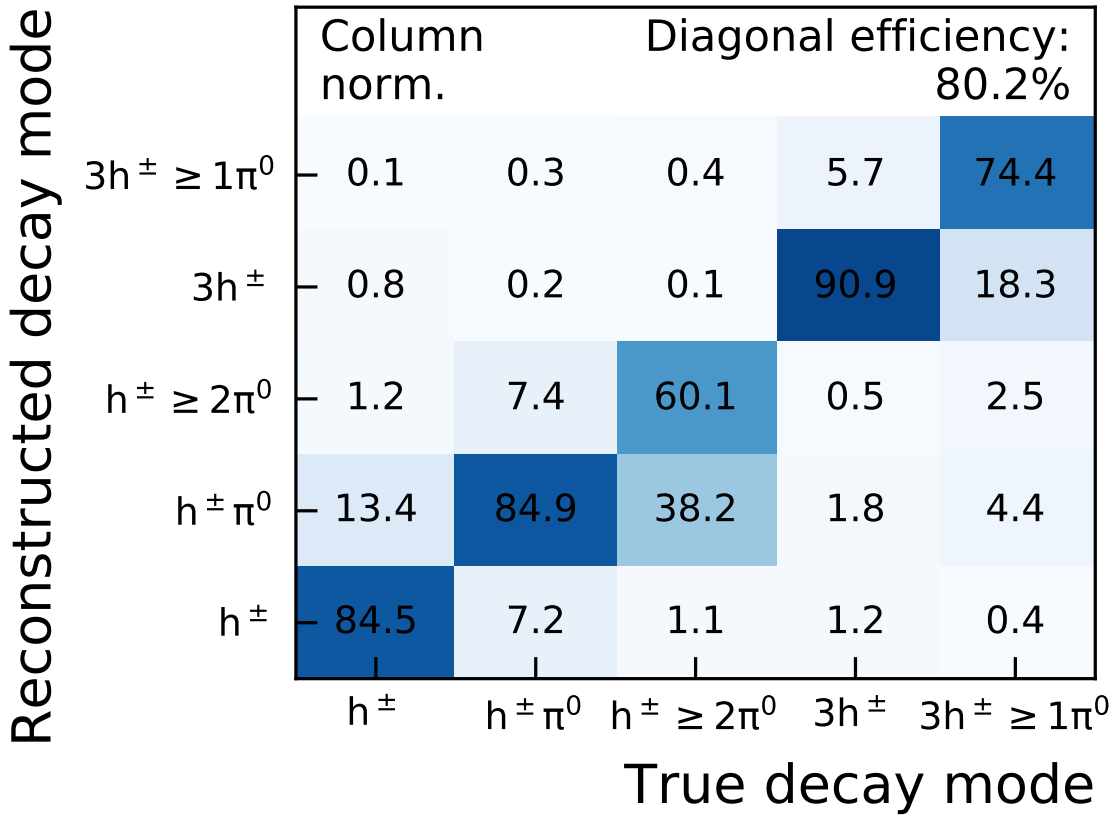


Figure 6.5: Baseline which serves as starting point for RNN optimization; obtained by training and evaluating an RNN as described in [22]

For this study, the variables are ranked by the resulting migration matrix, namely the resulting diagonal efficiency. Table 6.3 shows the already ordered list. It should be mentioned that, to cut down waiting time, the training was done on only half of the whole dataset. Nevertheless, the amount of data should be enough to see tendencies to distinguish the important variables from omittable ones.

Watching the ranking, it can be seen that omitting the variables which appear first on the list does not decrease the overall performance as much. For example, omitting ChargedPFO/eta results in a nominal drop of 0.007 %; the relative efficiency drop amounts to 0.01 %. It is noticeable that especially eta and phi are ranked first. Since this also happens for all four categories, this hints that these variables do not provide crucial, discriminating information as the other ones. Because of this reasoning, these eight variables, two for each category, are decided to be left out from the set of input variables as long as the efficiency drop will not be to big. Furthermore, it is also indicated that the momentum of the tau axis and neutral cluster variables are valued higher since omitting them yields the highest efficiency losses in this study. This knowledge can be used for future researches regarding new additional inputs for the net.
Training the neural network with the full dataset, but without eta and phi for all four categories, is summarized in Figure 6.6.

Baseline estimator	0.7406	1.0000
examined variable	resulting efficiency	$\frac{\epsilon_n}{\epsilon_{base}}$
ChargedPFO/eta	0.7336	0.9905
NeutralPFO/phi	0.7334	0.9902
ShotPFO/eta	0.7333	0.9901
NeutralPFO/eta	0.7309	0.9869
NeutralPFO/pt_log	0.7298	0.9853
ChargedPFO/dphi	0.7282	0.9832
ShotPFO/phi	0.7264	0.9808
ConvTrack/eta	0.7259	0.9801
Convtrack/phi	0.7221	0.9749
NeutralPFO/deta	0.7214	0.9740
ChargedPFO/deta	0.7199	0.9720
ChargedPFO/phi	0.7196	0.9716
NeutralPFO/energyfrac_EM2	0.7187	0.9704
NeutralPFO/nHitsInEM1	0.7186	0.9702
ShotPFO/dphi	0.7155	0.9660
NeutralPFO/ENG_FRAC_CORE	0.7105	0.9595
ConvTrack/deta	0.7093	0.9577
NeutralPFO/dphi	0.7077	0.9555
NeutralPFO/pi0BDT	0.7056	0.9527
ConvTrack/dphi	0.7006	0.9459
NeutralPFO/ptSubRatio	0.7003	0.9455
ConvTrack/pt_log	0.6979	0.9423
ShotPFO/pt_log	0.6966	0.9406
ShotPFO/deta	0.6797	0.9177
ChargedPFO/pt_log	0.6790	0.9168
NeutralPFO/SECOND_R	0.6582	0.8887
NeutralPFO/secondEtaWRTClusterPosition_EM1	0.6499	0.8775
NeutralPFO/pt_jet_log	0.6387	0.8623
ShotPFO/pt_jet_log	0.6299	0.8505
ConvTrack/pt_jet_log	0.5988	0.8086
ChargedPFO/pt_jet_log	0.5658	0.7639
NeutralPFO/nPosECells_EM1	0.4728	0.6383

Table 6.3: Results of the variable ranking starting with the least important variable; the order is decided by the diagonal efficiency ϵ. Training done on half of the whole dataset.

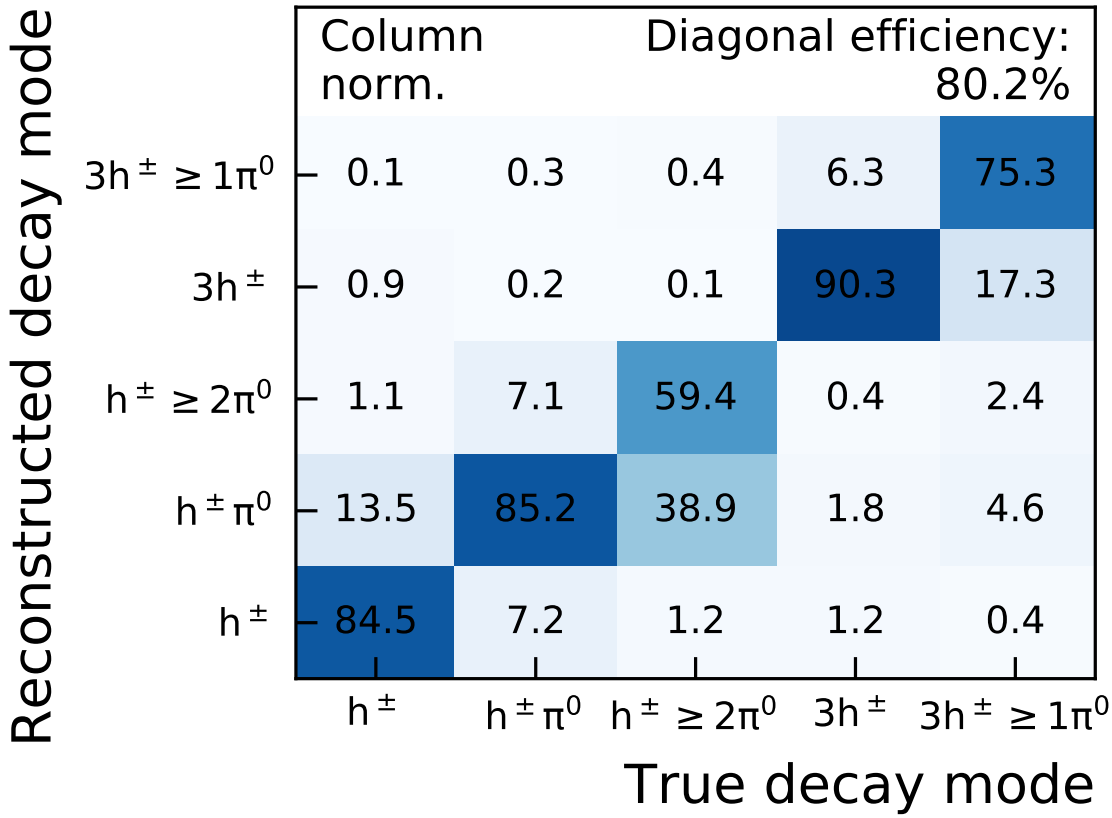


Figure 6.6: Migration matrix obtained by using the new input variable set

Comparing this migration matrix with Figure 6.5, the overall diagonal efficiency stays the same. From this, it can be reasoned that the overall choice was not optimal and the question stands if an even further reduction can be achieved. However, by only doing a basic variable ranking, it was already accomplished to reduce the number of input variables by 8 from 32 to 24. The neural network presently available is less complex which helps, although still difficult, understanding and interpreting the algorithm. What is also important and should be emphasized is that the overall classification efficiency stays the same, thus, no limitations in this regard are to be expected.

6.3.3 Hyperparameter optimization

The neural network structure has a big influence on the ability how many features it can learn from any given data and how it connects them. For example, a model with little layers and nodes will not be able to derive as many and also as complex relations as one having a more complicated structure. That is why it is important to find an optimal set of hyperparameters to process data as efficient as possible. Since the optimization of hyperparameters is an extensive task, this study focused on three hyperparameters: the number of nodes for all layers, the number of dense layers after concatenating and the activation function between all layers (compare Figure 5.1. The used method can be described as follows:

First, for every hyperparameter which is wished to be optimized, a range is defined to define upper and lower limits for the hyperparameter values. Since there was no prior knowledge about some prestudies regarding hyperparameter optimization was available, the search range was kept liberal. Table 6.4 depicts the defined scopes.

hyperparameter	search scope
Chrg dense node	[8, 64]
Neut dense node	[8, 64]
Shot dense node	[8, 64]
Conv dense node	[8, 64]
Chrg LSTM node	[8, 64]
Neut LSTM node	[8, 64]
Shot LSTM node	[8, 64]
Conv LSTM node	[8, 64]
Final dense node 1	[8, 64]
Final dense node 2	[8, 64]
Dense layers after merging	[2, 5]
Activation function	sigmoid, tanh

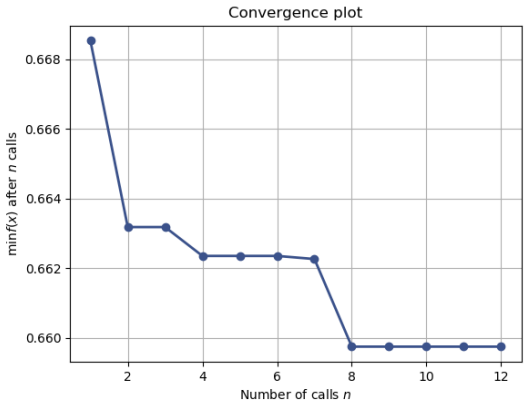
Table 6.4: Chosen search scopes for the to be optimized hyperparameters. The first four hyperparameters denote the number of nodes for the shared dense layers for each category respectively while the *final dense node* denotes the number of nodes of the layers after merging. The remaining variables should be explanatory.

There are various approaches to find an optimal set. Of course, the most constructive one is *grid search* which can be summarized as a brute-force search, i.e. trying out every combination of hyperparameters. Although naive, this algorithm will produce the most optimal hyperparameter combination for the given search area, but it is also the most time-consuming one and, depending on the complexity of the task and time constraints, not always the optimal choice. It is rather suited to be used if the domain of variables is smaller. Because of this, a different approach is used: *Bayesian optimization* [25, 26]. This ansatz follows the motivation that deviation of the best hyperparameters follows a multivariate gaussian. Based on this, "smart" choices of hyperparameter sets are made to minimize a function which is dependent on the examined hyperparameters on every optimization step. Both the function and the optimization steps need to be defined beforehand. In this case, the function is the loss function while the number of steps was chosen to be 12. Furthermore, like in 6.3.2, the dataset used first, was only half of the whole because the optimization process is still computationally expensive although already an improvement from using grid-search. While the RNN was simplified by slimming the input variable set, it can stll be seen as complex. Thus, it was the main goal to look for tendencies of the optimal values first. Then, the search range could be restricted and a hyperparameter search can be done again. Doing this iteratively would possibly lead to an optimal set. To do this, optimization runs were done from the same starting point which was defined as the model structure discussed in 5.2. The results are listed in Table 6.5. In addition to this, the convergence plots visualizing the progression per optimization step can be seen in Figure 6.7.

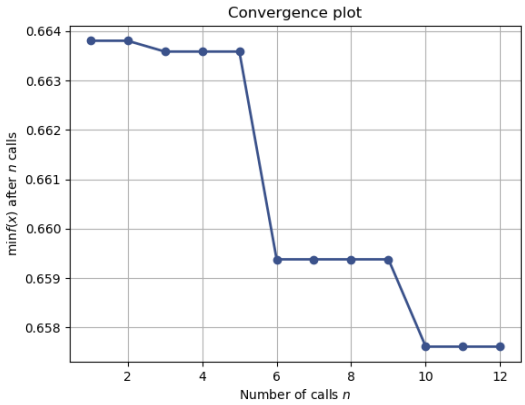
First, it can be seen that the choice of the number of dense layers and the activation function seems to be the most optimal in this search space. Unfortunately, apart from that, the other hyperparameters, the number of nodes, cannot be interpreted to have a tendency. It can be argued that the output can be

hyperparameter	Run 1	Run 2	Run 3
Chrg dense node	16	45	43
Neut dense node	37	63	33
Shot dense node	52	38	60
Conv dense node	12	36	38
Chrg LSTM node	47	11	41
Neut LSTM node	38	44	52
Shot LSTM node	53	11	42
Conv LSTM node	50	41	31
Final dense node 1	61	44	60
Final dense node 2	37	33	30
Dense layers after merging	3	3	3
Activation function	tanh	tanh	tanh

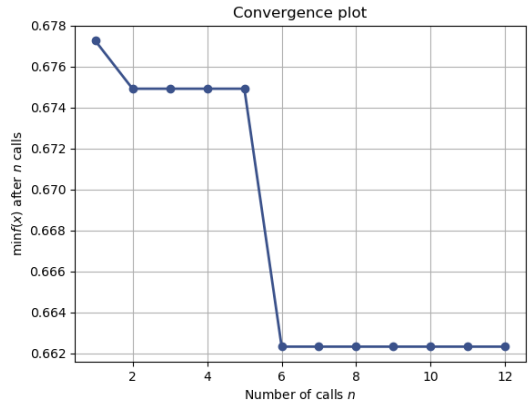
Table 6.5: Output of optimal hyperparameters based on Bayesian optimization



(a) Run 1



(b) Run 2



(c) Run 3

Figure 6.7: Convergence plots of three optimization runs; the starting point is the same for all three cases

interpreted that every layer should at least have 30 nodes since, apart from some cases, all layers are suggested to have at least this number. However, considering these outliers are seemingly distributed randomly, this assumption cannot be seen as absolutely backed. Besides, the randomness indicates that the loss landscape seems to be too chaotic. Although with hyperparameter optimization, the most optimal hyperparameters shall be found, this task gets more and more difficult the more complex the RNN, resulting in more possible local minima in the loss function, is. Since the training algorithm is based on gradient descent, a hyperparameter set is searched which makes it possible to converge to, or even reach, the global minimum. In this case though, multiple runs yielded multiple results which means that every optimization run reached another local minimum. The convergence plots also show that the optimum is found after less steps than defined which can be a hint that the training of the neural net cannot "escape" a local minimum anymore. This supports the assumption mentioned in 6.3.2 that it should be possible to further simplify the model or that it is recommended to achieve this. Given that no real tendency can be observed, one optimization run was done on the whole dataset to adapt the chosen hyperparameters eventually. These can be seen in Table 6.6; the convergence plot can be found in Figure 6.8.

hyperparameter	final hyperparameters
Chrg dense node	17
Neut dense node	22
Shot dense node	60
Conv dense node	55
Chrg LSTM node	29
Neut LSTM node	25
Shot LSTM node	58
Conv LSTM node	28
Final dense node 1	64
Final dense node 2	57
Dense layers after merging	3
Activation function	tanh

Table 6.6: Final hyperparameters after performing hyperparameter optimization on the whole dataset

The convergence plot shows that, in this case, the algorithm already found an optimum after three optimization steps. It has yet to be seen if using these hyperparameters results in an improvement. The corresponding migration matrix is Figure 6.9.

Evaluating this, a small improvement is achieved. Thus, it can be concluded that it is possible to, even though a little, increase the performance of the RNN by using hyperparameter optimization.

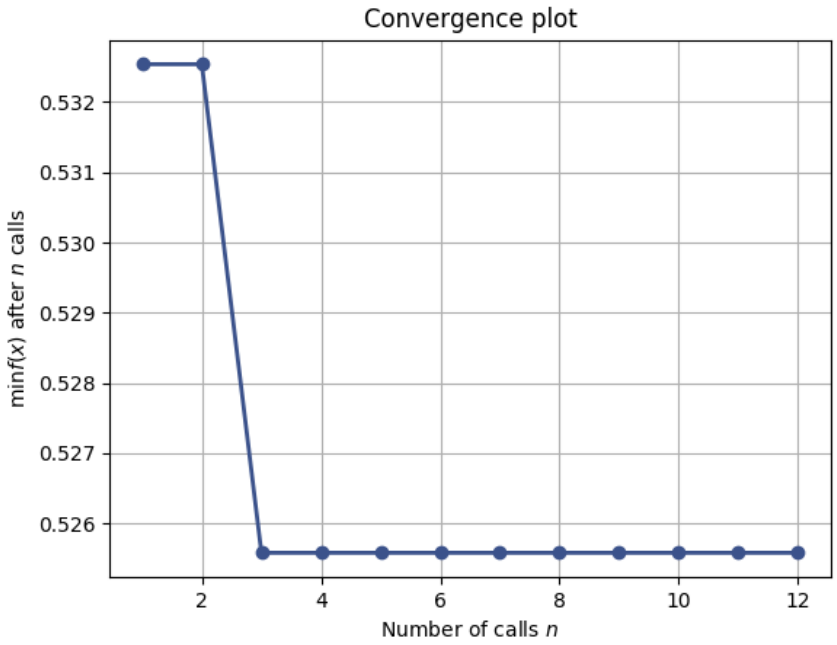


Figure 6.8: Convergence plot of optimization run on whole dataset

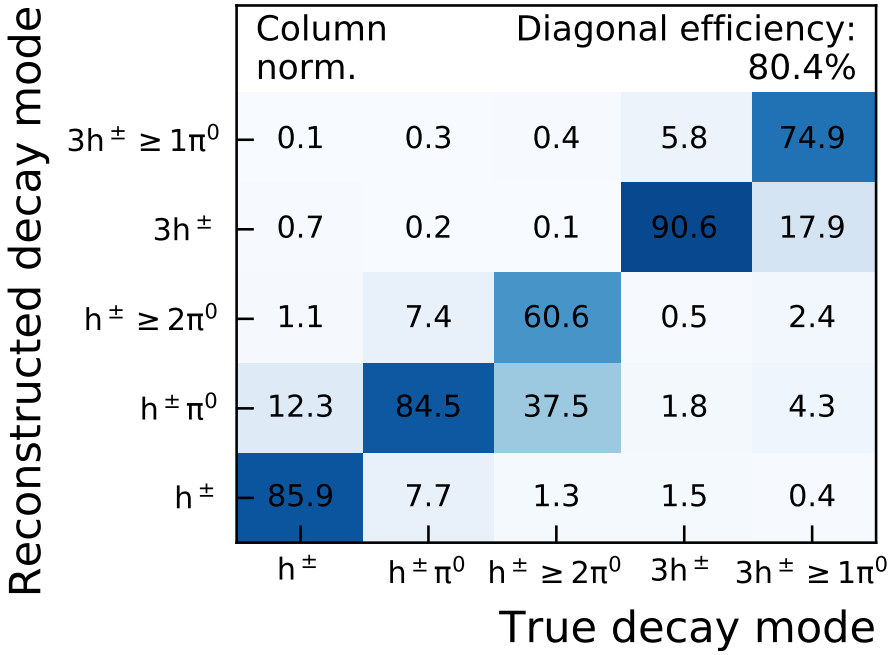


Figure 6.9: Migration matrix with hyperparameters from Table 6.6

CHAPTER 7

Data/Monte Carlo comparisons of variables

The recurrent neural network which is presented in this work is trained by supervised learning. To be able to do this, it is required that the used data is labelled so that the algorithm can find general relations which can be exploited to perform the given task. Apart from that, with increasing complexity, it is also advantageous to have as much representative data as possible although there will be diminishing returns from some point on. Nevertheless, the more general a problem is, the more data should be available. Therefore, Monte Carlo (MC)-generated data is perfect to train neural networks. First, since it is simulated data, it can be labelled accordingly so that supervised learning is made possible. Experimental data taken from the detector is initially not suited because, especially in the case of decay modes, it is not known for sure which decay occured. Second, there is practically no limit on available data because getting simulated data is only restricted by the time needed to produce them. However, to also ensure the representativeness, this means that it, nevertheless, needs to behave as real world data. In this chapter, it is investigated how real and simulated data of the input variables used for training the RNN compare to each other. Therefore, a well-known resonance is considered which is the Z boson peak in this case.
Since kinematic variables are usually well understood, these are first examined and the results presented. Following that, the same procedure is done for the RNN variables.

7.1 Dataset and event selection

The data used during the studies are samples of pp collision data recorded at a center-of-mass energy of $\sqrt{s} = 13\,\text{TeV}$ in 2018. This corresponds to an integrated luminosity of $54\,450\,\text{fb}^{-1}$.
The simulated MC samples contain simulations of a $Z \rightarrow \tau_{lep}\tau_{had}$ signal and of various other backgrounds, namely $Z \rightarrow ll$, W+jets and top quark production. For each of them, the event production is simulated by the PowhegBox generator while Pythia8 is used for the modelling of the parton shower and hadronisation and underlying event. The inclusive samples for the W and Z boson production can be seen in Table 7.1. More information can be found in [27]. The list of used datasets are listed in appendix 9.2.2.

A *tag-and-probe* approach is used to have an as unbiased selection of tau objects as possible. Therefore, the approach consists of selecting events triggered by the presence of a lepton (tag) and containing a hadronically decaying tau lepton candidate (probe) in the final state. In this case, the *tag* is a muon from a leptonic tau decay. The only requirement for the tau object is to have at least an

DSID range	Description
361100 - 361108	$W^{\pm}.Z/\gamma^{*}$ with e, μ, τ decays

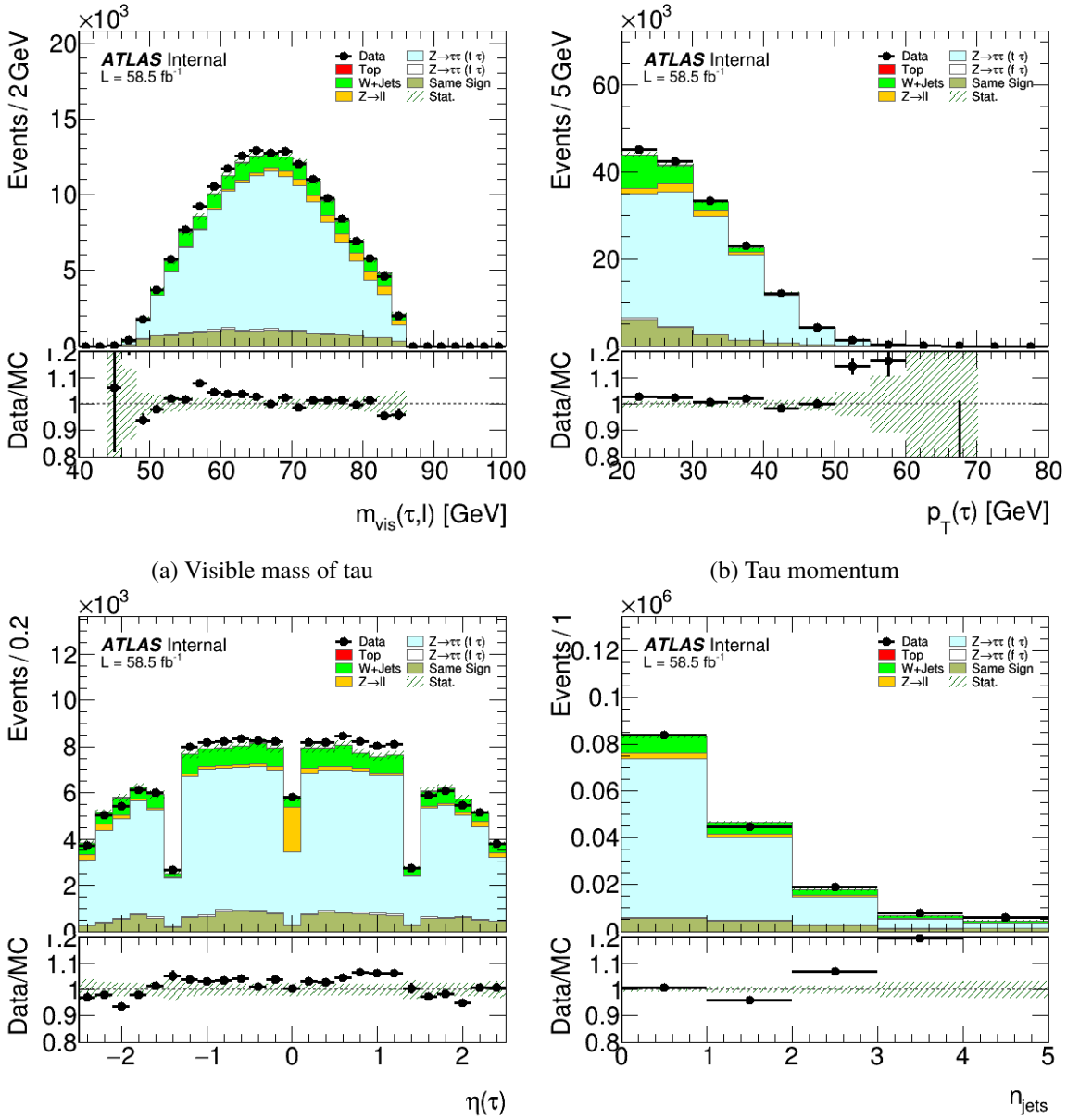
Table 7.1: Inclusive samples with Powheg

energy of 20 GeV. Otherwise, the used cuts on the *tag* side to get a pure selection of hadronic taus are as follows:

- `tau_0_p4.Pt() > 20 && abs(tau_0_q) == 1 && abs(tau_0_p4.Eta()) < 2.47`
- `n_bjets_DL1r_FixedCutBEff_70 == 0`
- `lep_0_iso_TightTrackOnly_VarRad == 1`
- `lephad_mt_lep0_met < 50`
- `lep_0_id_medium == 1`
- `lephad_met_sum_cos_dphi > -0.15`
- `lephad_p4.M() < 90 && lephad_p4.M() > 45`
- `lephad_dphi > 2.4`
- `abs(tau_0_p4.Eta() > 0.05)`
- `lephad_p4.M() < 85`

7.2 Data/Monte Carlo comparisons for kinematic variables

Comparing data and MC data on basic variables at first allows to set up the environment for later researches. With this, the reliability of the plotting framework can be ensured. Furthermore, the cuts to define the selection can be, if needed, adjusted, too, to obtain as many $Z \rightarrow \tau\tau$ events as possible while keeping the background small. This results in a as pure as possible selection. These region cuts are also included in 7.1 and are the last two ones listed. However, for this study, it is also demanded that the number of charged tracks is 1 which means that only the 1-prong region is examined.
Figure 7.1 shows how some variables agree to each other.



(a) Visible mass of tau

(b) Tau momentum

(c) Pseudorapidity

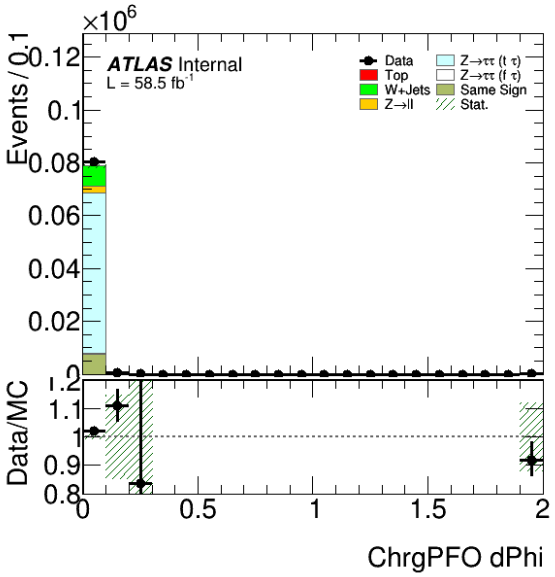
(d) Number of jets

Figure 7.1: Data/Monte Carlo comparison plots for selected variables; here, only the 1-prong region is regarded

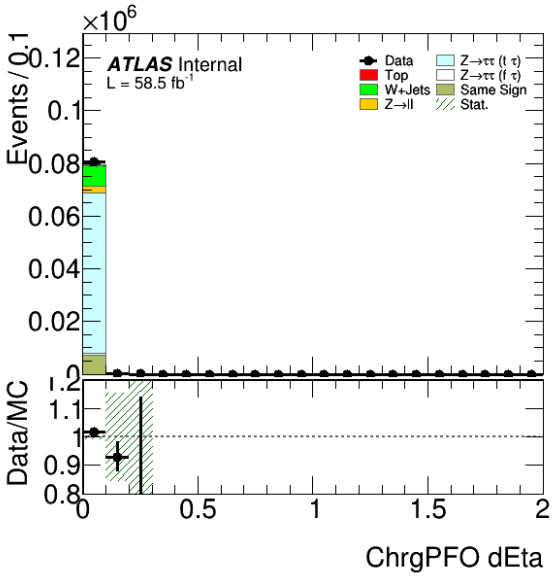
On the basis of the plot of the visible mass, it is quickly described what can be seen in the plot. It consists of two subplots. The upper plot depicts two histograms, one for the data and for the MC data. The latter one is also split up into the underlying processes in the simulated data. The lower plot depicts the ratio of data and MC data so that the exact agreement per bin can be read.
All in all, it can be concluded that for the shown variables, the agreement is acceptable. The deviation per bin is overall not higher than 10 %. Additionally, the second goal for plotting these variables, the purity of $Z \rightarrow \tau\tau$ events is now given, too.

7.3 Data/Monte Carlo comparisons for RNN input variables

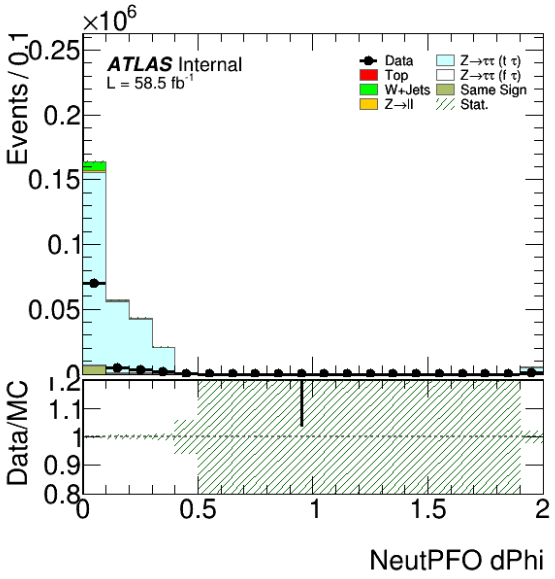
After completing setting up the environment for plotting, the input variables used in the RNN can be checked (compare Table 5.1). As their number amounts to 24, only a few representative ones are shown for each category, namely the angular distances. The plots can be seen in Figures 7.2 to 7.3.



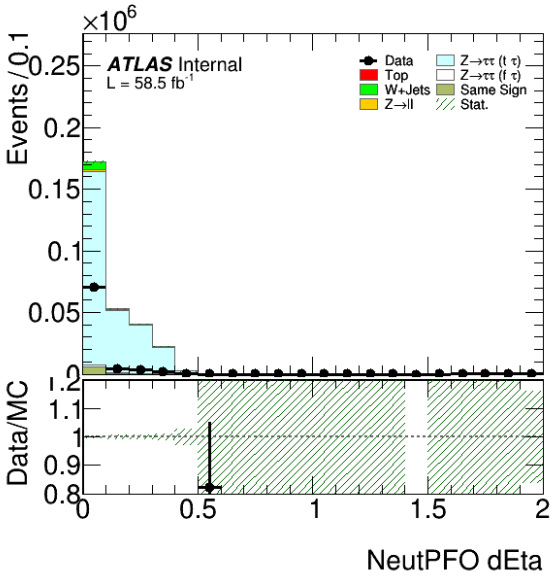
(a) transverse angular distance of charged PFOs



(b) longitudinal angular distance

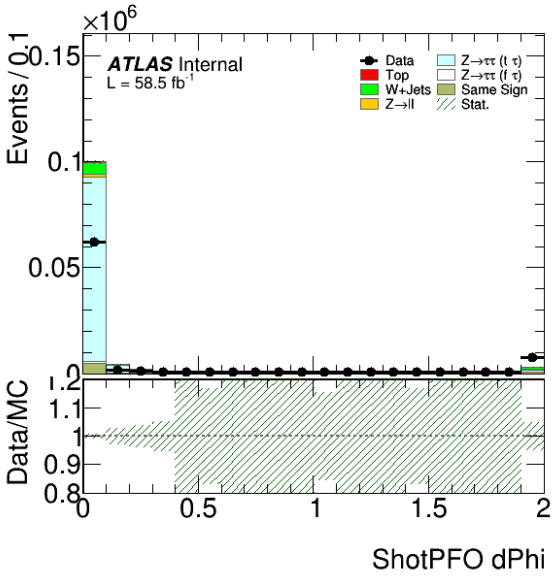


(c) transverse angular distance of neutral PFOs

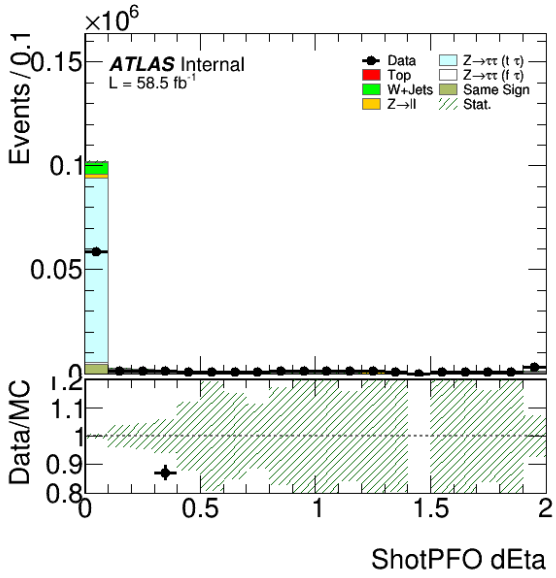


(d) longitudinal angular distance of neutral PFOs

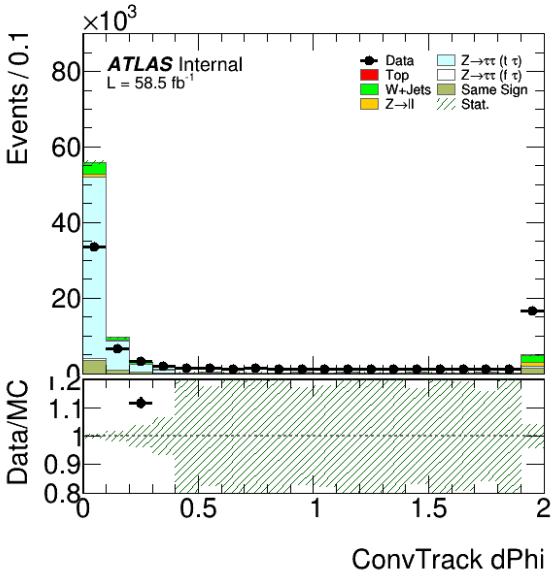
Figure 7.2: Data/Monte Carlo comparison plots for both angular distances used in charged and neutral PFOs; here, only the 1-prong region is regarded



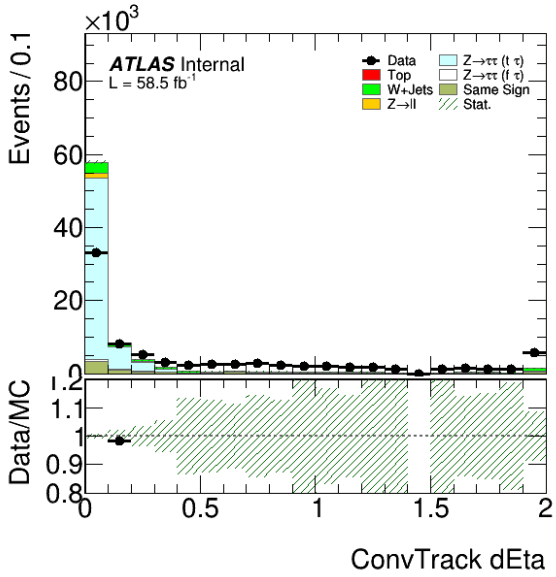
(a) transverse angular distance of photon shots



(b) longitudinal angular distance of photon shots



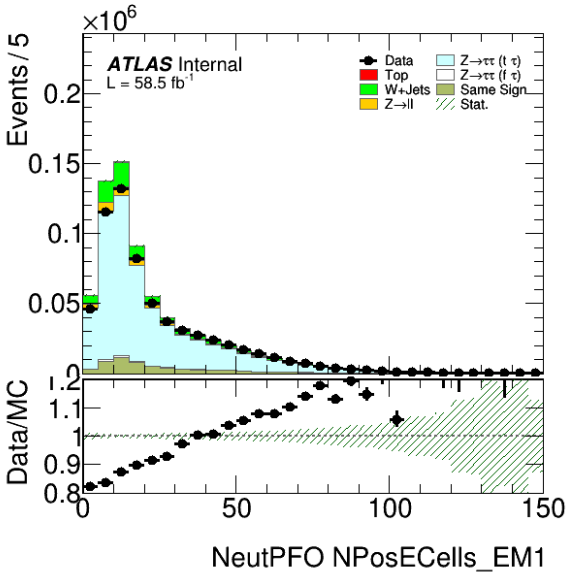
(c) transverse angular distance of conversion tracks



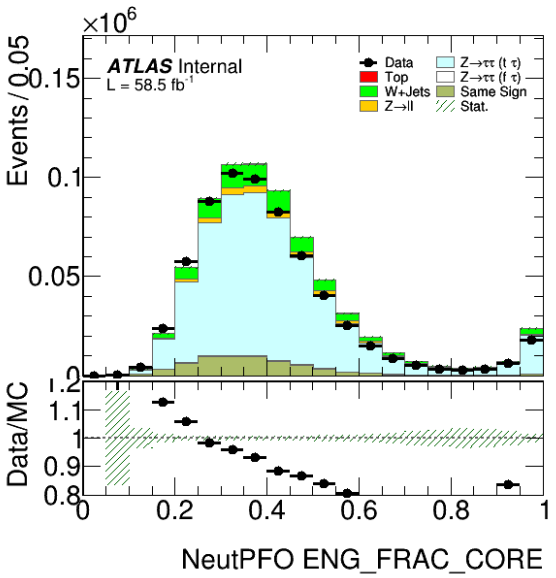
(d) longitudinal angular distance of conversion tracks

Figure 7.3: Data/Monte Carlo comparison plots for both angular distances used in photon shots and conversion tracks; here, only the 1-prong region is regarded

It is directly noticeable that, although the agreement for variables which are associated with charged PFOs seems to be good, the modelling for the remaining variables shows some problems. For them, there seems to be a mismatch between data and MC data. Considering the obtained knowledge in 6.3.2 that the contribution of cluster variables is higher than those of others, some more Data/Monte Carlo plots for neutral cluster variables are shown in Figure 7.4. Like before, these plots support the assumption that the mismodelling is not variable-dependent, but seems to be a general problem.



(a) number of cells with positive energy in EM1 (NPosECells_EM1)



(b) ENG_FRAC_CORE

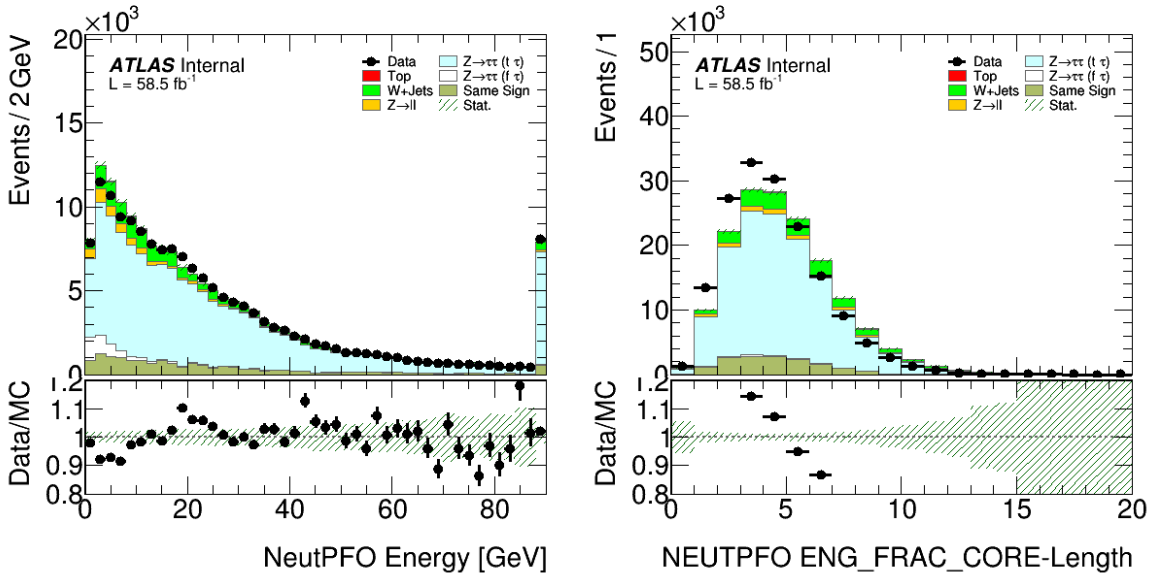
Figure 7.4: Data/Monte Carlo comparison plots for some more neutral cluster variables; here, only the 1-prong region is regarded

To get a better understanding of the source of this behaviour, some further tests were made. First, the cluster energy associated with neutral PFOs and the length of the vector, containing the entries for a cluster variable (in this case ENG_FRAC_CORE), were investigated. They are depicted in Figure 7.5.

As can be seen, the cluster energies between data and MC data seem to agree well. The mismatch is at least, if any, negligible. However, the vector length indicates that data has a tendency of having less cluster than MC data. This means that the problem could lie in the length of the vectors of cluster variables. To exclude the possibility that the entries itself do not pose a problem, they were checked as well (Figure 7.6). This was done for the variables SECOND_R and ENG_FRAC_CORE (compare Table 5.1).

The modelling for ENG_FRAC_CORE is in comparison to Figure 7.4(b) improved. Combining this fact with the plot of SECOND_R which looks good, the vector entries are, if at all, not the main source of these strange results.

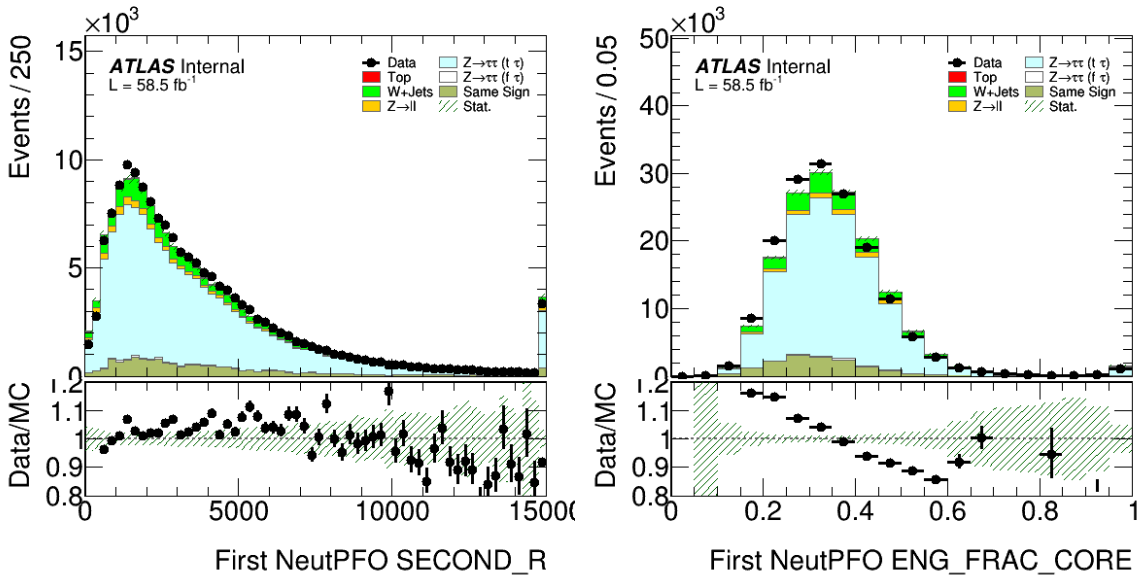
To conclude, Data/Monte Carlo comparisons show that there is a problem with the modelling of cluster variables apart from those associated with charged PFOs. There are signs that the source lies in the vector length of data and MC data because, plotting it, it was depicted that the number of clusters differs. However, due to time constraints, a solution cannot be presented in this work, but this should be nevertheless researched in future studies. The plotting framework as a problem source can be excluded, too, because, otherwise, such effects should have emerged in other plots as well. This is not the case though which points to a data-based problem.



(a) neutral cluster energy

(b) vector length(ENG_FRAC_CORE)

Figure 7.5: Data/Monte Carlo comparison plots the neutral cluster energy and and the vector length of ENG_FRAC_CORE; here, only the 1-prong region is regarded



(a) SECOND_R

(b) vector length(ENG_FRAC_CORE)

Figure 7.6: Data/Monte Carlo comparison plots for the first vector entry of SECOND_R and and ENG_FRAC_CORE; here, only the 1-prong region is regarded

CHAPTER 8

Validation

As concluded in Chapter 6, a recurrent neural network proves to have a superior performance in comparison to PanTau. However, unlike the latter algorithm, RNNs have only been evaluated on simulated data yet. Thus, it is not certain if the neural network is able to handle real experimental data or if the use of such data induces problems. The next step can be imagined as leaving laboratory conditions.
To verify that the studied RNN can be generalized to all kinds of data as long as the necessary variables are available, this chapter will discuss validation plots made by the same plotting framework as in Chapter 7. For this, [28] was used as an orientation.

8.1 Validating overall modelling

Since the RNN's task is mainly to classify decay modes, the focus during the studies mainly lied on this aspect. As described in 5.2, the output of the net is a *score* with probabilities for all five examined tau decay modes. It is known that the classification on simulated Monte Carlo data, which can be extracted from this output, is more efficient than PanTau. To find out if this is also the case for data obtained through the detector, the already trained net just needs to be evaluated on said data. It was already confirmed through Data/Monte Carlo comparisons that the modelling of the kinematic variables of both datasets, experimental and simulated data, mentioned in 7.1 have an acceptable agreement. Although this could unfortunately not be observed for the RNN's input variables except for the ones which are associated with charged PFOs, it was nevertheless attempted to evaluate the net on them. The reason behind this is that, first, it cannot be really said if there really is a problem with the data or not, and, second, if the modelling is faulty, a validation plot could give further evidence which could lead to a solution. Thus, the trained network was evaluated on both datasets, leading to two classification outputs. These output scores can be compared similar as in Chapter 7. The result can be seen in Figure 8.1.

Both figures are the same plot, but with the difference that, in the right one, the $Z \rightarrow \tau\tau$ process is split into different tau decay modes according to the labelled tau decay mode classification in MC data. With this, it can be, similar to a migration matrix, determined which fractions of the reconstructed decay modes are correctly classified and which are misclassified. For example, it can be seen that the bar at 1p1n has a yellow fraction which is denoted as "$Z \rightarrow \tau\tau$ 1p1n". This yellow fraction shall depict true 1p1n events meaning that the net classified these taus correctly. However, the light blue

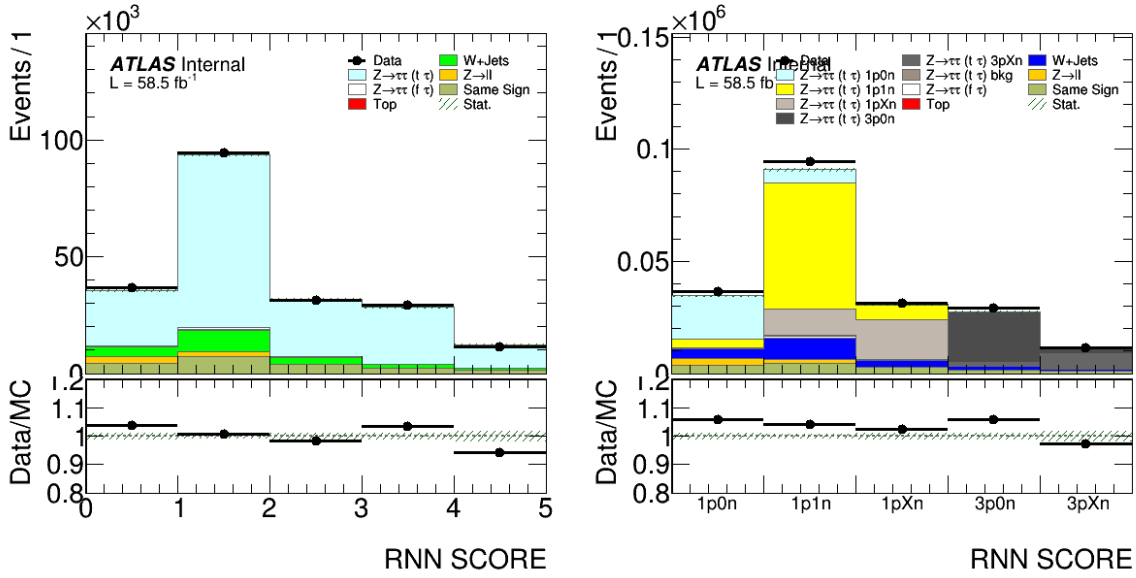


Figure 8.1: Comparison of reconstructed decay modes on experimental and simulated data by the RNN

section which is representative for 1p0n were also classified to 1p1n by the RNN, thus, showing that these taus were misclassified. It should be mentioned that the contribution called $Z \rightarrow \tau\tau(t\tau)$bkg denotes all events with other decay modes as the ones which are studied.
It is important to recognize that these results show that the studied RNN seems to be able to handle experimental data and to give an classification which fits the true decay modes according to the MC data. Furthermore, the output is also consistent to former results. If the net's performance is still comparable to former studies like in Chapter 6, it can be seen in Figure 6.9 that most of the tau candidates are classified correctly. However, for example, a third of all 1pXn decays are misclassified as 1p1n. This effect can also be observed in Figure 8.1.
Based on these results, the following can be concluded:

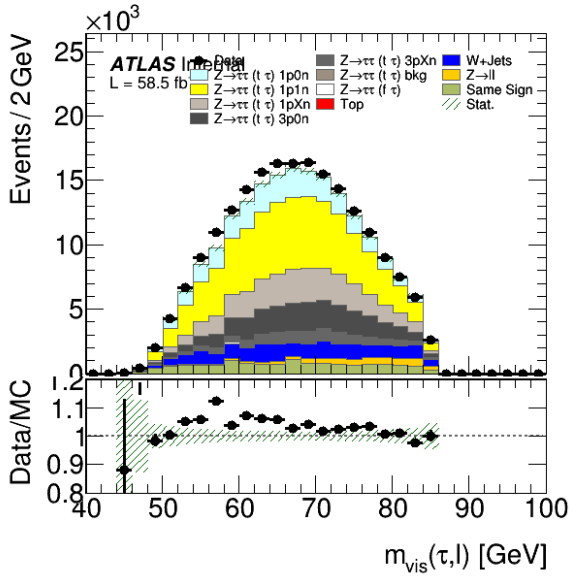
- A trained recurrent neural network like the one which was studied in this work is able to receive experimental data and, based on this, can perform tau decay mode classification with a performance which is indicated to be superior to `PanTau`.

- This performance does not seem to be affected by the experienced modelling problems. This could mean two things: There is either no problem with the dataset, implying that the plotting framework is faulty or at least cannot handle cluster variables after all, or, on average, the various issues cancel each other out so that, in the picture, there are no problems. However, these assumptions are still speculation and to find the truth, this has to be investigated in the future. Nevertheless, it can be said that the mismodelling does not limit the performance right now.

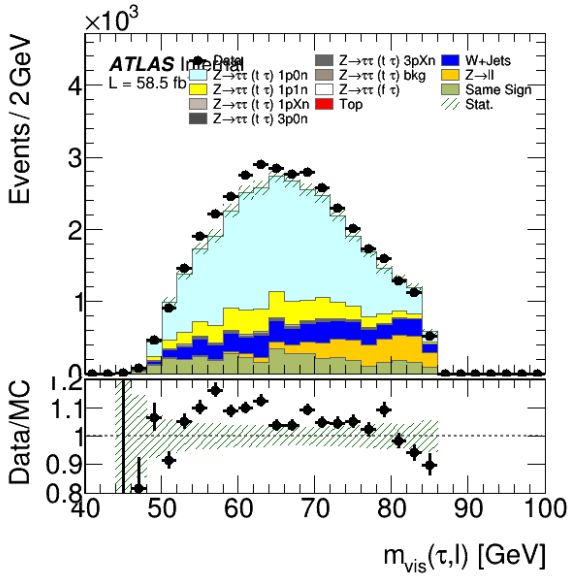
8.2 Validating the output of the RNN on variables

The output of the RNN can also be validated on the input variables. However, since the mismatching problem still persists, the plots for the visible mass will be used to discuss this. It can be seen in 8.2. These plots were generated by cutting on the RNN classification output. With this, it can be evaluated if the data and MC data are treated ny the RNN in the same way. For the visible mass, it can be, first, observed that the agreement of data and MC fits for all cases although it gets worse for the 3pXn case. The splitting of the $Z \rightarrow \tau\tau$ part is also to be expected since, for each cut on the classification output, most of the events are really taus with the respective decay move according to the MC data.

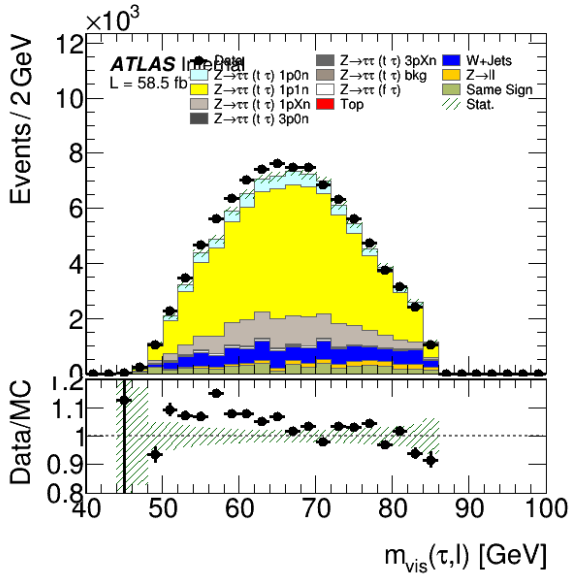
This can be, of course, done for all input variables. The respective plots for all of them can be found in Appendix 9.4.



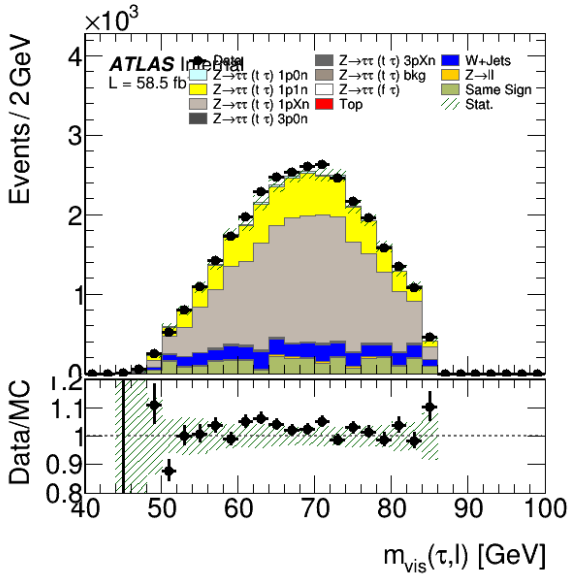
(a) before applying the classification



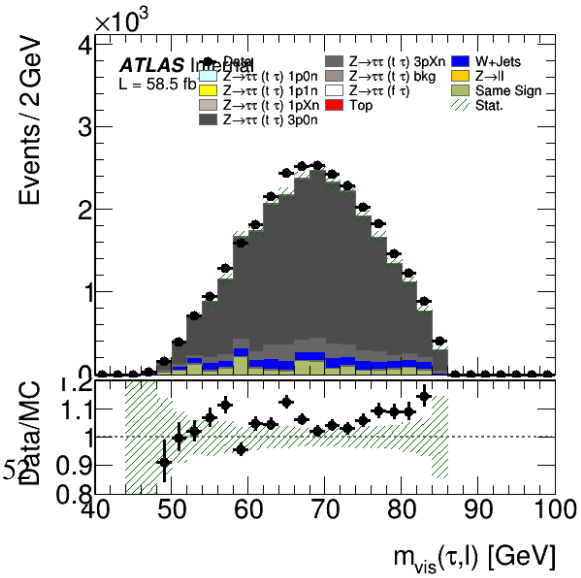
(b) 1p0n



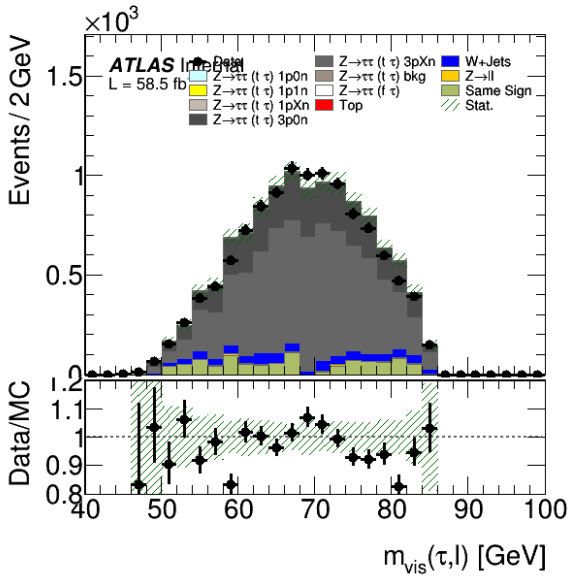
(c) 1p1n



(d) 1pXn



(e) 3p1n



(f) 3pXn

Figure 8.2: Visible mass

CHAPTER 9

Summary and outlook

The performance of classifying hadronic tau decays is important for many physics analyses to be accurate. PanTau is already used successfully to perform this task, but since neural networks are well-suited for classification tasks, it should be considered to pursue this idea further. This is supported by the results achieved by this thesis. As a continuation of previous studies regarding this topic, the focus of this thesis, first, lied on getting further insights and possibly improving the used neural net.
First studies towards optimizing the RNN were made by performing a variable ranking. During this process, it was observed that cluster variables seem to contain insightful information regarding the neural net's task since omitting them leads to a significant decrease of performance. However, coordinate variables like phi and eta, regardless of the input variable category, do not have a notable impact. Thus, the complexity could be simplfied by reducing the number of variables by 8 from 32 to 24 which means a cut of the number of input variables by 25 %. Although using lesser variables, the performance did not change, indicating that the former variable set was not optimal.
Moreover, it was examined if the efficiency can be further improved by choosing another set of hyperparameters. For this, hyperparameter optimization based on Bayesian optimization was performed. However, although the resulting new set of hyperparameters leads to a better efficiency, no clear pattern for choosing the optimal set could be observed. On top of that, the improvement was 0.2 %. So, although the performance could be improved even more, other optimization methods may have a bigger impact.
Since there are numerous ways to improve the performance, it can be more resourceful to set a greater focus on understanding the neural network. Until now, it is only known that it performs better and methods to get better results are based on structure-related changes. Another ansatz is to understand what specific information leads to an improvement and to concentrate on the provided data. This will also help understanding the neural network and the examined physical process, too.
Apart from attempts to improve the neural net by itself, first evidence that an RNN is suitable to classify tau decay modes could be seen in a second step. Evaluating the trained neural net on real experimental data has shown promising results. The performance is comparable to evaluations on only simulated data. This hints that the RNN is able to handle real-world data. If applied in the future, this would mean an improvement of classification accuracy which can lead to improved energy calibrations used for hadronic tau decays. This results to more accurate mass reconstructions of tau resonances which again improves the signal extraction processes in analyses.
Unfortunately, a problem with the modelling of cluster variables occured. The source of the observed

effect is still not known, but since the variable ranking indicated that these variables have a big impact on the training of the neural network, tests regarding this problem should be performed in the future.

References

[1] MissMJ, *Standard Model of Elementary Particles*, [Online; accessed 23.03.2021], 2019, URL: https://commons.wikimedia.org/wiki/File:Standard_Model_of_Elementary_Particles.svg (cit. on p. 4).

[2] P.A. Zyla et al. (Particle Data Group), *Review of Particle Physics*, Progress of Theoretical and Experimental Physics **2020** (2020), 083C01, ISSN: 2050-3911, eprint: https://academic.oup.com/ptep/article-pdf/2020/8/083C01/33653179/ptaa104.pdf, URL: https://doi.org/10.1093/ptep/ptaa104 (cit. on pp. 3–5).

[3] S. Glashow, *Partial Symmetries of Weak Interactions*, Nucl. Phys. **22** (1961) 579; A. Salam, "Weak and Electromagnetic Interactions", *Elementary particle theory. Relativistic groups and analyticity. Proceedings of the Eighth Nobel Symposium*, ed. by N. Svartholm, Stockholm: Almquist & Wiksell, 1968 367; S. Weinberg, *A Model of Leptons*, Phys. Rev. Lett. **19** (1967) 1264, cit. on p. 4.

[4] M. L. P. et al., *Evidence for Anomalous Lepton Production in $e^+ - e^-$ Annihilation*, Phys. Rev. Lett. **35** (22 1975) 1489, URL: https://link.aps.org/doi/10.1103/PhysRevLett.35.1489 (cit. on p. 5).

[5] *Lessons learnt from the heavy tau lepton*, URL: https://cerncourier.com/a/lessons-learnt-from-the-heavy-tau-lepton/ (cit. on p. 7).

[6] E. Mobs, *The CERN accelerator complex - 2019. Complexe des accélérateurs du CERN - 2019*, (2019), General Photo, URL: https://cds.cern.ch/record/2684277 (cit. on p. 10).

[7] J. Pequenao, "Computer generated image of the whole ATLAS detector", 2008, URL: https://cds.cern.ch/record/1095924 (cit. on p. 11).

[8] A. collaboration, *The ATLAS Experiment at the CERN Large Hadron Collider*, JINST **3** (2008) S08003. 437 p, Also published by CERN Geneva in 2010, URL: https://cds.cern.ch/record/1129811 (cit. on p. 12).

[9] A. Collaboration, *Performance of the ATLAS Trigger System in 2015. Performance of the ATLAS Trigger System in 2015*, Eur. Phys. J. C **77** (2016) 317. 76 p, 77 pages in total, author list starting page 61, 50 figures, 1 table. Published in Eur. Phys. J. C. All figures including auxiliary figures are available at http://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/PAPERS/TRIG-2016-01/, URL: https://cds.cern.ch/record/2235584 (cit. on p. 14).

[10] P. Mathur, *A Simple Multilayer Perceptron with TensorFlow*, URL: https://medium.com/pankajmathur/a-simple-multilayer-perceptron-with-tensorflow-3effe7bf3466 (cit. on p. 16).

[11] N. Fronzetti, *Predictive Neural Network Applications for Insurance Processes*, 2019 (cit. on p. 16).

[12] R. Rojas, *Neural Networks - A Systematic Introduction* (cit. on p. 17).

[13] S. Ruder, *An overview of gradient descent optimization algorithms*, 2016, arXiv: 1609.04747 [cs.LG] (cit. on p. 17).

[14] D. P. Kingma and J. Ba, *Adam: A Method for Stochastic Optimization*, 2014, arXiv: 1412.6980 [cs.LG] (cit. on p. 17).

[15] X. Gao et al., *Recurrent neural networks for real-time prediction of TBM operating parameters*, Automation in Construction **15** (2019) 130 (cit. on p. 18).

[16] Y. LeCun, Y. Bengio and G. Hinton, *Deep Learning*, Nature **521** (2015) 436 (cit. on p. 20).

[17] S. Hochreiter, *Untersuchungen zu dynamischen neuronalen Netzen*, (1991) (cit. on p. 20).

[18] Y. Bengio, P. Simard and P. Frasconi, *Learning long-term dependencies with gradient descent is difficult*, IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council **5** (1994) 157 (cit. on p. 20).

[19] S. Hochreiter and J. Schmidhuber, *Long Short-term Memory*, Neural computation **9** (1997) 1735 (cit. on p. 20).

[20] X. Yuan, L. Li and Y. Wang, *Nonlinear dynamic soft sensor modeling with supervised long short-term memory network*, IEEE Transactions on Industrial Informatics **PP** (2019) 1 (cit. on p. 21).

[21] C. Deutsch, *Identification and Classification of Hadronic Tau Lepton Decays in the ATLAS Experiment for Run 2 pf the LHC*, MA thesis, 2017 (cit. on pp. 23, 28).

[22] C. Deutsch, URL: https://test-rnn-tauid-docs.web.cern.ch/test-rnn-rauid-docs/decaymode-workflow (cit. on pp. 23, 32, 33).

[23] B. T. Winter, *Reconstruction of neutral pions in hadronic tau lepton decays in the ATLAS detector*, Presented on Jan 2014, 2013, URL: http://cds.cern.ch/record/2318236 (cit. on p. 28).

[24] H. Li et al., *Visualizing the Loss Landscape of Neural Nets*, (2017), arXiv: 1712.09913 [cs.LG] (cit. on p. 32).

[25] J. Snoek, H. Larochelle and R. P. Adams, *Practical Bayesian Optimization of Machine Learning Algorithms*, 2012, arXiv: 1206.2944 [stat.ML] (cit. on p. 36).

[26] B. Shahriari et al., *Taking the Human Out of the Loop: A Review of Bayesian Optimization*, Proceedings of the IEEE **104** (2016) 148 (cit. on p. 36).

[27] S. Amoroso et al., *PMG references document*, tech. rep. ATL-COM-PHYS-2019-701, CERN, 2019, URL: https://cds.cern.ch/record/2678867 (cit. on p. 41).

[28] G. Aad et al., *Reconstruction of hadronic decay products of tau leptons with the ATLAS experiment*, The European Physical Journal C **76** (2016), ISSN: 1434-6052, URL: http://dx.doi.org/10.1140/epjc/s10052-016-4110-0 (cit. on p. 49).

9.1 Appendix

9.2 Datasets

9.2.1 Sample for RNN-based decay mode classification

Sample
mc16_13TeV.452000.Pythia8EvtGen_A14NNPDF23LO_Gammatautau_MassWeight.merge.AOD.e5468_s3123_r10201_r10210

9.2.2 Samples used for Data/Monte Carlo comparisons

Sample
group.perf-tau.v11.data18_13TeV.periodB.physics_Main.P3.grp18_v01_p4113.20_5_18_2_Ta.root
group.perf-tau.v11.data18_13TeV.periodC.physics_Main.P3.grp18_v01_p4113.20_5_18_2_Ta.root
group.perf-tau.v11.data18_13TeV.periodD.physics_Main.P3.grp18_v01_p4113.20_5_18_2_Ta.root
group.perf-tau.v11.data18_13TeV.periodF.physics_Main.P3.grp18_v01_p4113.20_5_18_2_Ta.root
group.perf-tau.v11.data18_13TeV.periodI.physics_Main.P3.grp18_v01_p4113.20_5_18_2_Ta.root
group.perf-tau.v11.data18_13TeV.periodK.physics_Main.P3.grp18_v01_p4113.20_5_18_2_Ta.root
group.perf-tau.v11.data18_13TeV.periodL.physics_Main.P3.grp18_v01_p4113.20_5_18_2_Ta.root
group.perf-tau.v11.data18_13TeV.periodM.physics_Main.P3.grp18_v01_p4113.20_5_18_2_Ta.root
group.perf-tau.v11.data18_13TeV.periodO.physics_Main.P3.grp18_v01_p4113.20_5_18_2_Ta.root
group.perf-tau.v11.data18_13TeV.periodQ.physics_Main.P3.grp18_v01_p4113.20_5_18_2_Ta.root

Table 9.1: Data from the v11 mu+tau ntuples

Sample
group.perf-tau.v11.mc16_13TeV.361100.PoPy8_Wplusenu.P3.e3601_s3126_r10724_p4112.20_5_18_2_Ta.root
group.perf-tau.v11.mc16_13TeV.361101.PoPy8_Wplusmunu.P3.e3601_s3126_r10724_p4112.20_5_18_2_Ta.root
group.perf-tau.v11.mc16_13TeV.361102.PoPy8_Wplustaunu.P3.e3601_s3126_r10724_p4112.20_5_18_2_Ta.root
group.perf-tau.v11.mc16_13TeV.361103.PoPy8_Wminusenu.P3.e3601_s3126_r10724_p4112.20_5_18_2_Ta.root
group.perf-tau.v11.mc16_13TeV.361104.PoPy8_Wminusmunu.P3.e3601_s3126_r10724_p4112.20_5_18_2_Ta.root
group.perf-tau.v11.mc16_13TeV.361105.PoPy8_Wminustaunu.P3.e3601_s3126_r10724_p4112.20_5_18_2_Ta.root
group.perf-tau.v11.mc16_13TeV.361106.PoPy8_Zee.P3.e3601_s3126_r10724_p4112.20_5_18_2_Ta.root
group.perf-tau.v11.mc16_13TeV.361107.PoPy8_Zmumu.P3.e3601_s3126_r10724_p4112.20_5_18_2_Ta.root
group.perf-tau.v11.mc16_13TeV.361108.PoPy8_Ztt.P3.e3601_s3126_r10724_p4112.20_5_18_2_Ta.root
group.perf-tau.v11.mc16_13TeV.410470.PhPy8_A14_ttb_nonallh.P3.e6337_s3126_r10724_p4112.20_5_18_2_Ta.root
group.perf-tau.v11.mc16_13TeV.410471.PhPy8_A14_ttb_allh.P3.e6337_s3126_r10724_p4112.20_5_18_2_Ta.root
group.perf-tau.v11.mc16_13TeV.410644.PoPy8_A14_st_schan_lept_top.P3.e6527_s3126_r10724_p4112.20_5_18_2_Ta.root
group.perf-tau.v11.mc16_13TeV.410645.PoPy8_A14_st_schan_lept_atop.P3.e6527_s3126_r10724_p4112.20_5_18_2_Ta.root
group.perf-tau.v11.mc16_13TeV.410646.PoPy8_A14_Wt_DR_inclusive_top.P3.e6552_s3126_r10724_p4112.20_5_18_2_Ta.root
group.perf-tau.v11.mc16_13TeV.410647.PoPy8_A14_Wt_DR_inclusive_atop.P3.e6552_s3126_r10724_p4112.20_5_18_2_Ta.root
group.perf-tau.v11.mc16_13TeV.410658.PhPy8_A14_tchan_BW50_lept_top.P3.e6671_s3126_r10724_p4112.20_5_18_2_Ta.root
group.perf-tau.v11.mc16_13TeV.410659.PhPy8_A14_tchan_BW50_lept_atop.P3.e6671_s3126_r10724_p4112.20_5_18_2_Ta.root

Table 9.2: Used Monte Carlo datasets from the v11 mu+tau ntuples

9.3 Full workflow of PanTau

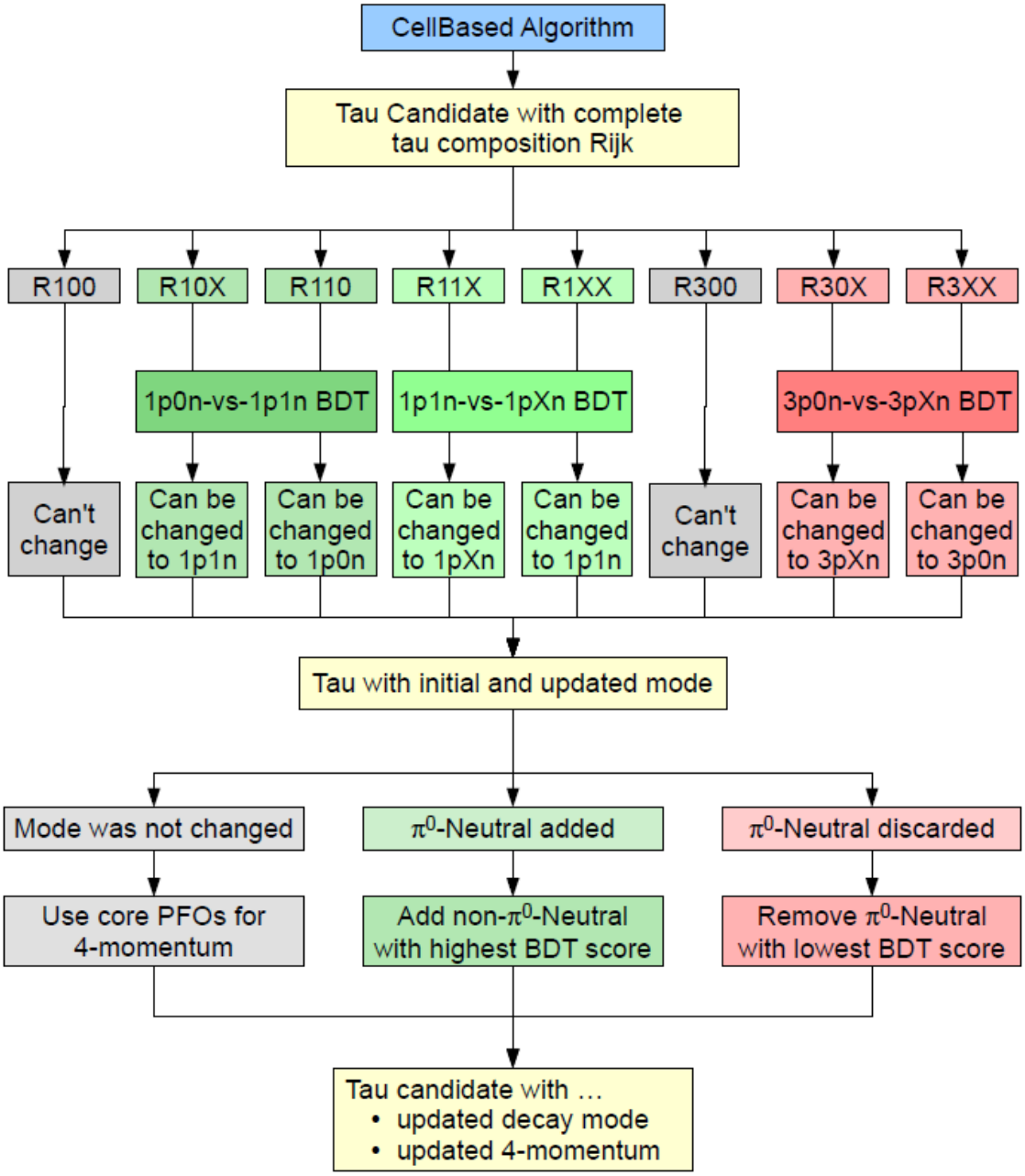


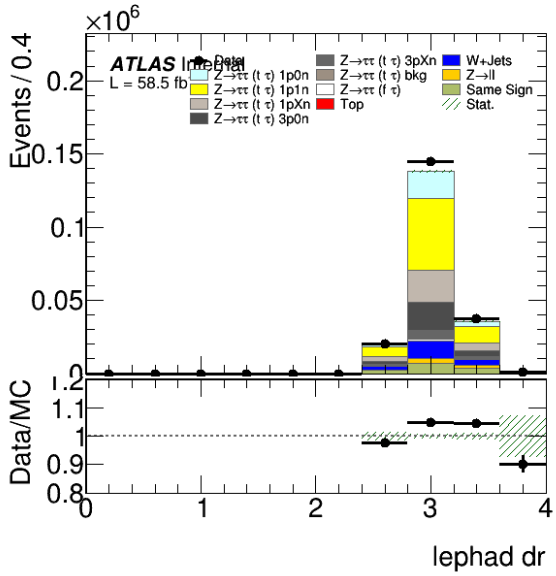
Figure 9.1: Full workflow of PanTau including updating the 4-momentum of the examined tau

9.4 Validation plots for all used varables

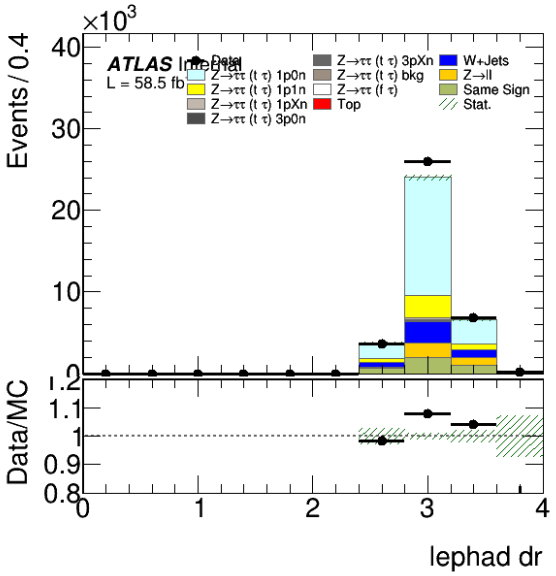
This section contains validation plots of all input variables. For each, there are are six plots. The first one depicts the unmodified distribution while the rest are plotted according to the classification made by the RNN. The scores are mapped like follows:

- 0 → 1p0n
- 1 → 1p1n
- 2 → 1pXn
- 3 → 3p1n
- 4 → 3pXn

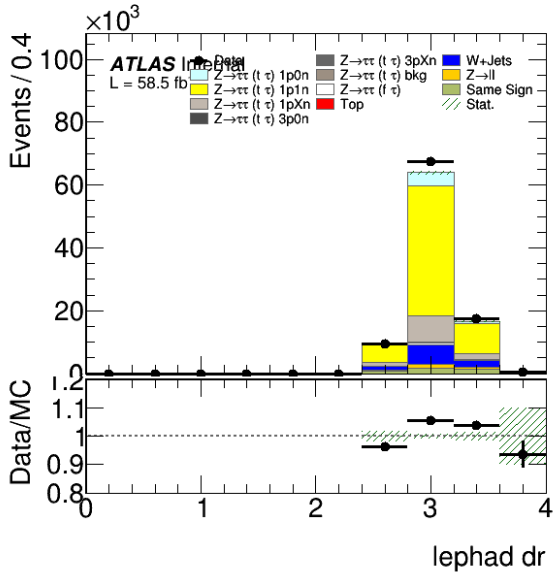
So, for example, a figure with the caption "RNN score == 0" means that, for this plot, the cut `RNN score == 0` was applied.



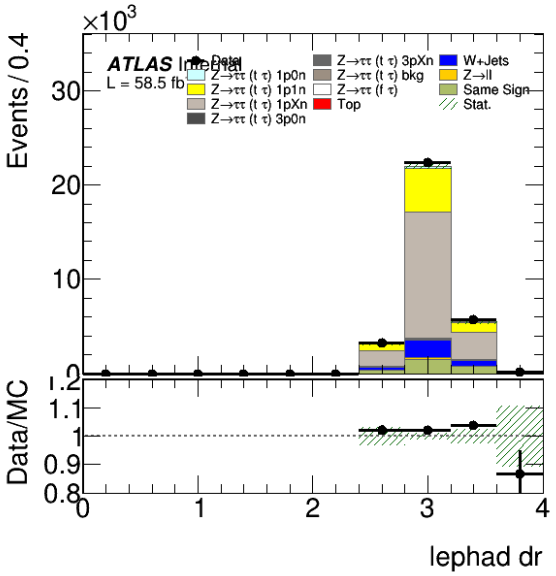
(a) no cut on RNN score



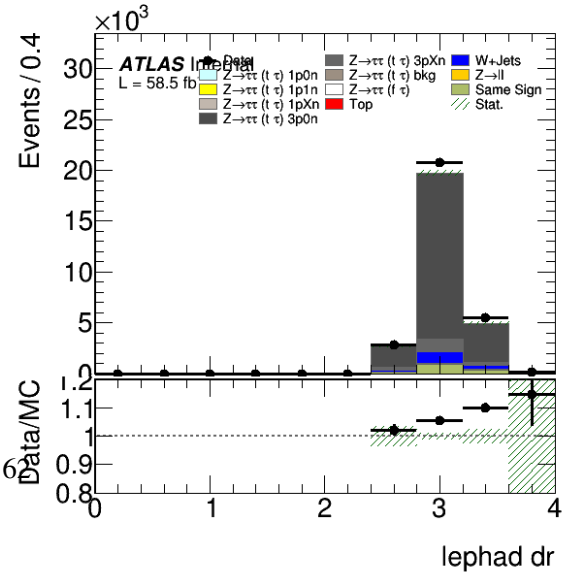
(b) RNN score == 0



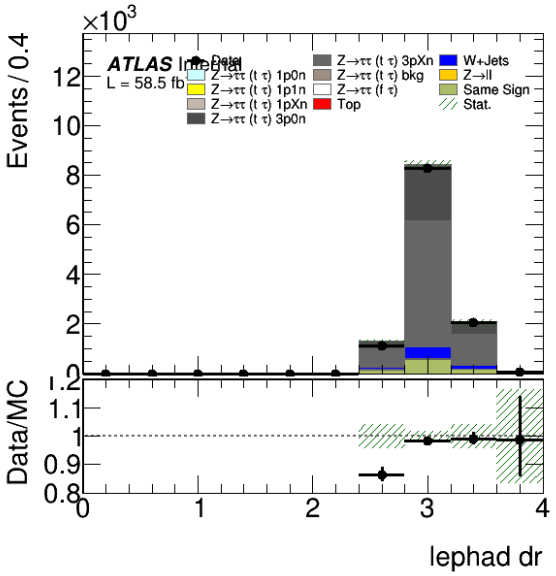
(c) RNN score == 1



(d) RNN score == 2

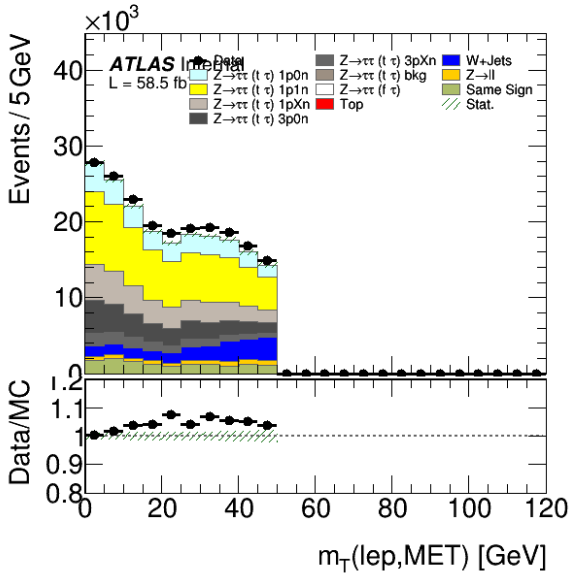


(e) RNN score == 3

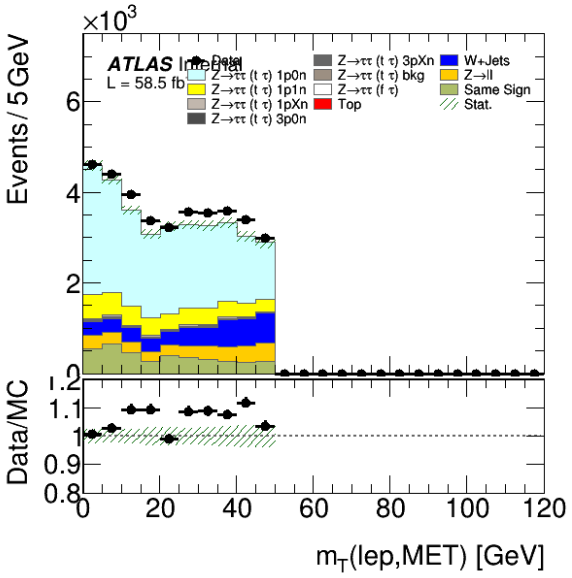


(f) RNN score == 4

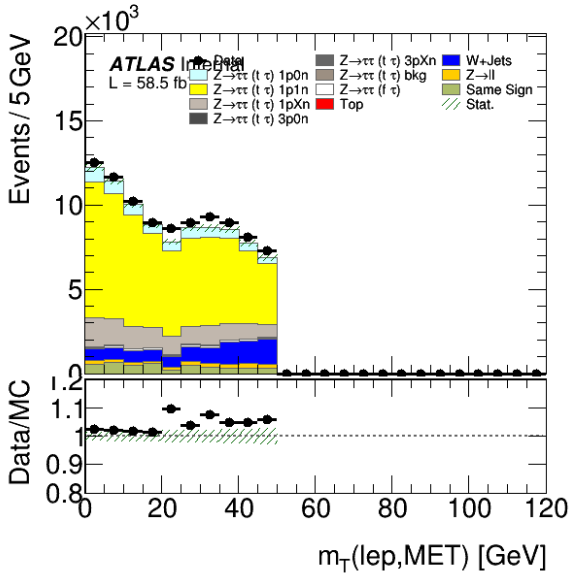
Figure 9.2



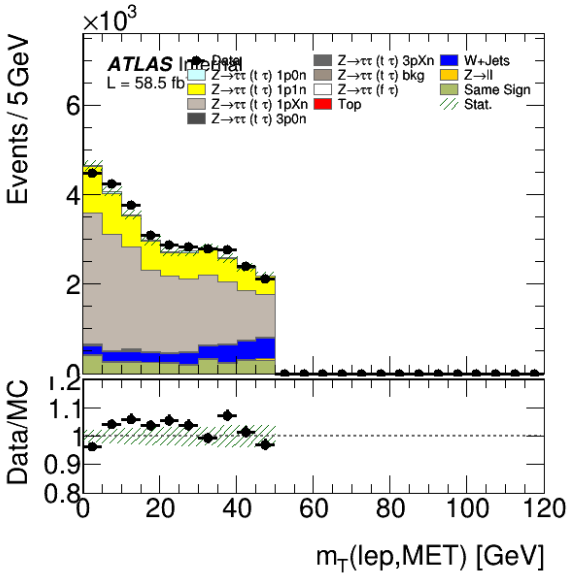
(a) no cut on RNN score



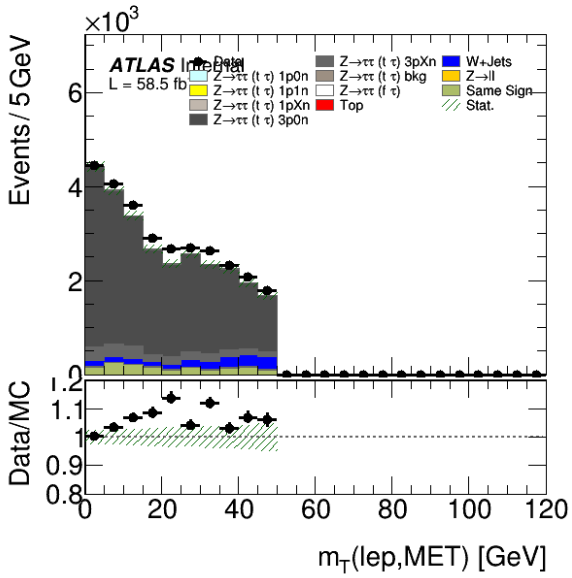
(b) RNN score == 0



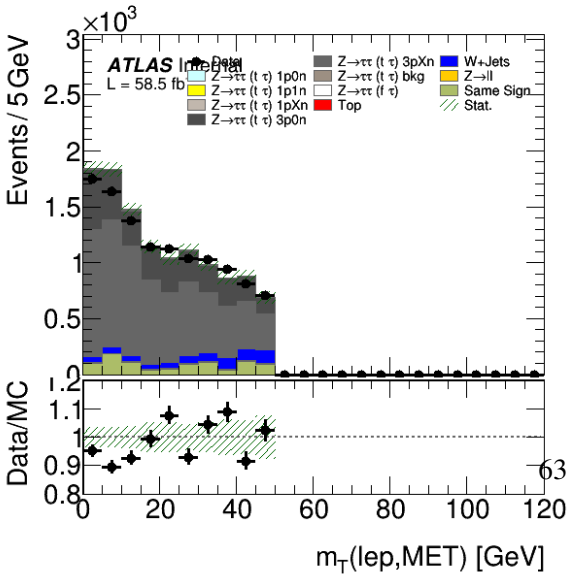
(c) RNN score == 1



(d) RNN score == 2

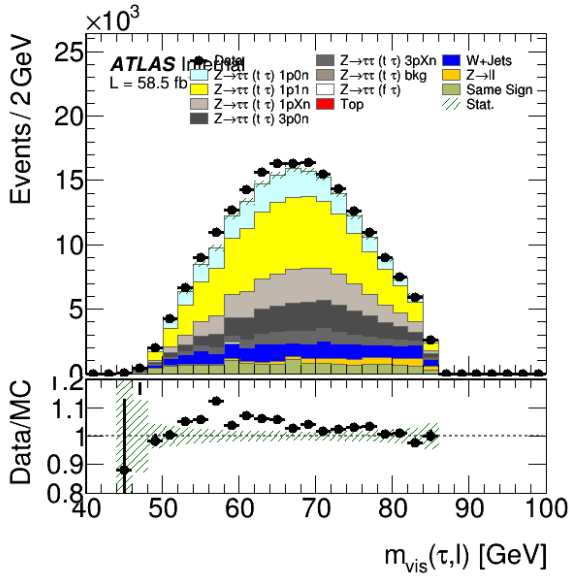


(e) RNN score == 3

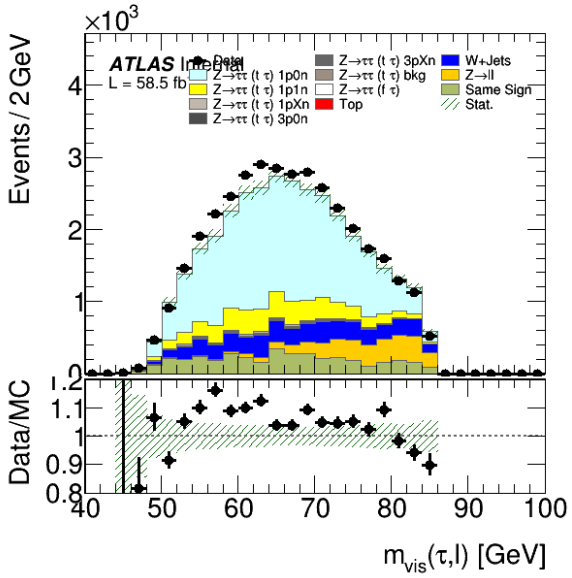


(f) RNN score == 4

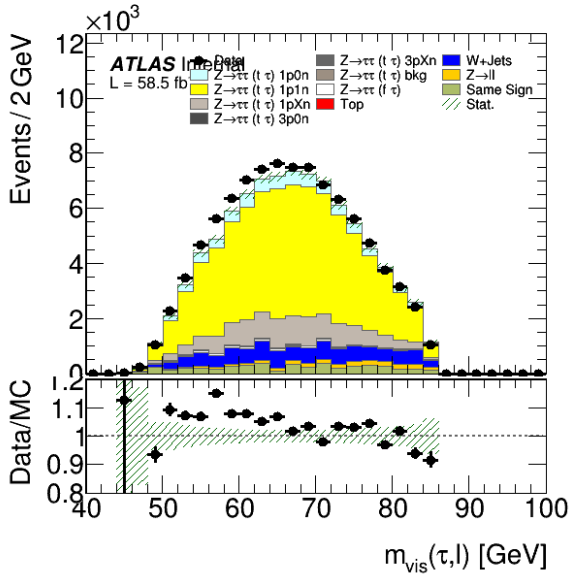
Figure 9.3



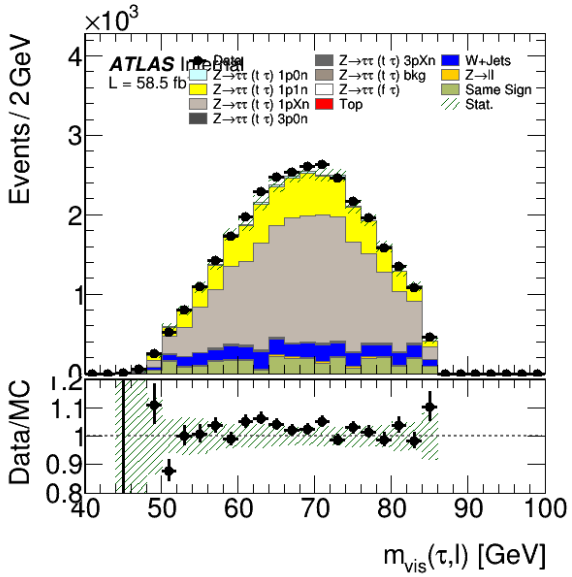
(a) no cut on RNN score



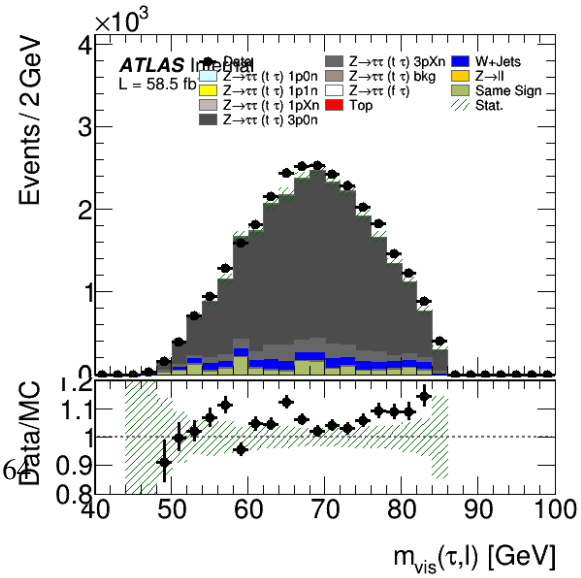
(b) RNN score == 0



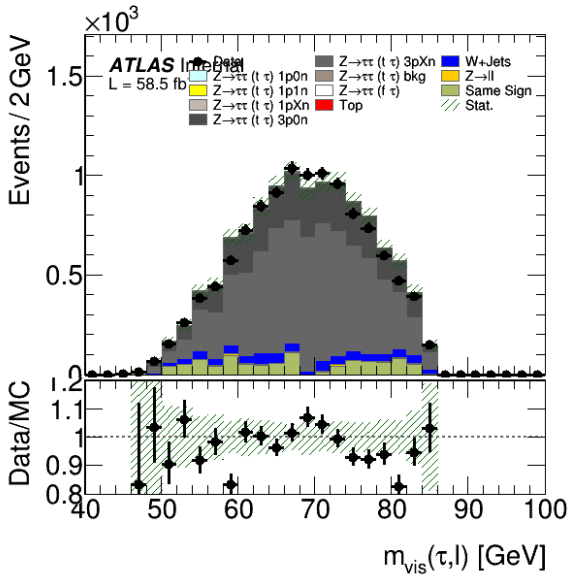
(c) RNN score == 1



(d) RNN score == 2

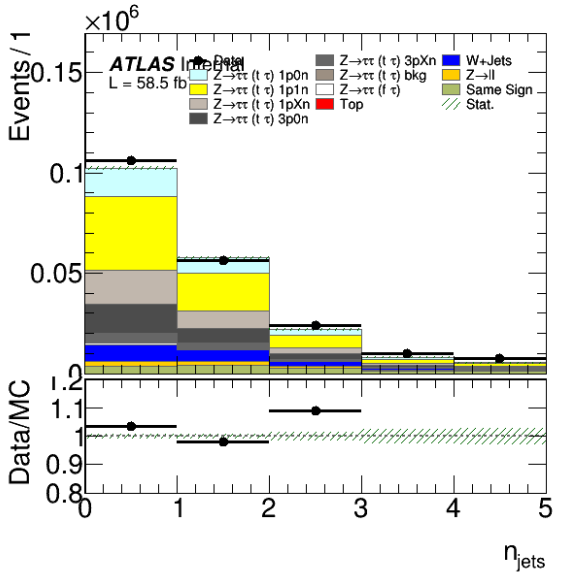


(e) RNN score == 3

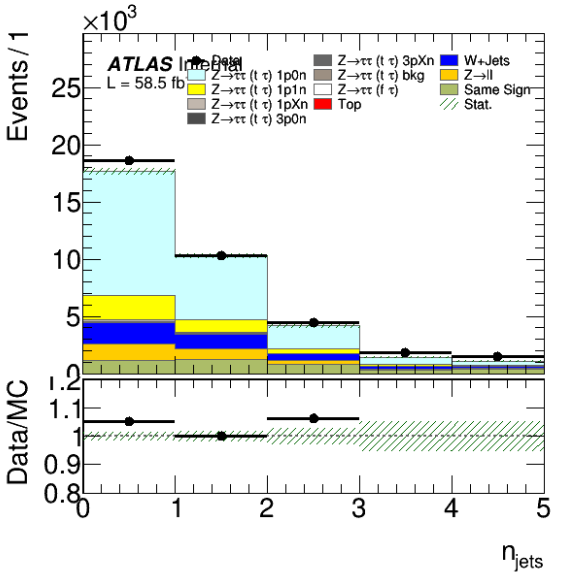


(f) RNN score == 4

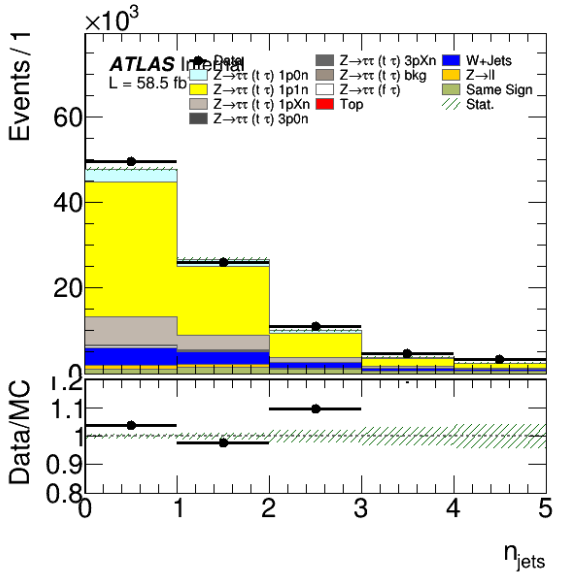
Figure 9.4: Visible mass



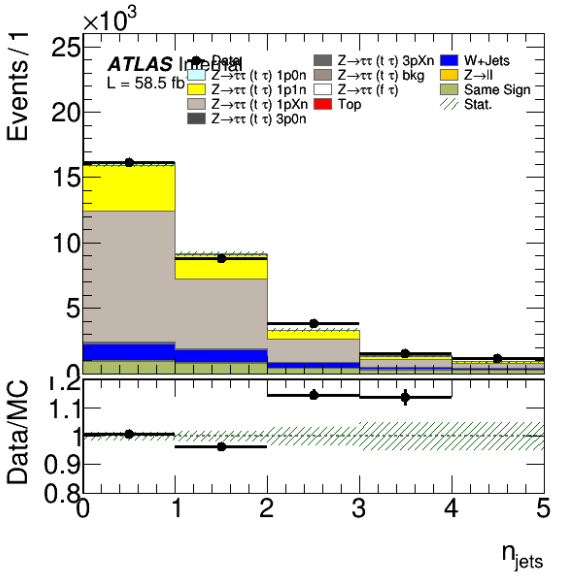
(a) no cut on RNN score



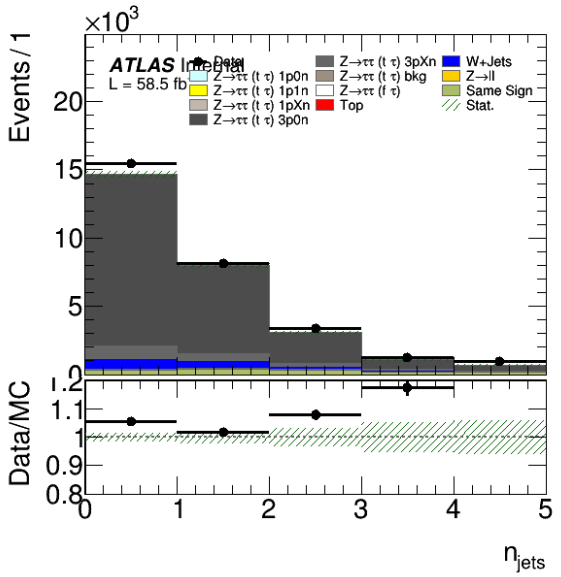
(b) RNN score == 0



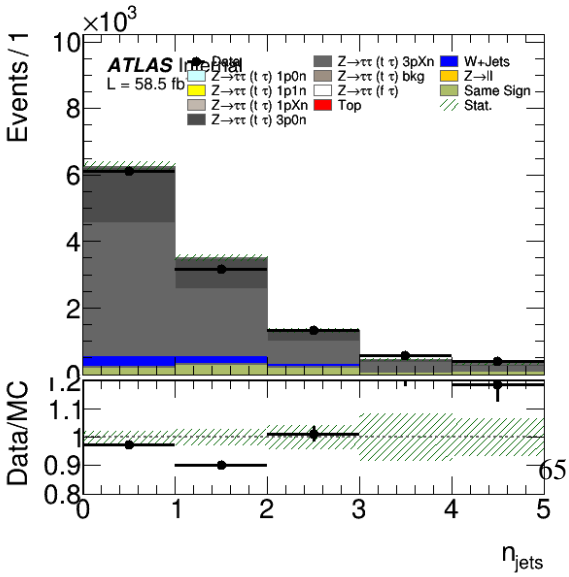
(c) RNN score == 1



(d) RNN score == 2

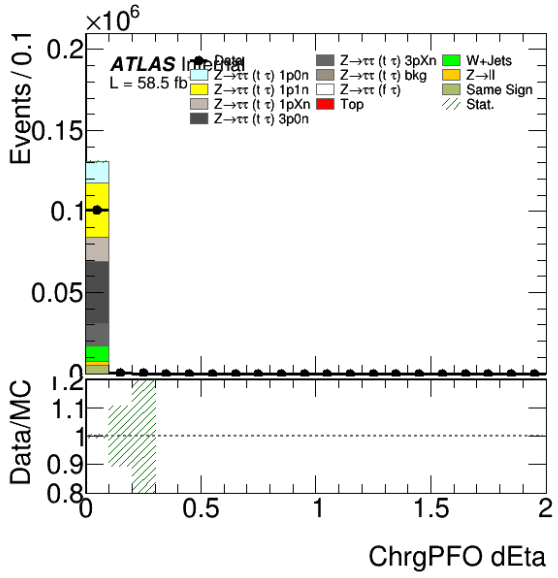


(e) RNN score == 3

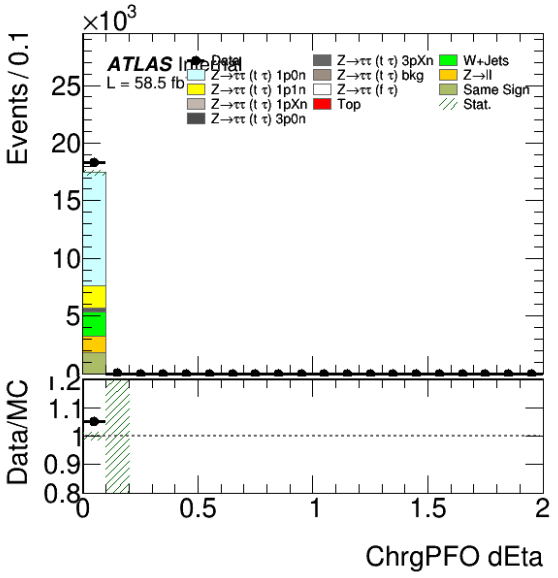


(f) RNN score == 4

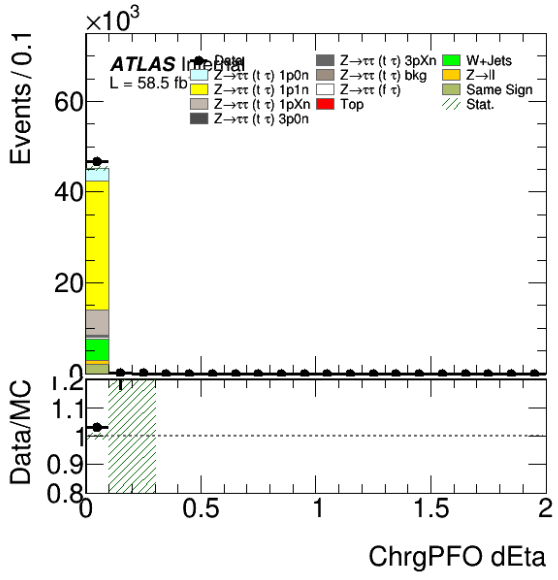
Figure 9.5: Number of jets



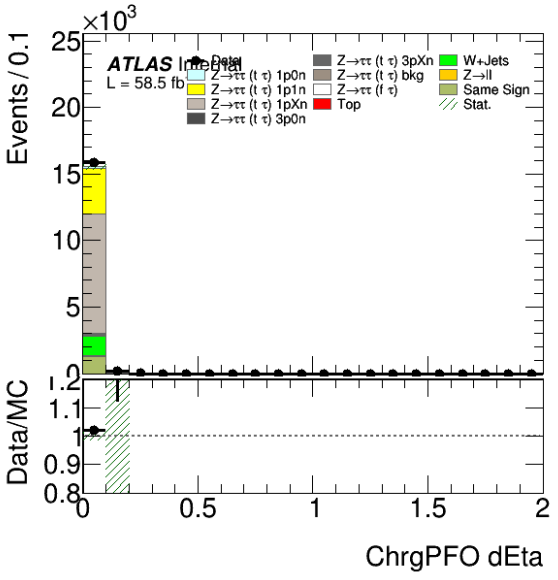
(a) no cut on RNN score



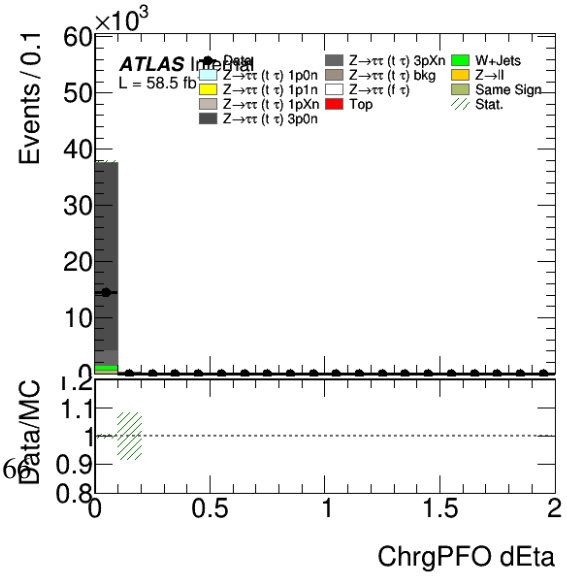
(b) RNN score == 0



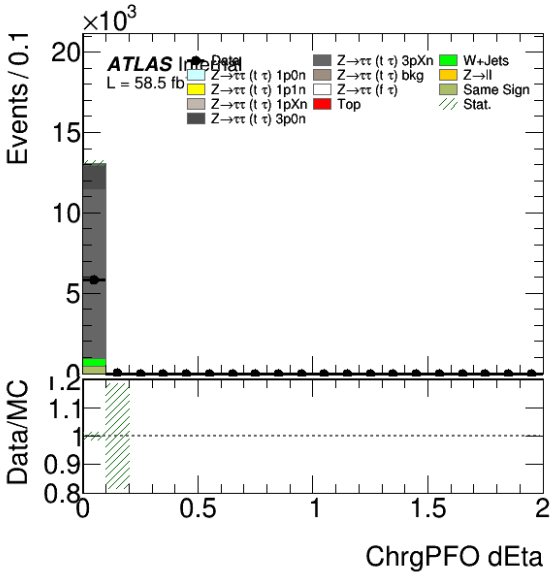
(c) RNN score == 1



(d) RNN score == 2

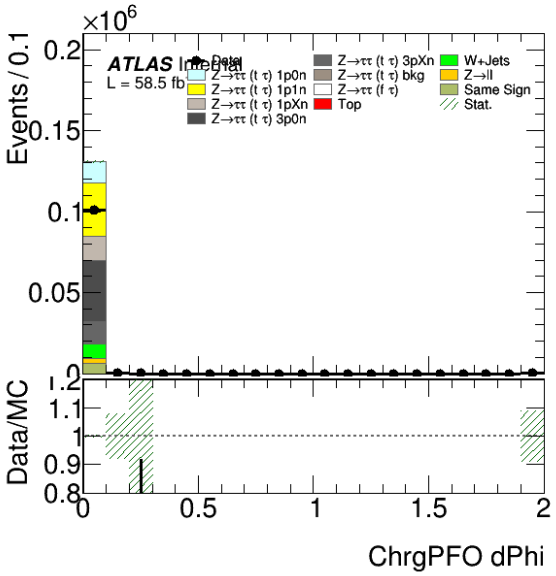


(e) RNN score == 3

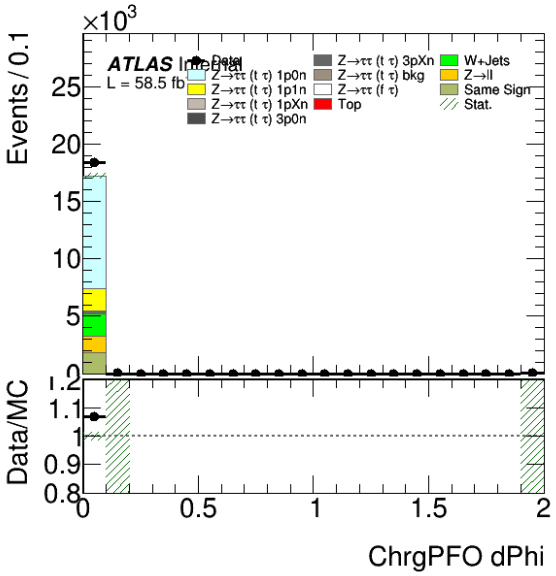


(f) RNN score == 4

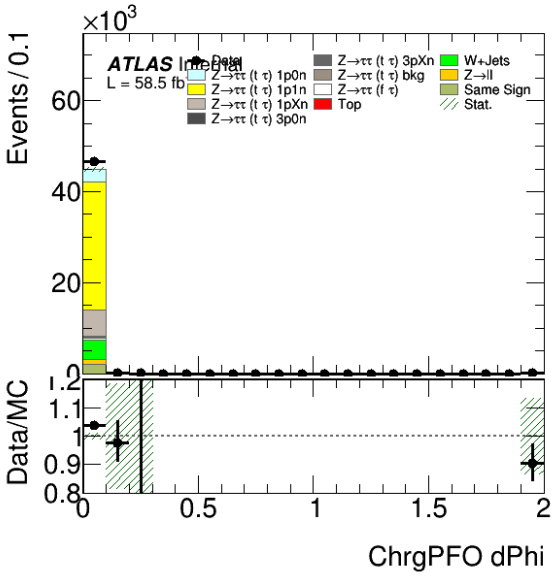
Figure 9.6: Charged PFO: dEta



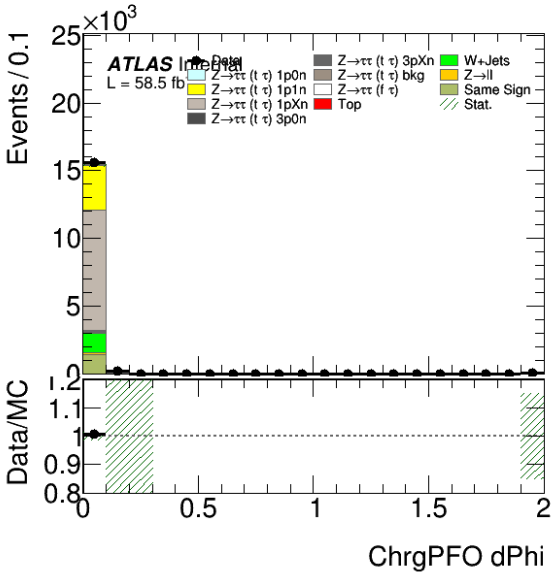
(a) no cut on RNN score



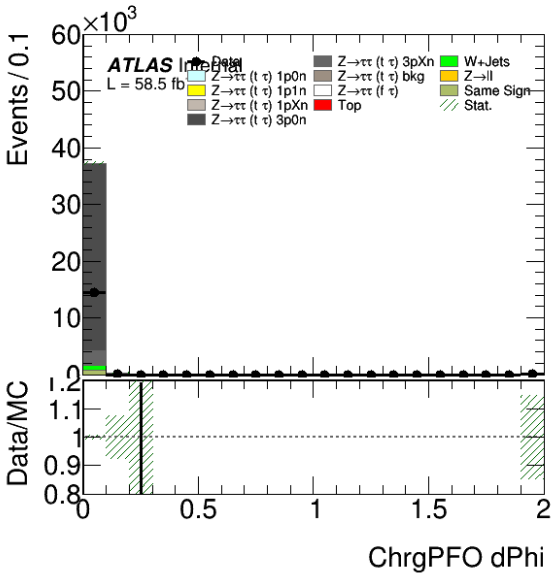
(b) RNN score == 0



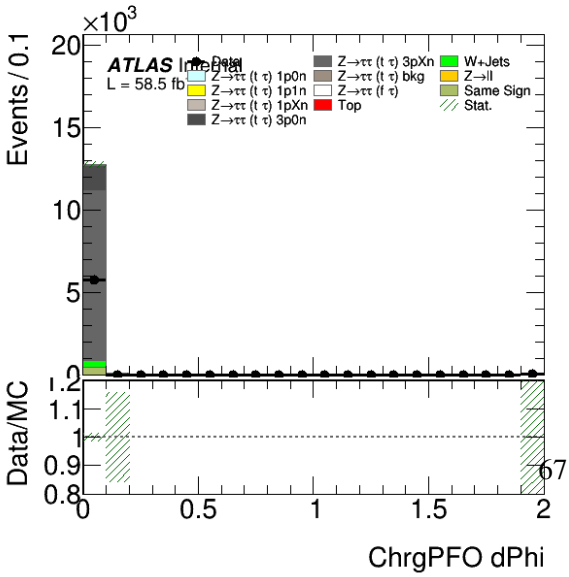
(c) RNN score == 1



(d) RNN score == 2

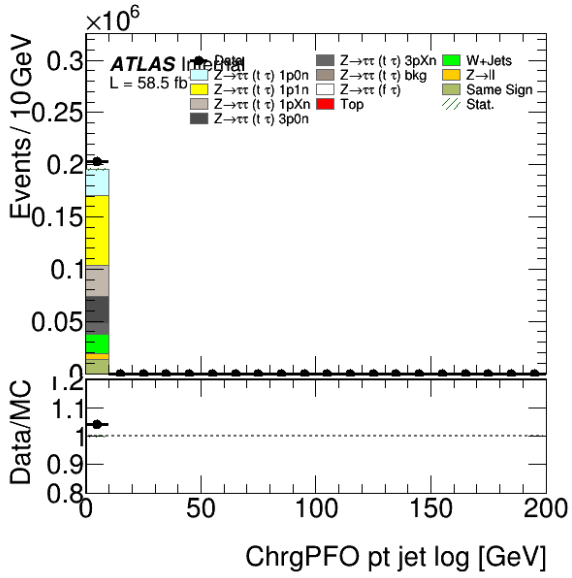


(e) RNN score == 3

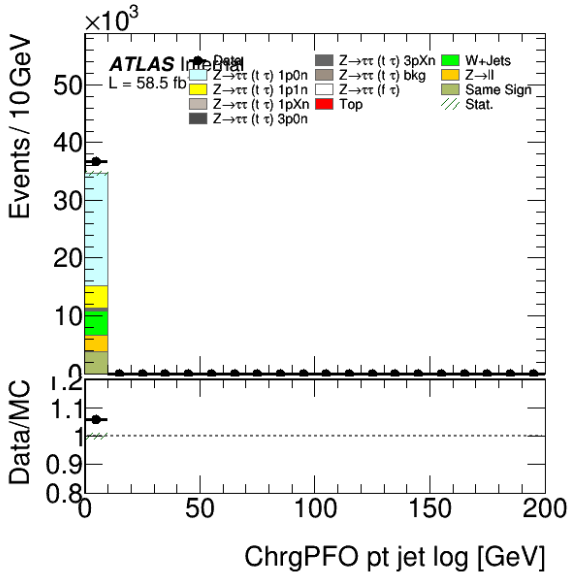


(f) RNN score == 4

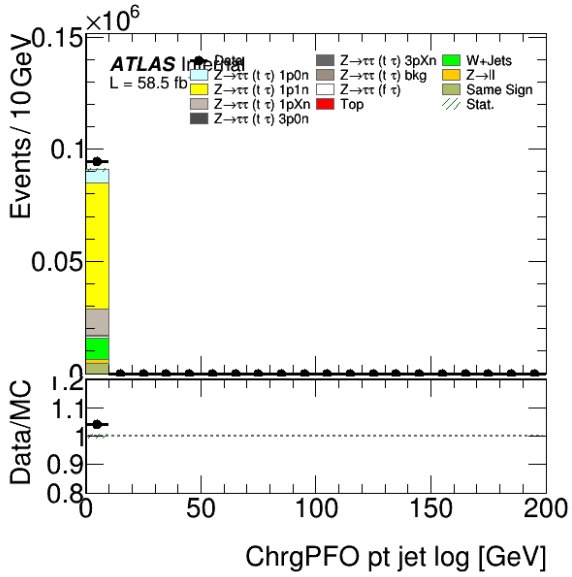
Figure 9.7: Charged PFO: dPhi



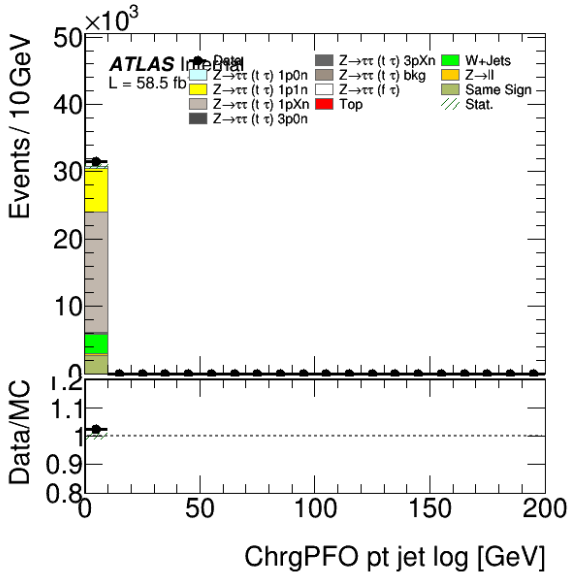
(a) no cut on RNN score



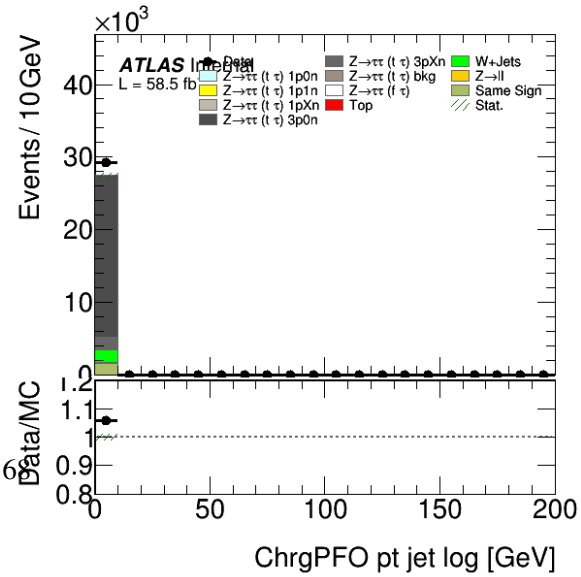
(b) RNN score == 0



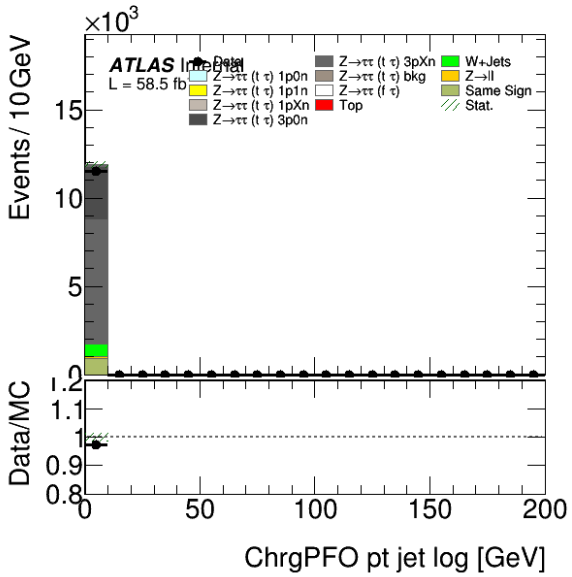
(c) RNN score == 1



(d) RNN score == 2

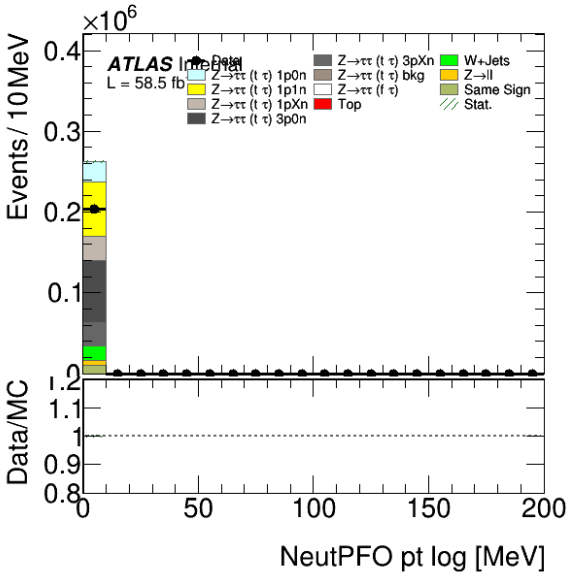


(e) RNN score == 3

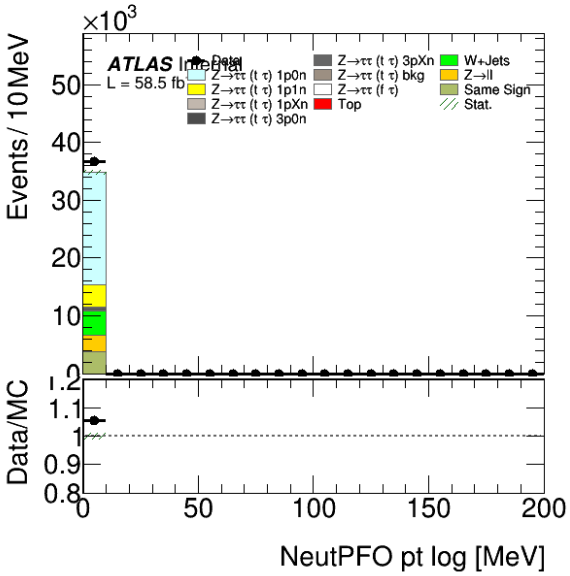


(f) RNN score == 4

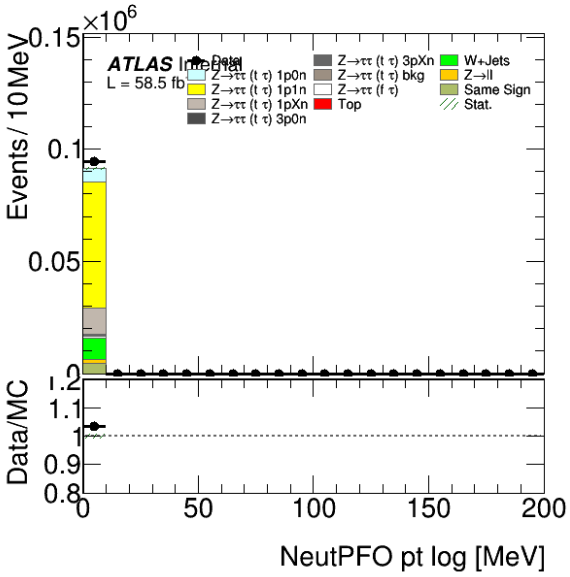
Figure 9.8: Charged PFO: ptjetlog



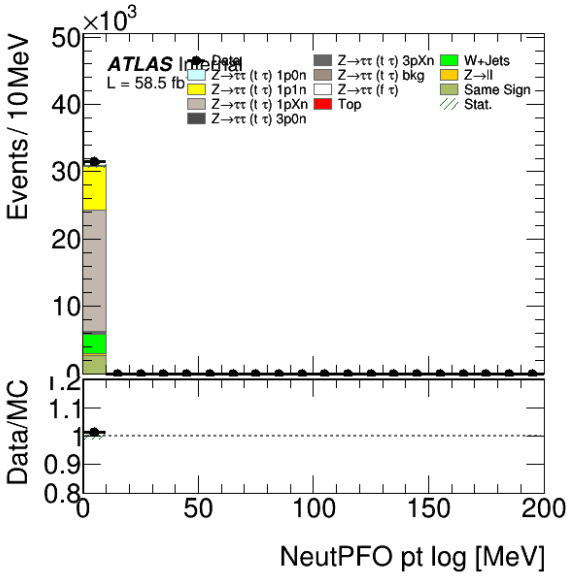
(a) no cut on RNN score



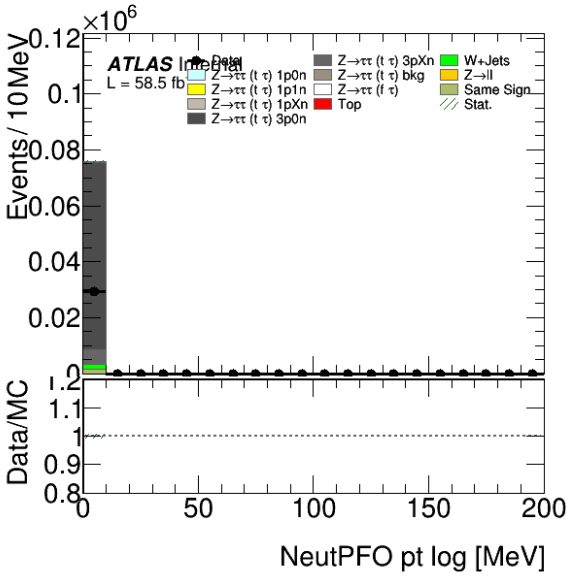
(b) RNN score == 0



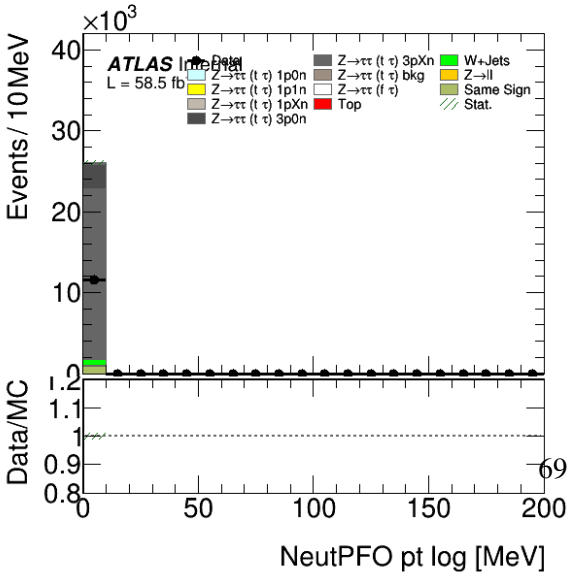
(c) RNN score == 1



(d) RNN score == 2

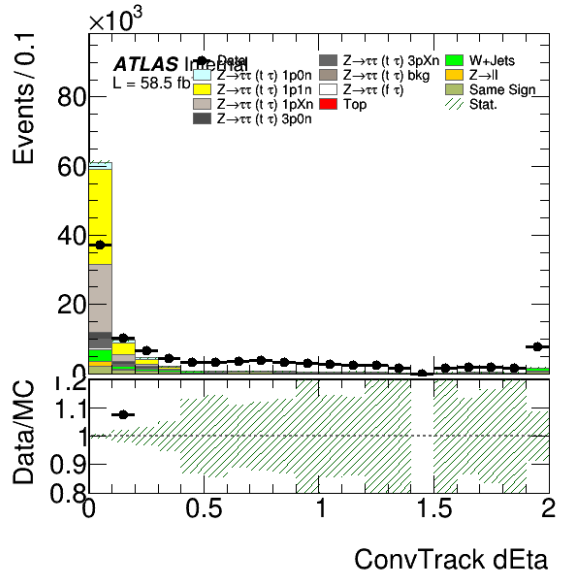


(e) RNN score == 3

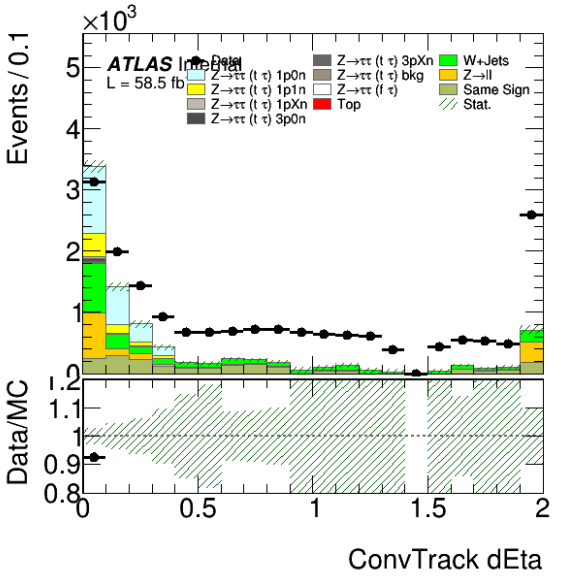


(f) RNN score == 4

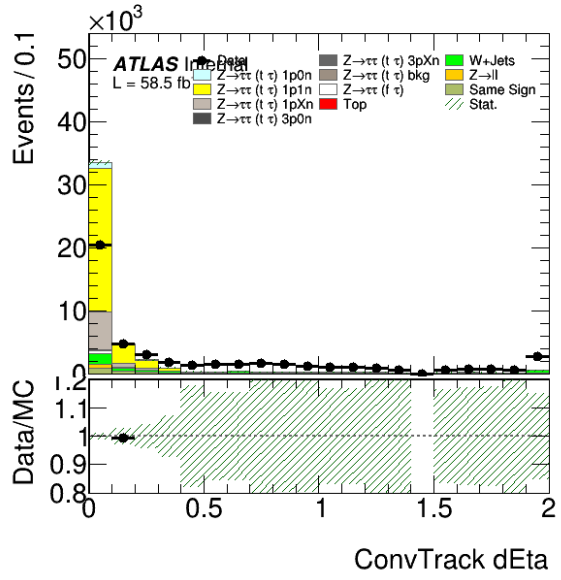
Figure 9.9: Charged PFO: ptlog



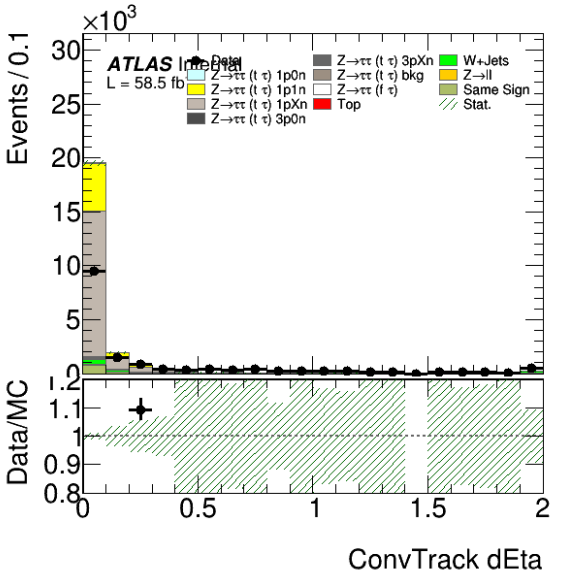
(a) no cut on RNN score



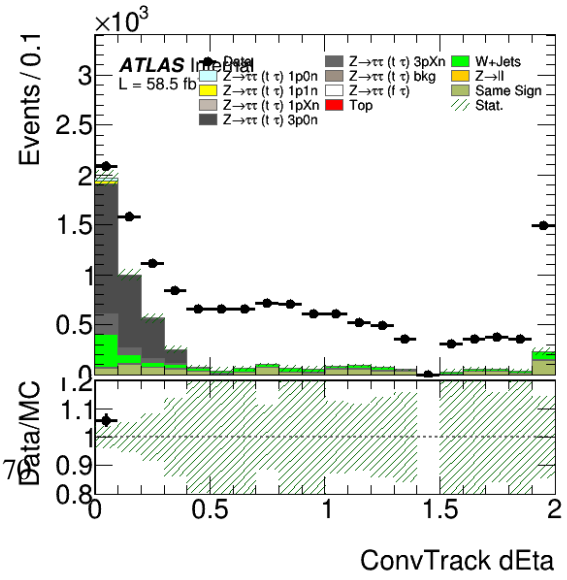
(b) RNN score == 0



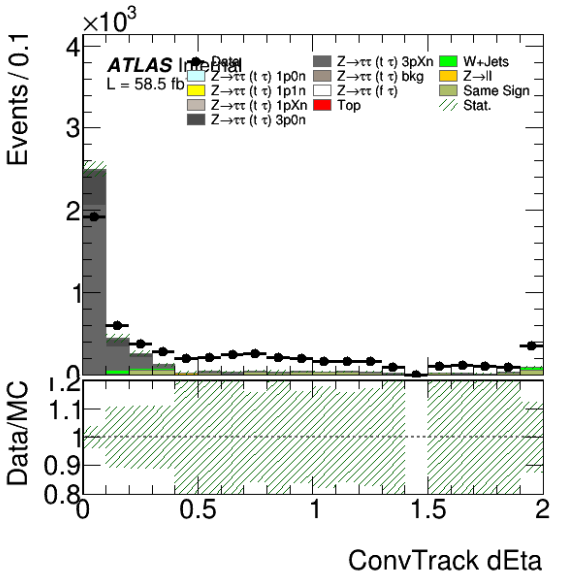
(c) RNN score == 1



(d) RNN score == 2

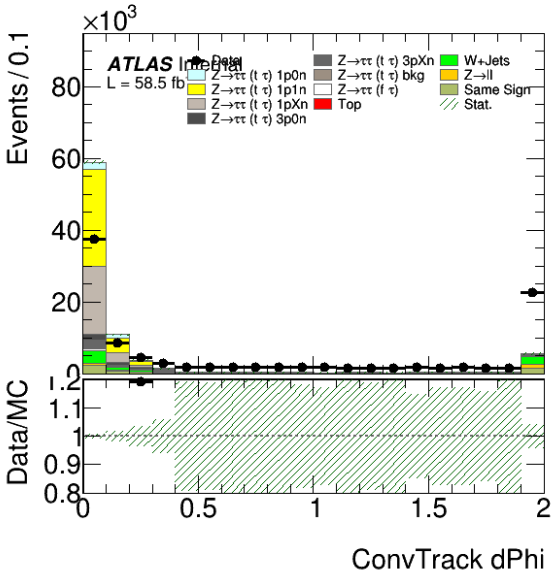


(e) RNN score == 3

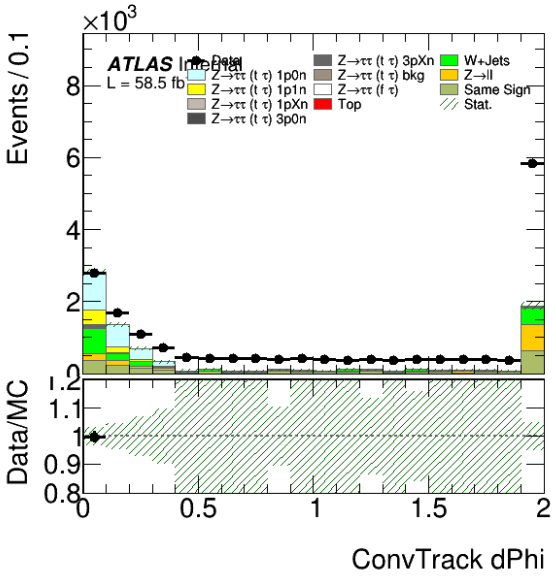


(f) RNN score == 4

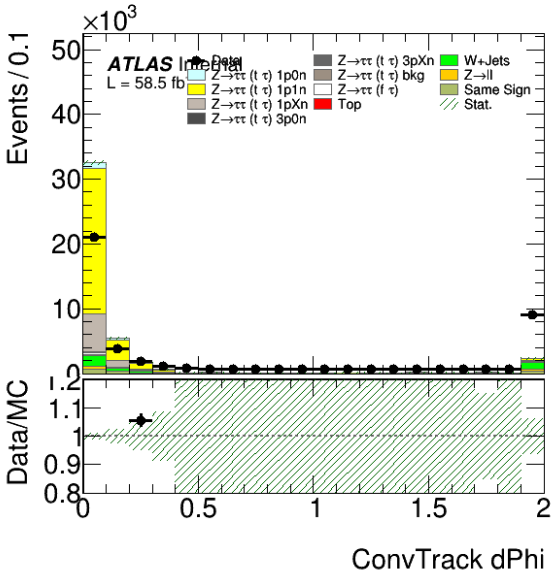
Figure 9.10: Conversion tracks: dEta



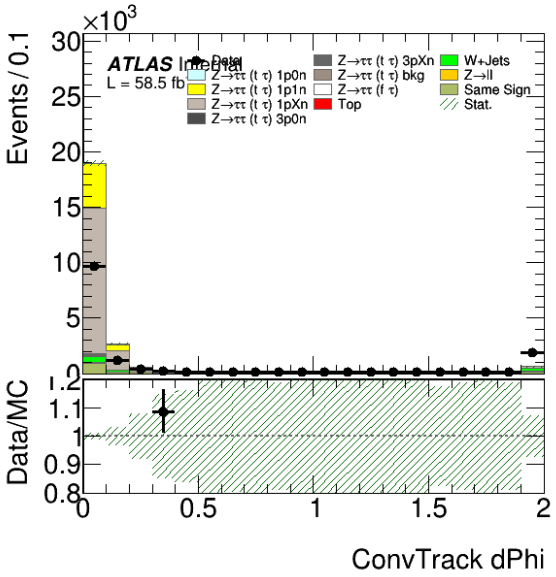
(a) no cut on RNN score



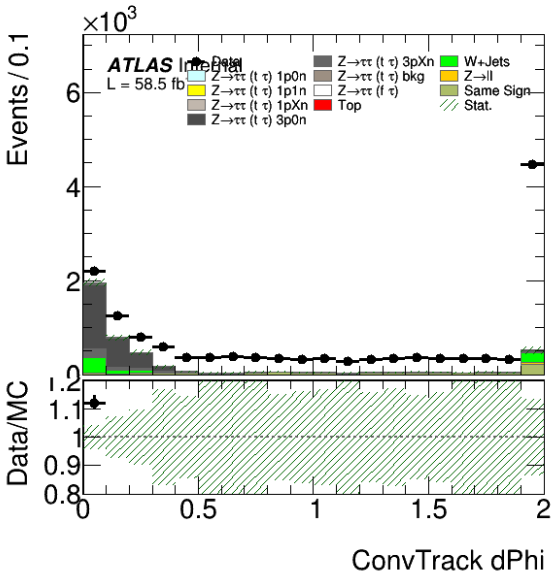
(b) RNN score == 0



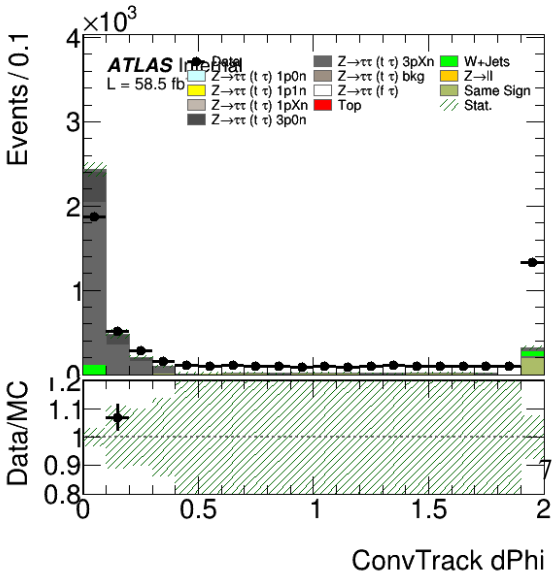
(c) RNN score == 1



(d) RNN score == 2

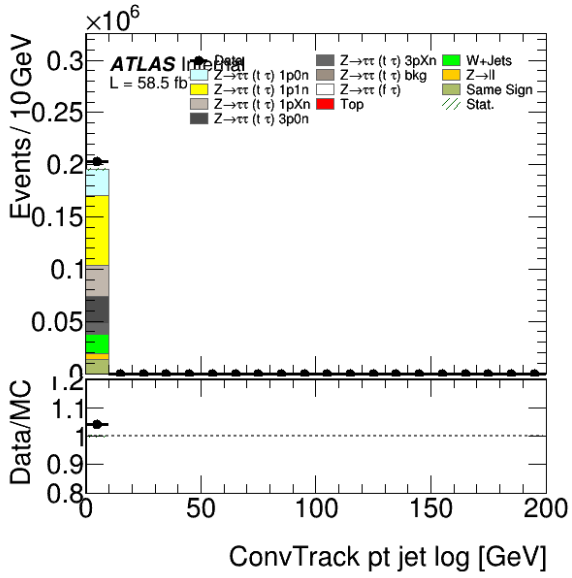


(e) RNN score == 3

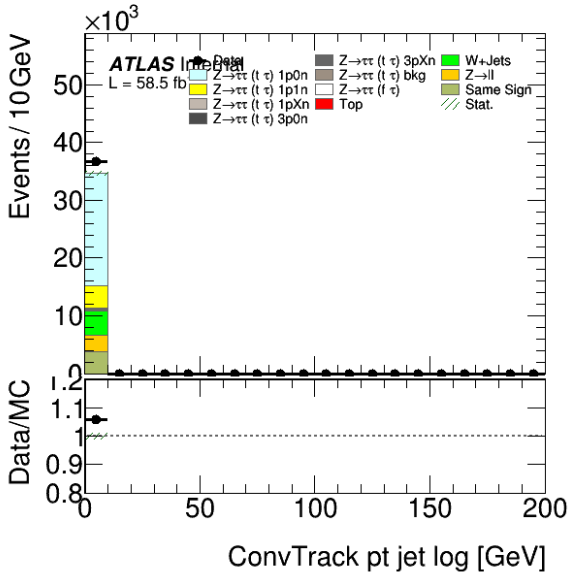


(f) RNN score == 4

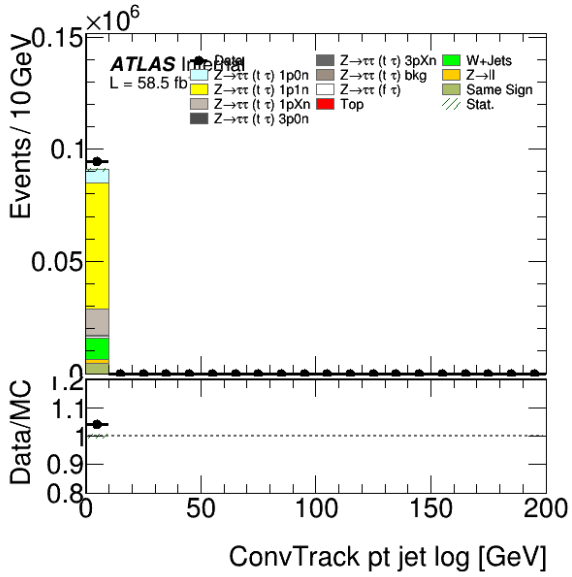
Figure 9.11: Conversion tracks: dPhi



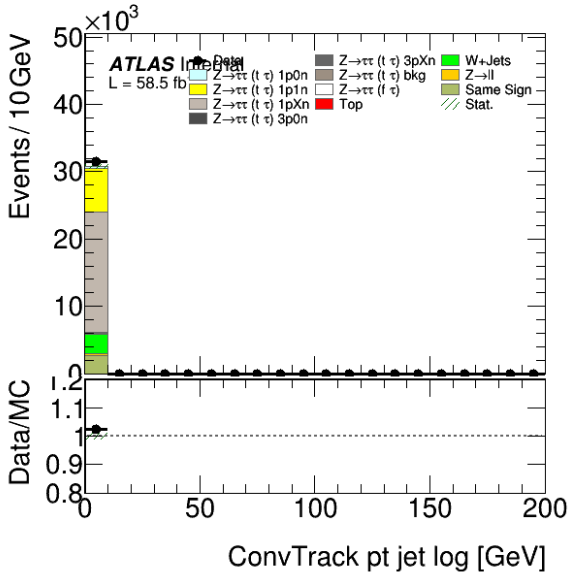
(a) no cut on RNN score



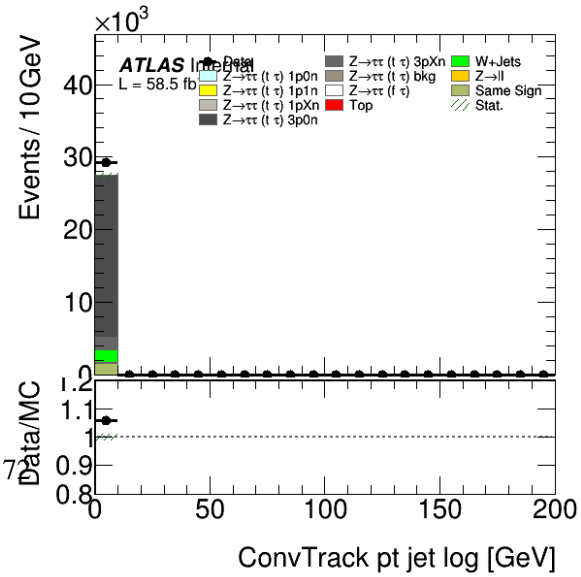
(b) RNN score == 0



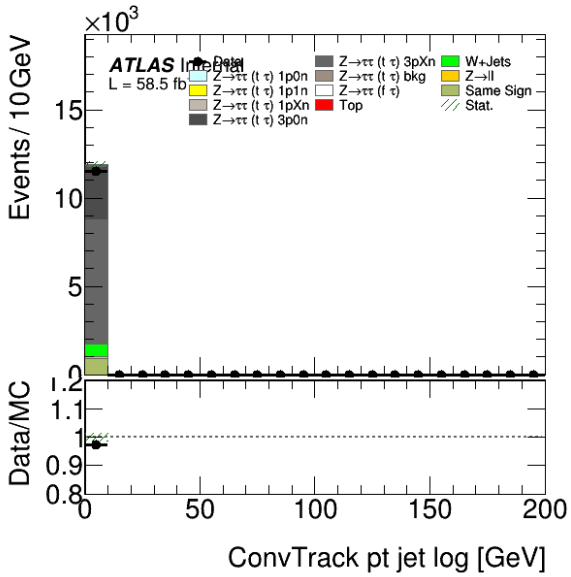
(c) RNN score == 1



(d) RNN score == 2

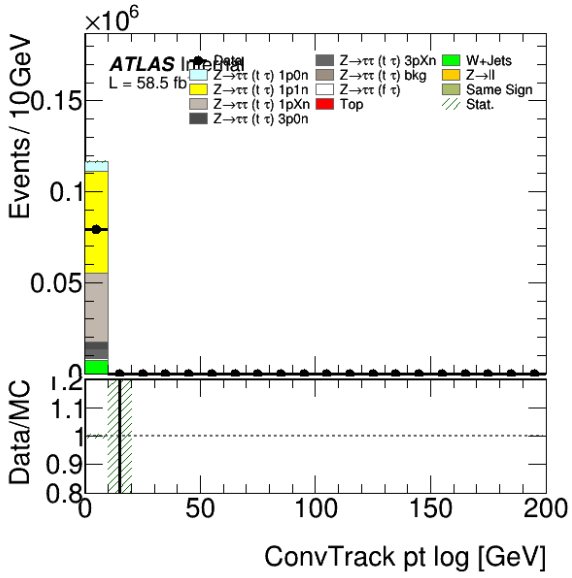


(e) RNN score == 3

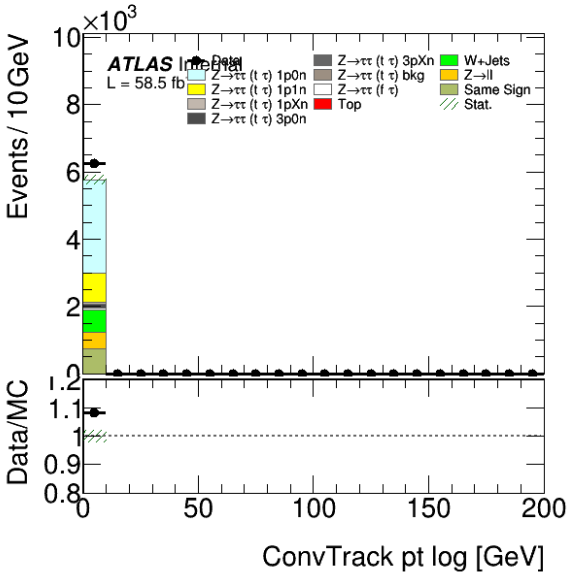


(f) RNN score == 4

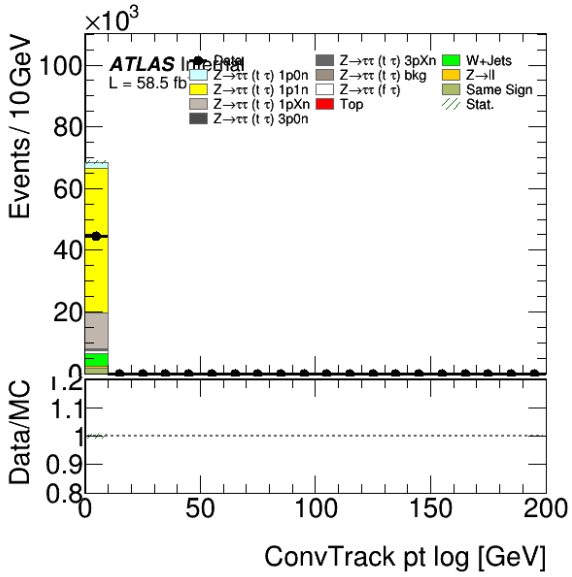
Figure 9.12: Conversion tracks: ptjetlog



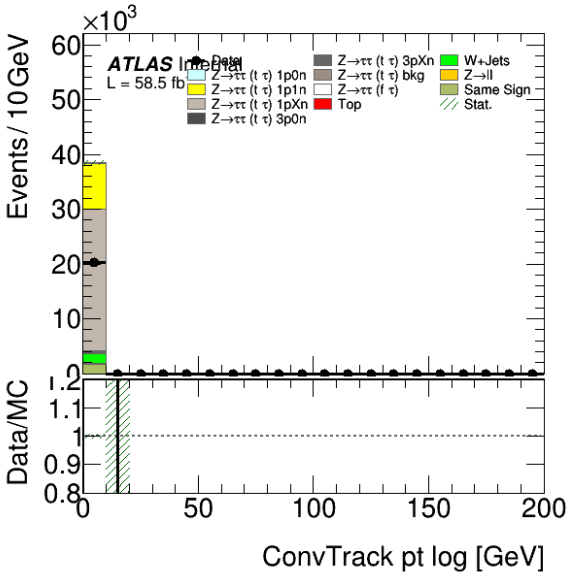
(a) no cut on RNN score



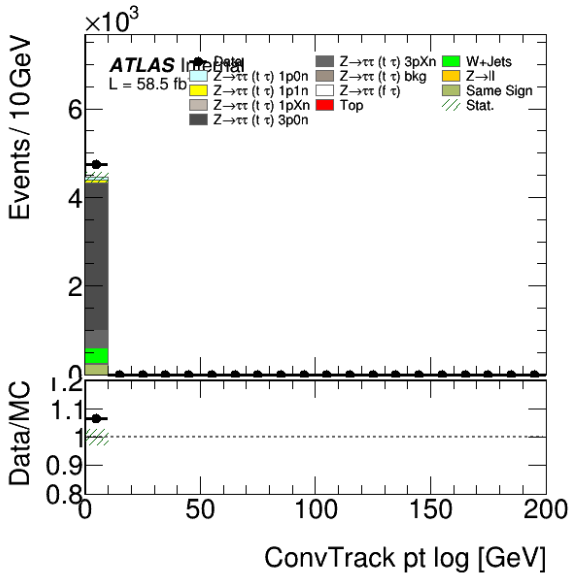
(b) RNN score == 0



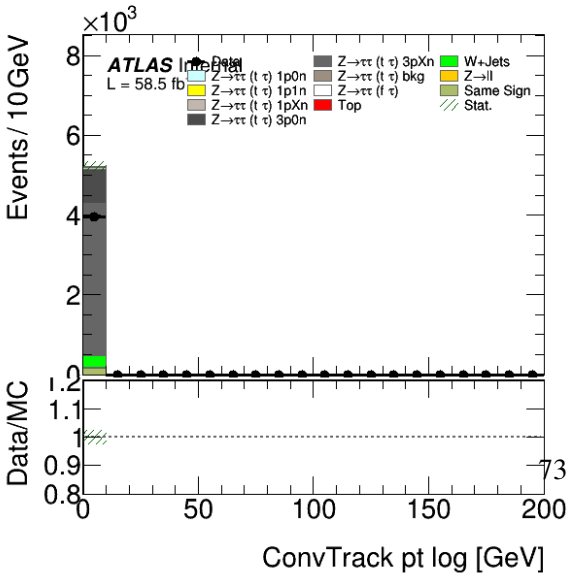
(c) RNN score == 1



(d) RNN score == 2

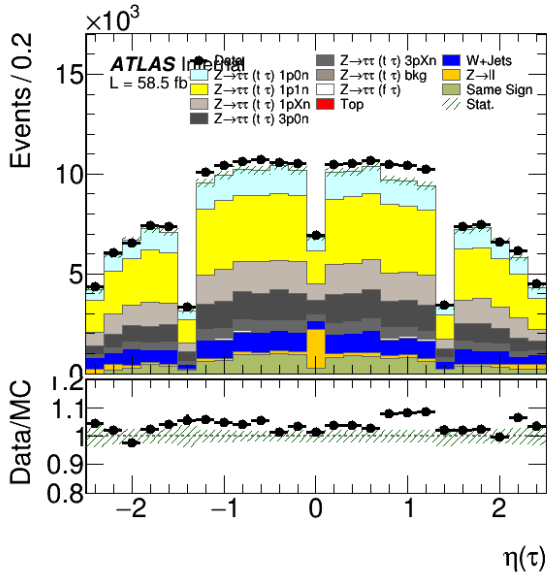


(e) RNN score == 3

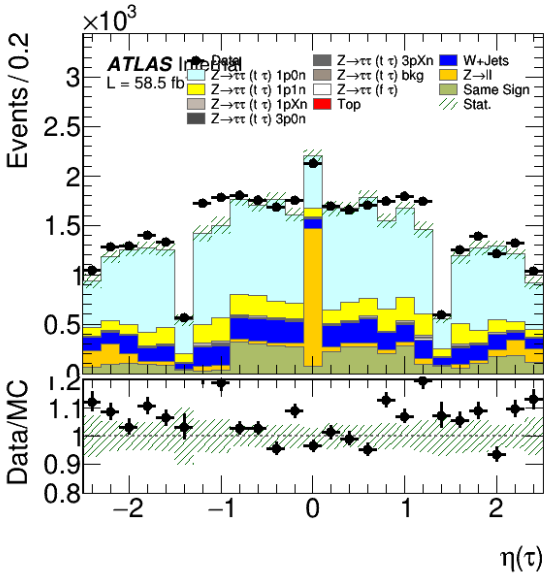


(f) RNN score == 4

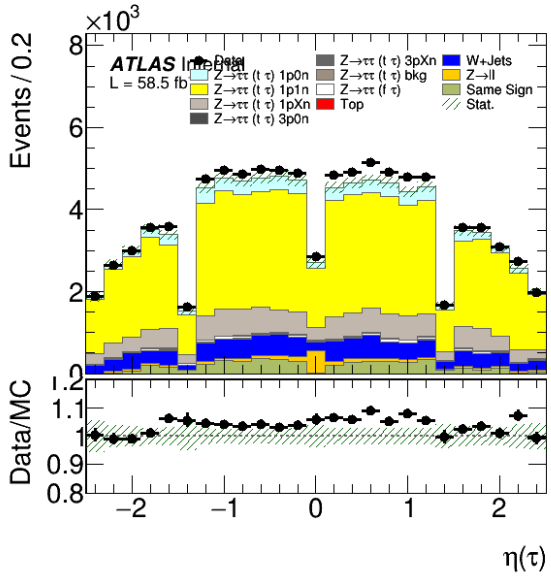
Figure 9.13: Conversion tracks: ptlog



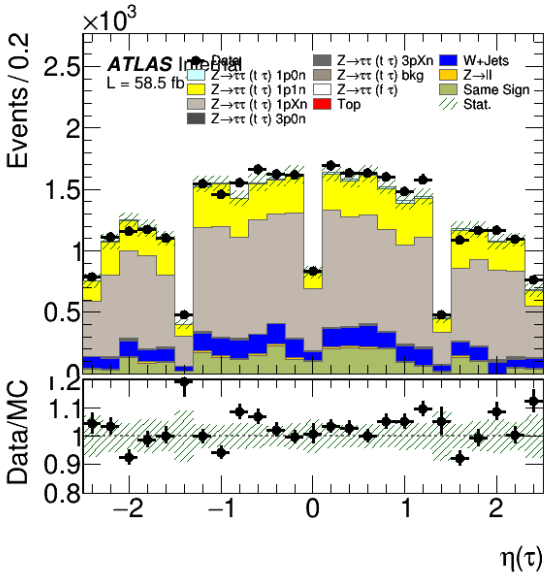
(a) no cut on RNN score



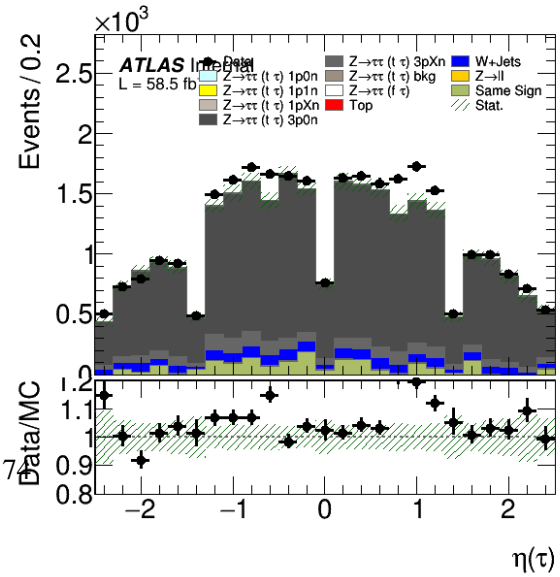
(b) RNN score == 0



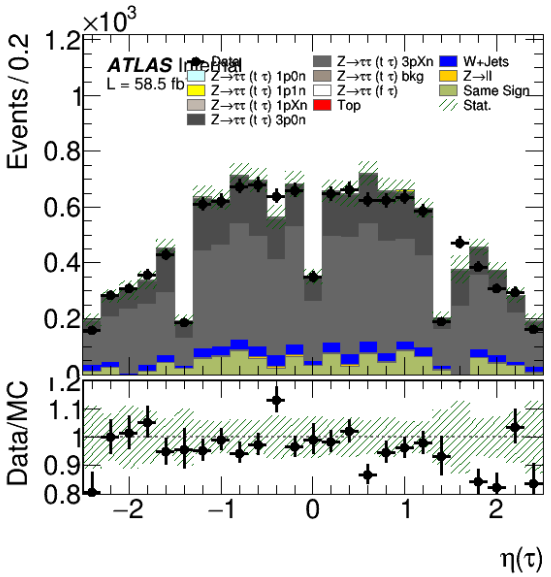
(c) RNN score == 1



(d) RNN score == 2

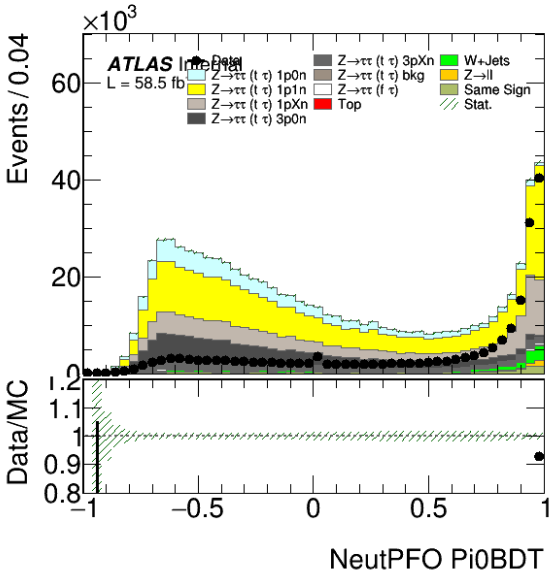


(e) RNN score == 3

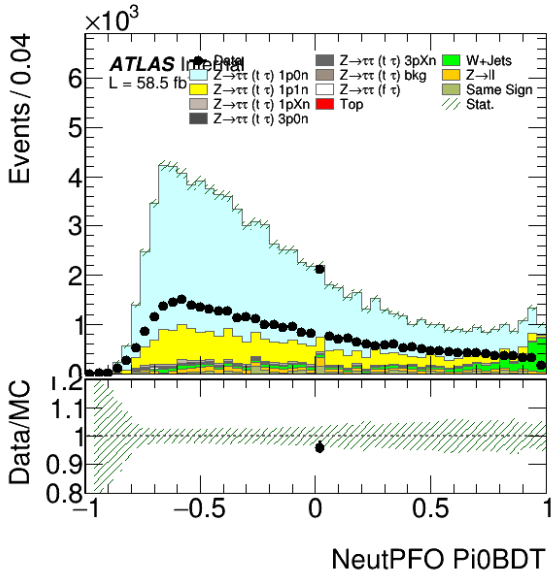


(f) RNN score == 4

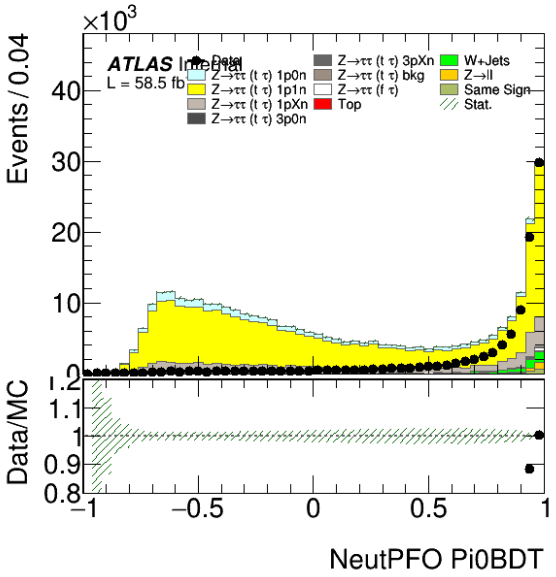
Figure 9.14: Tau_Eta



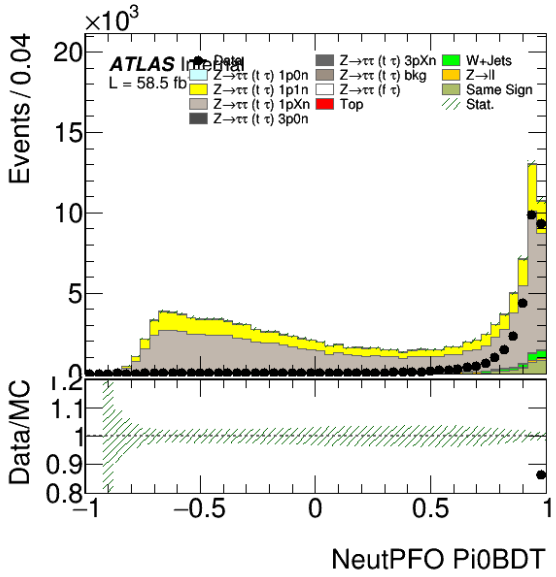
(a) no cut on RNN score



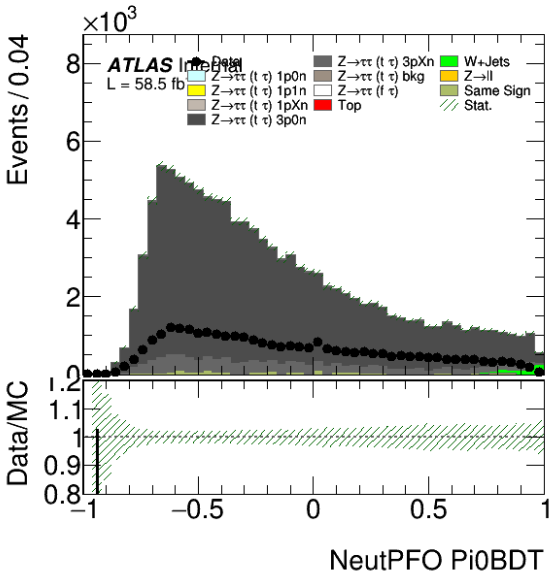
(b) RNN score == 0



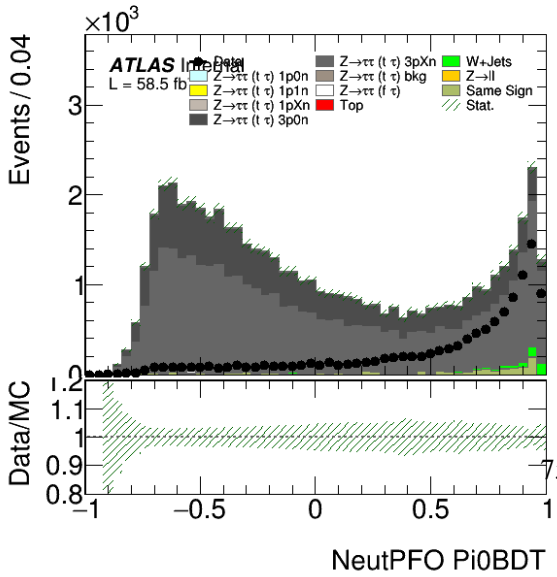
(c) RNN score == 1



(d) RNN score == 2

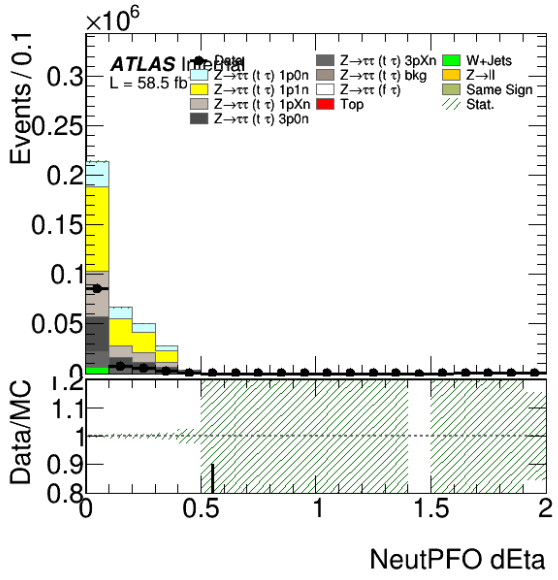


(e) RNN score == 3

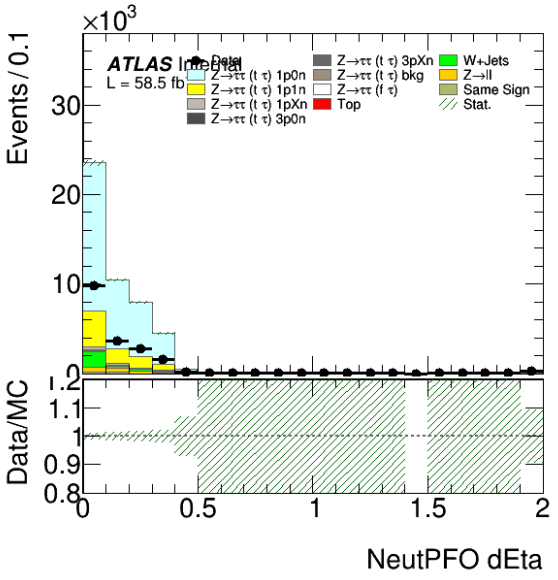


(f) RNN score == 4

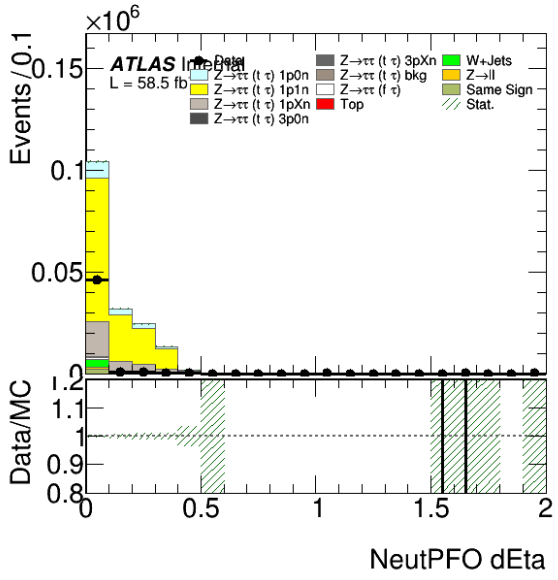
Figure 9.15: Neutral PFO: BDT-π^0 score



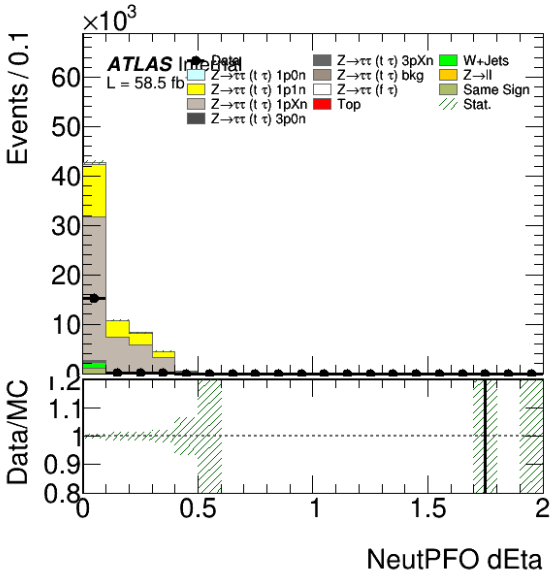
(a) no cut on RNN score



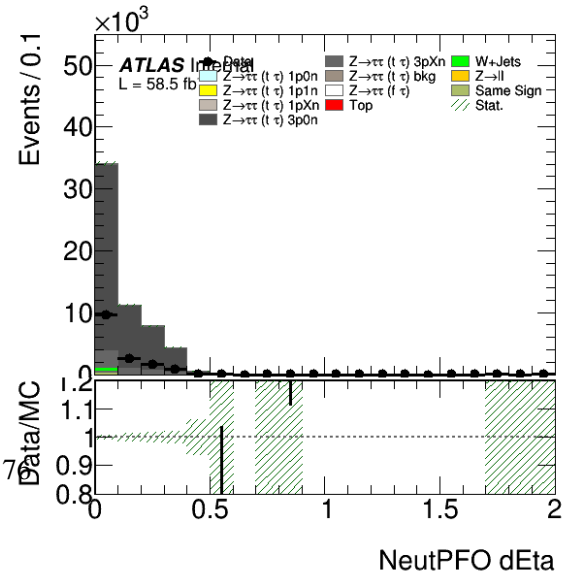
(b) RNN score == 0



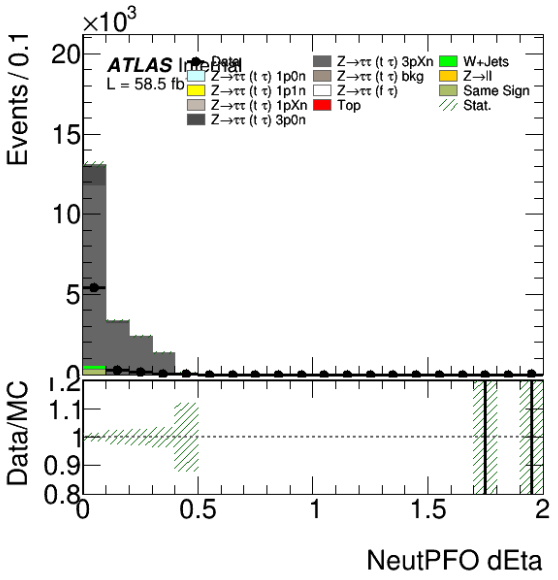
(c) RNN score == 1



(d) RNN score == 2

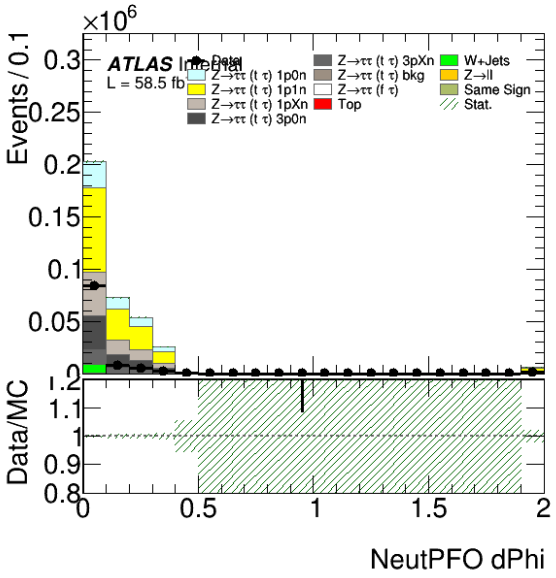


(e) RNN score == 3

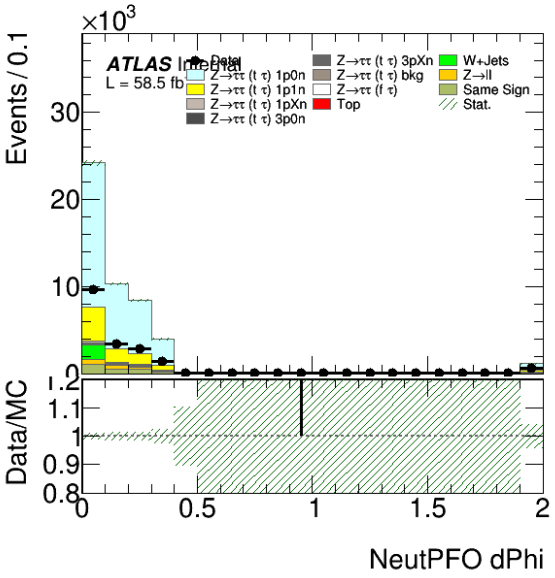


(f) RNN score == 4

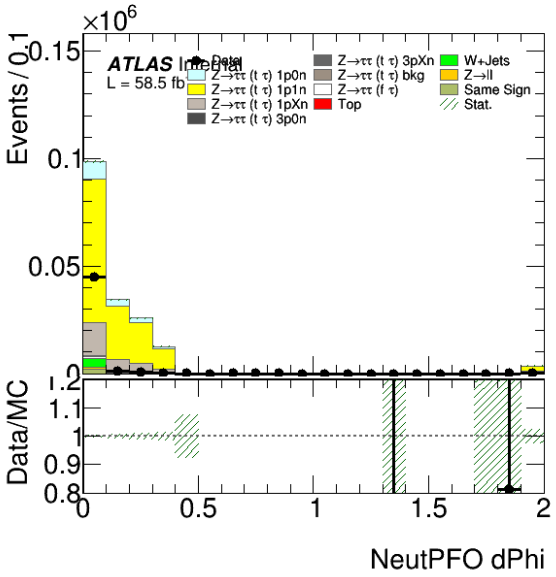
Figure 9.16: Neutral PFO: dEta



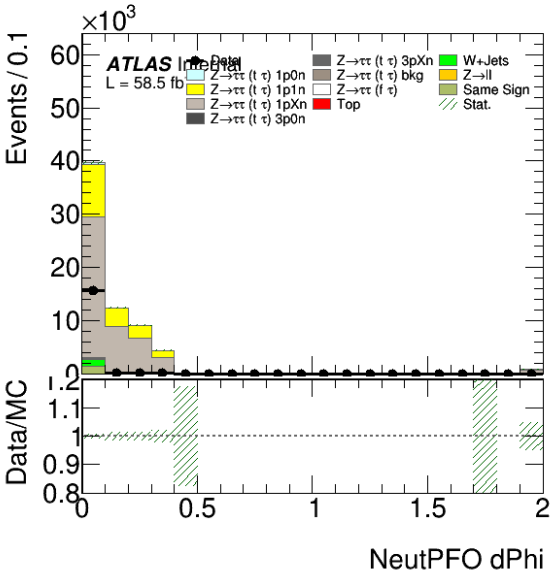
(a) no cut on RNN score



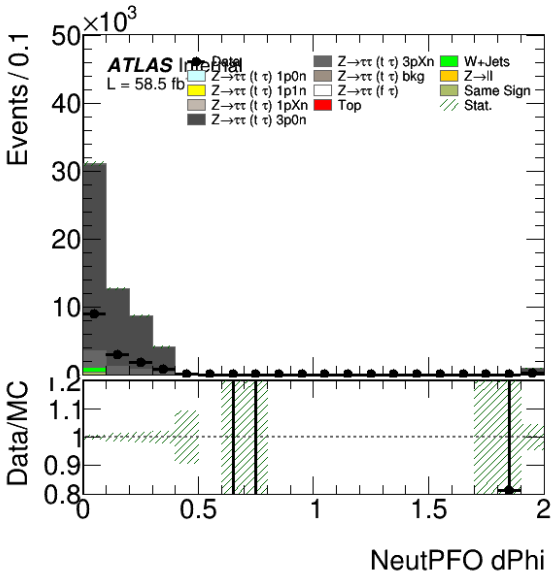
(b) RNN score == 0



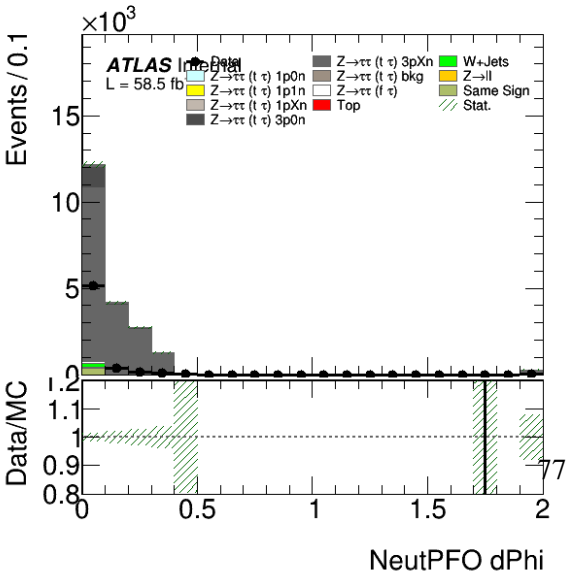
(c) RNN score == 1



(d) RNN score == 2

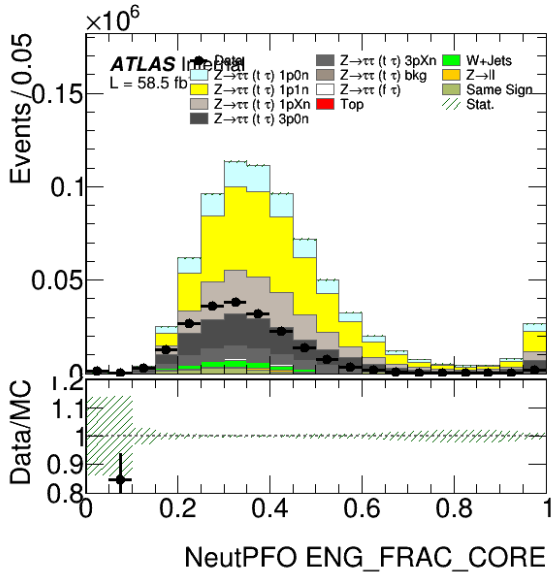


(e) RNN score == 3

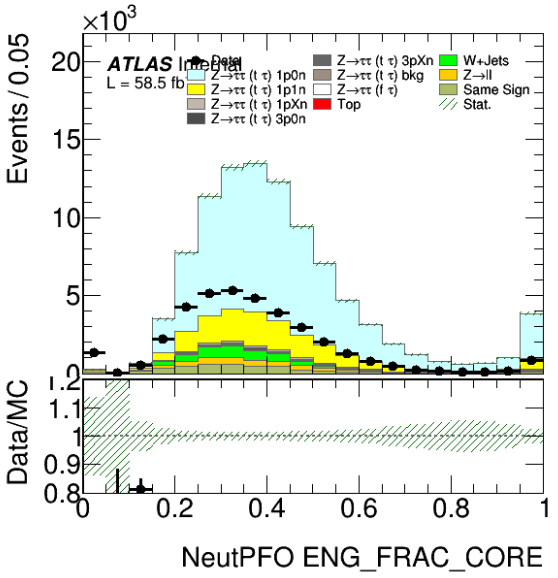


(f) RNN score == 4

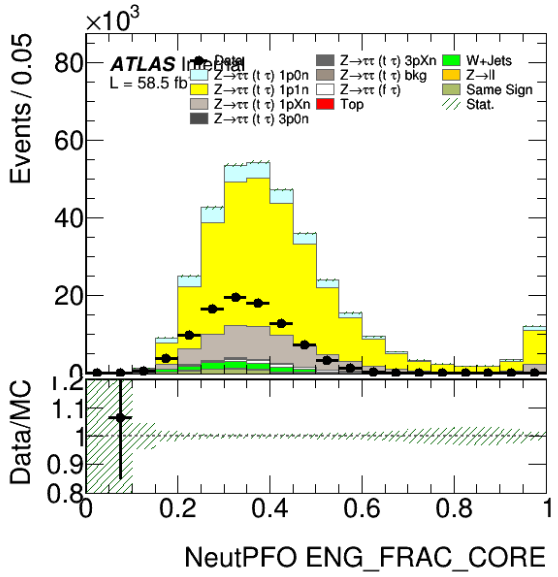
Figure 9.17: Neutral PFO: dPhi



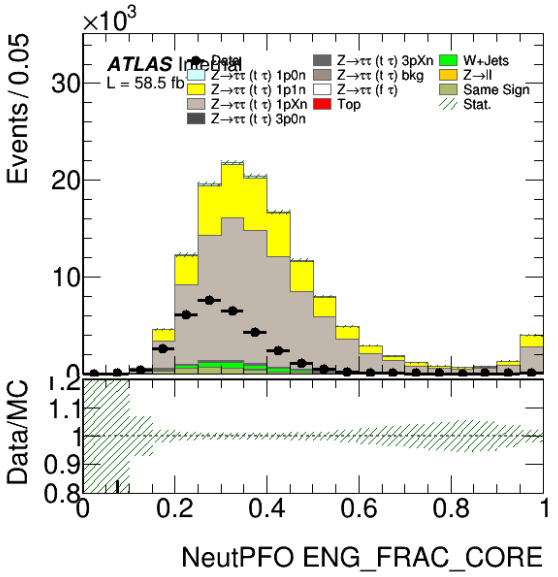
(a) no cut on RNN score



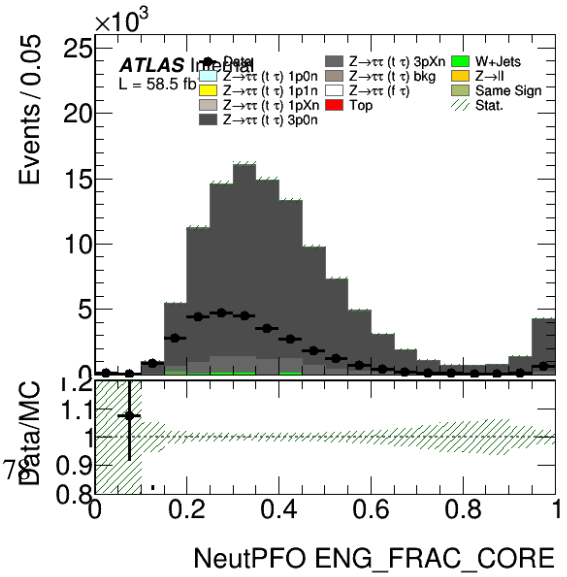
(b) RNN score == 0



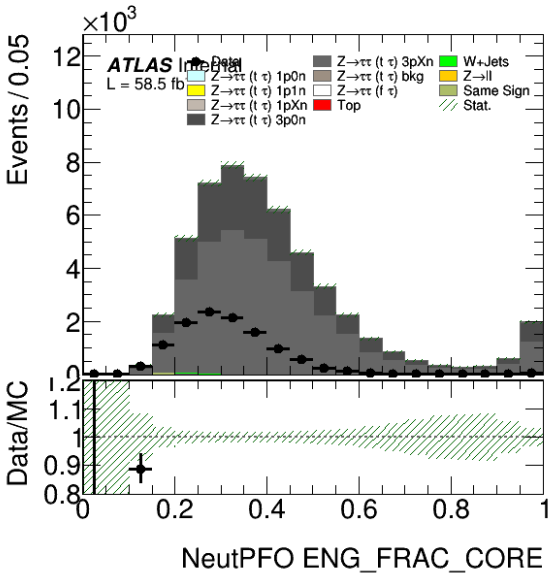
(c) RNN score == 1



(d) RNN score == 2

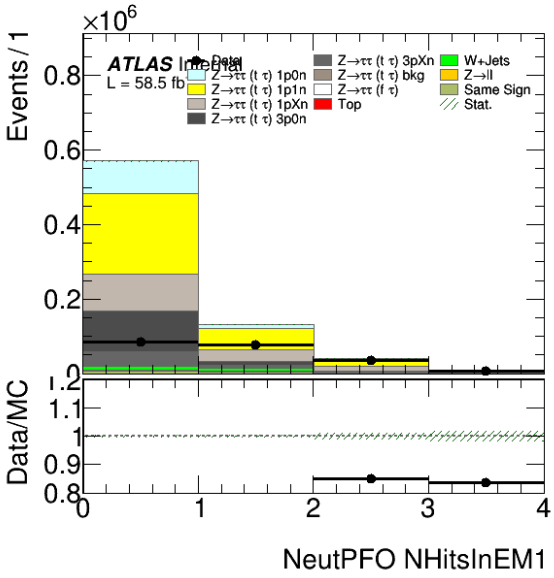


(e) RNN score == 3

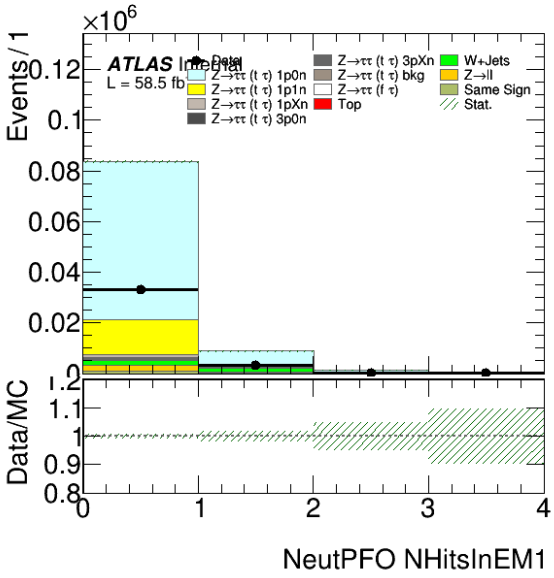


(f) RNN score == 4

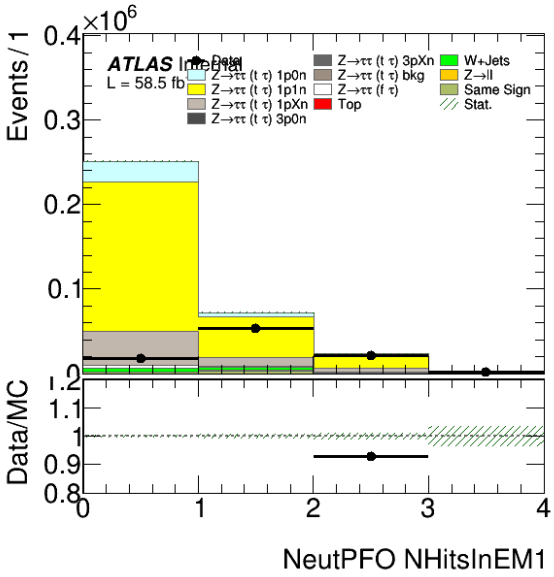
Figure 9.18: Neutral PFO: ENG_FRAC_CORE



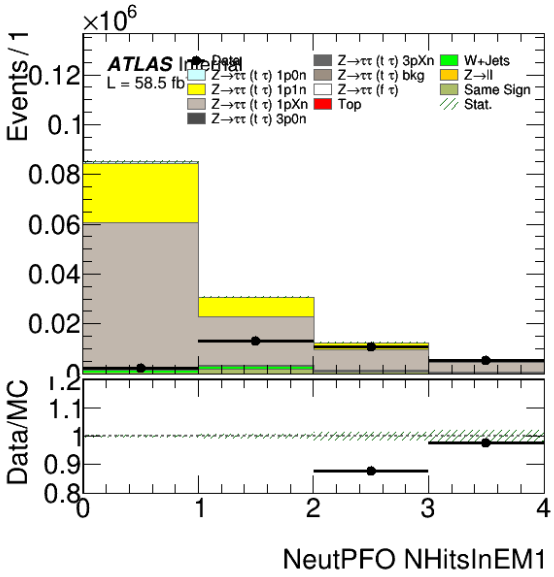
(a) no cut on RNN score



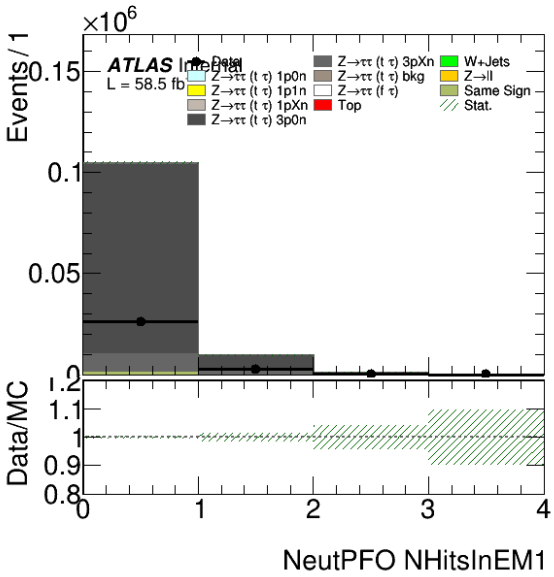
(b) RNN score == 0



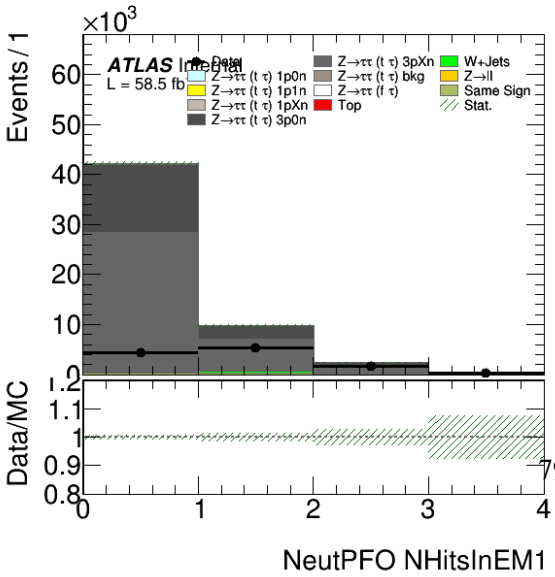
(c) RNN score == 1



(d) RNN score == 2

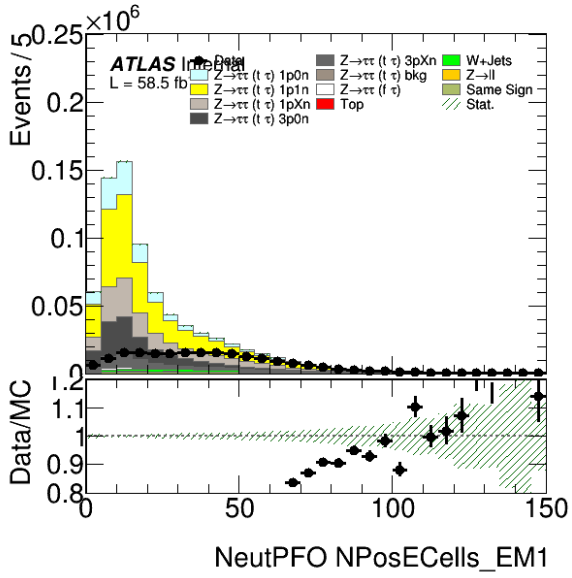


(e) RNN score == 3

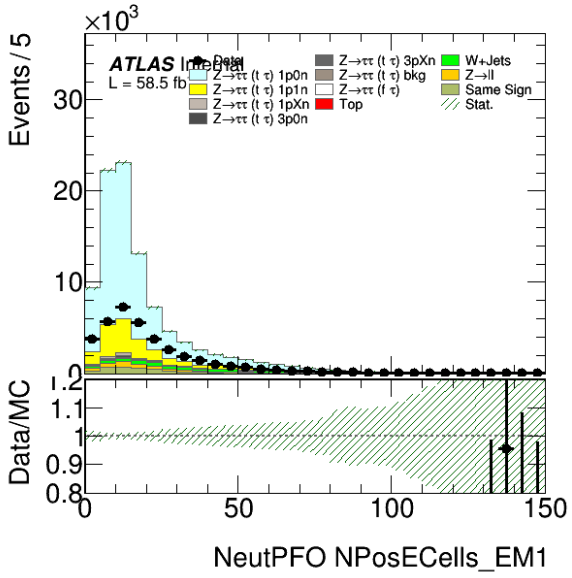


(f) RNN score == 4

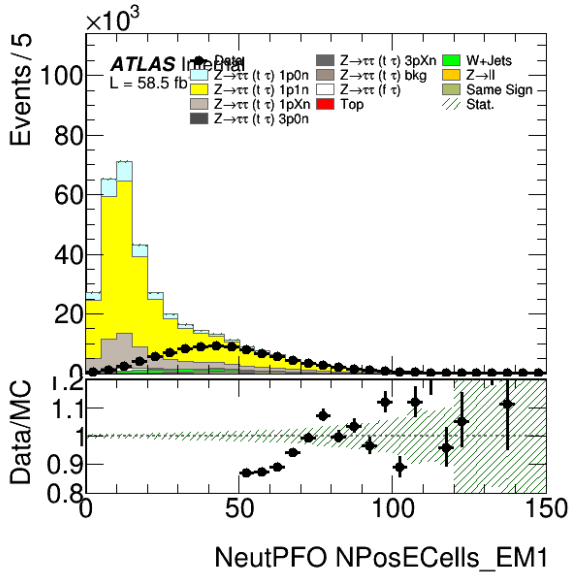
Figure 9.19: Neutral PFO: nHitsInEM1



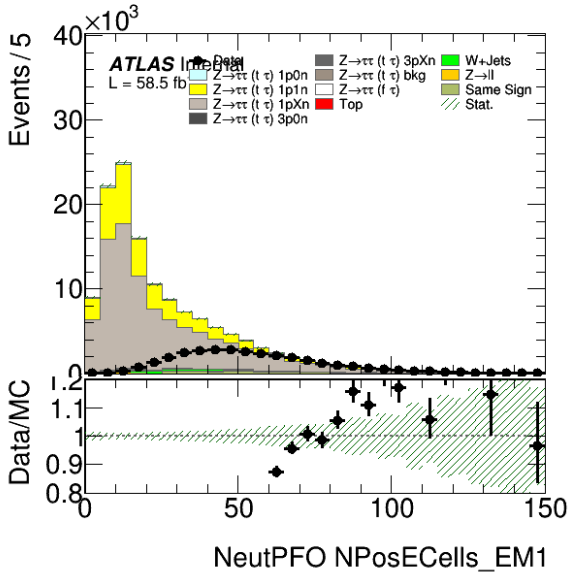
(a) no cut on RNN score



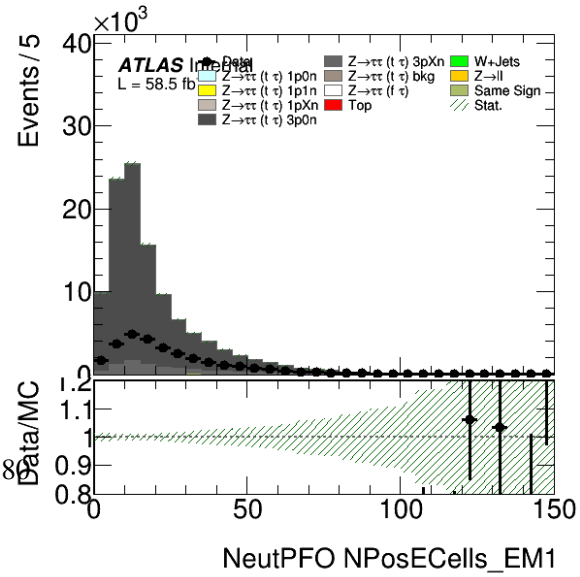
(b) RNN score == 0



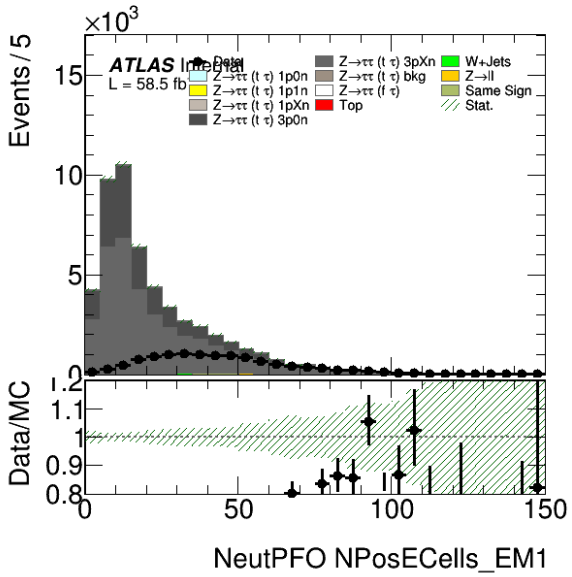
(c) RNN score == 1



(d) RNN score == 2

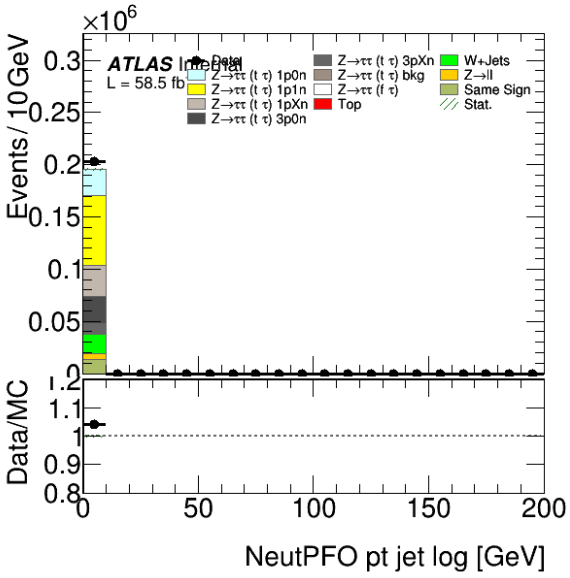


(e) RNN score == 3

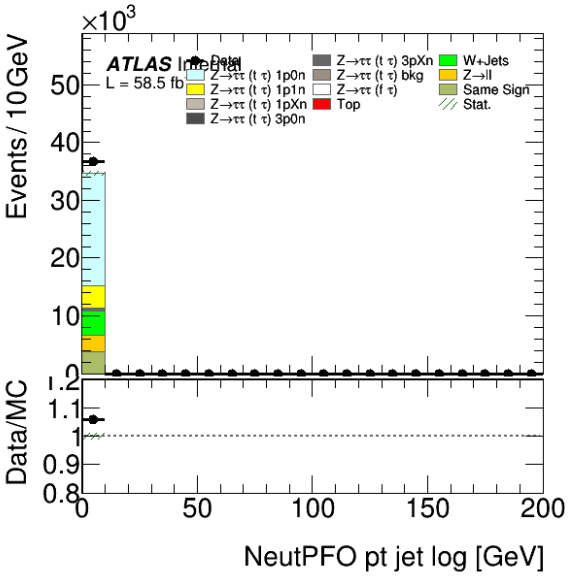


(f) RNN score == 4

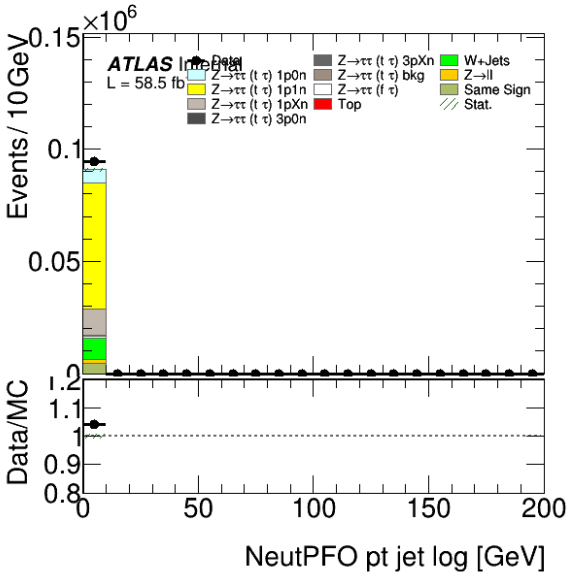
Figure 9.20: Neutral PFO: NPosECells_EM1



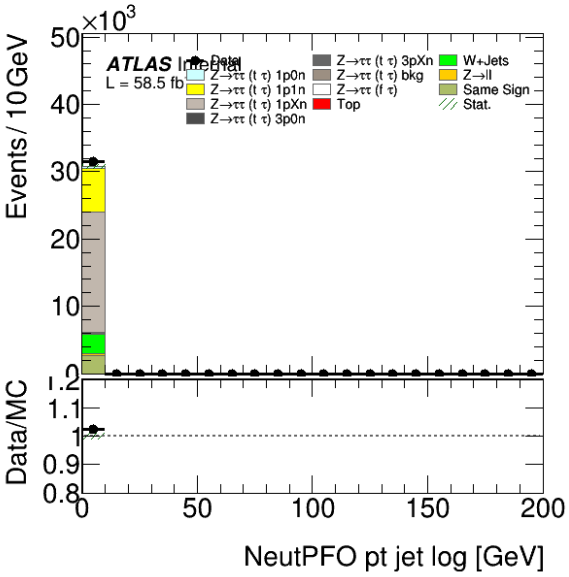
(a) no cut on RNN score



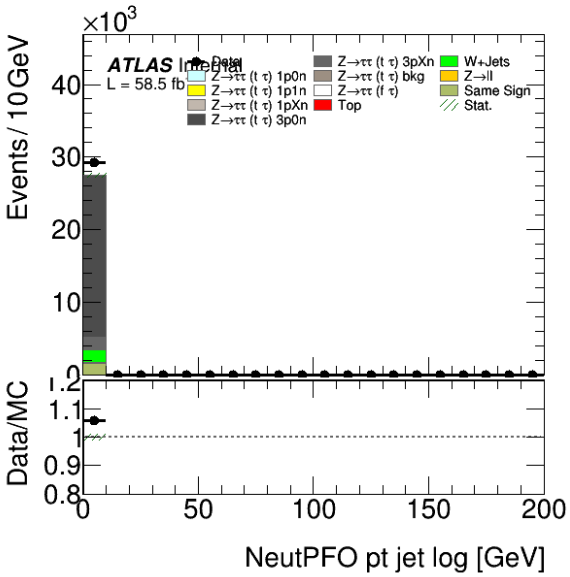
(b) RNN score == 0



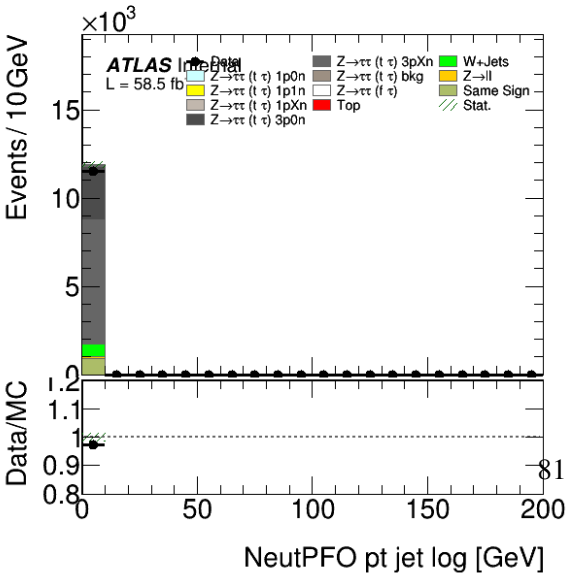
(c) RNN score == 1



(d) RNN score == 2

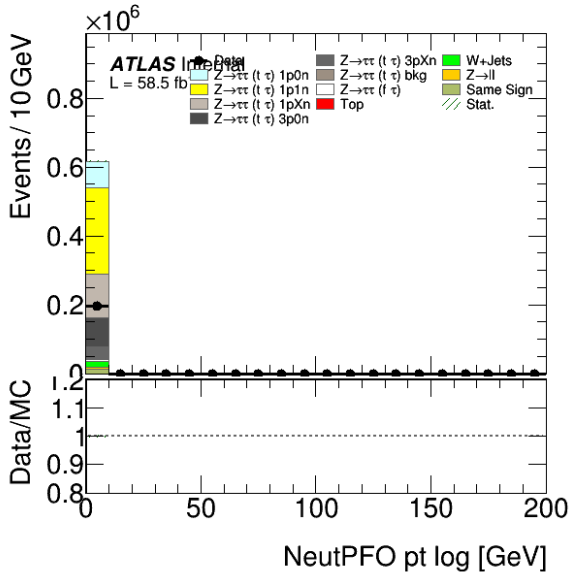


(e) RNN score == 3

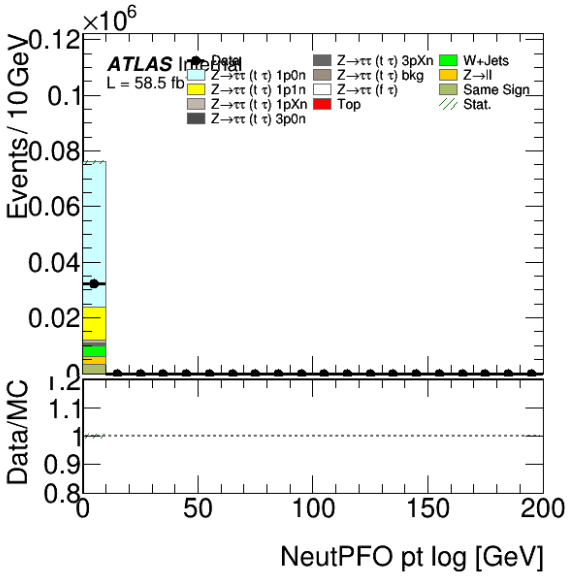


(f) RNN score == 4

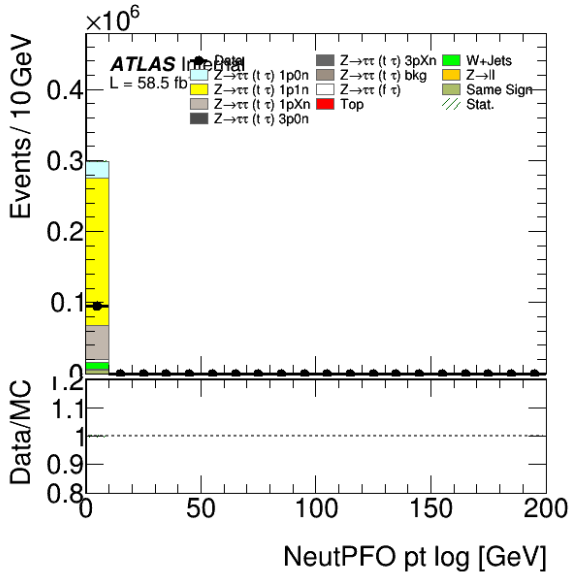
Figure 9.21: Neutral PFO: ptjetlog



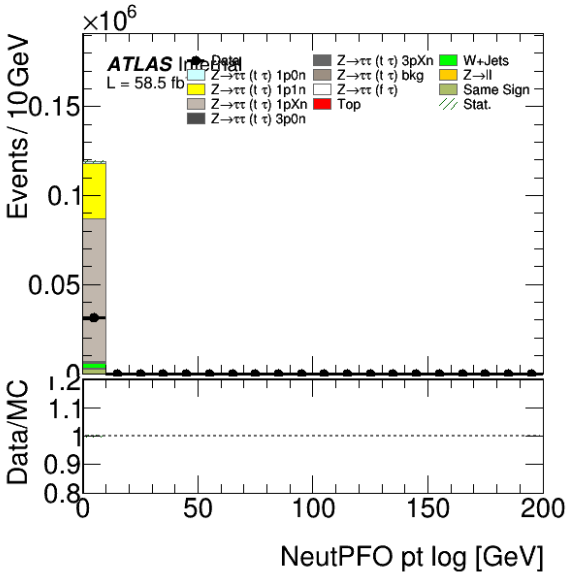
(a) no cut on RNN score



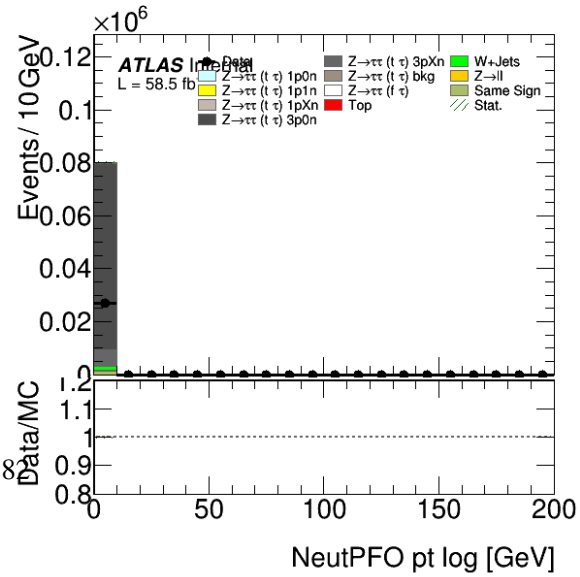
(b) RNN score == 0



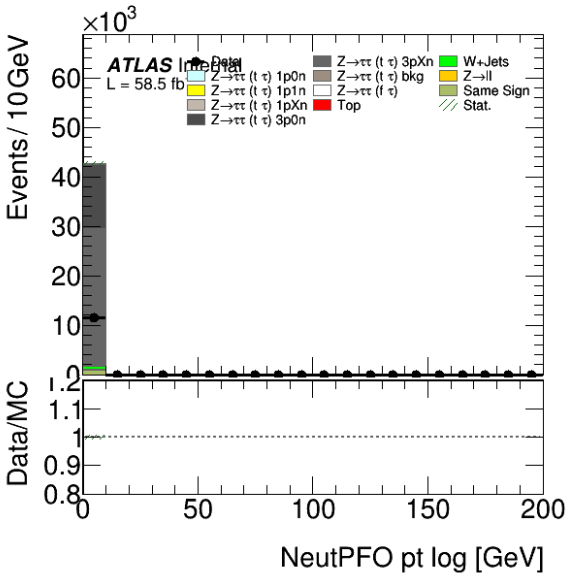
(c) RNN score == 1



(d) RNN score == 2

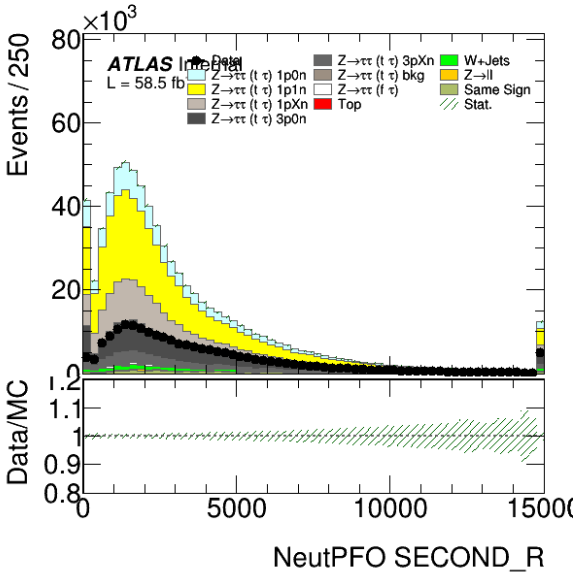


(e) RNN score == 3

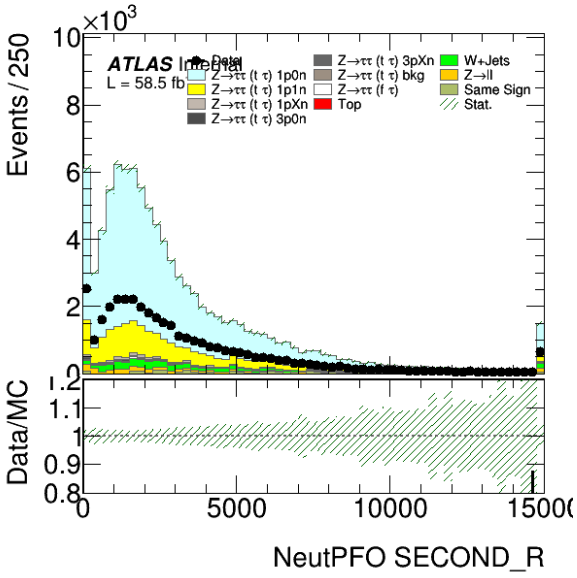


(f) RNN score == 4

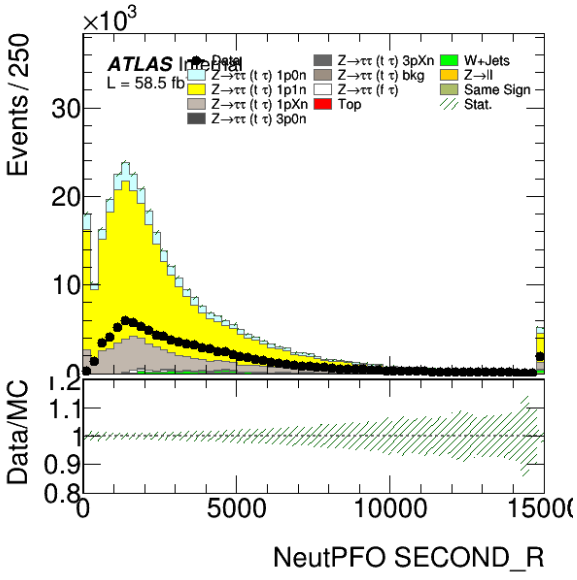
Figure 9.22: Neutral PFO: ptlog



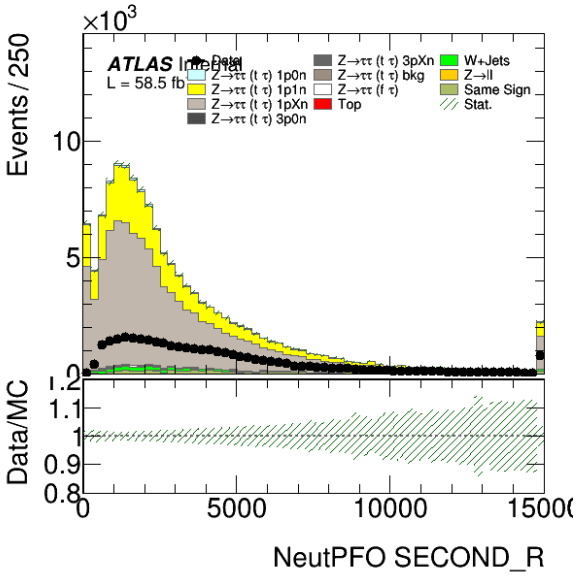
(a) no cut on RNN score



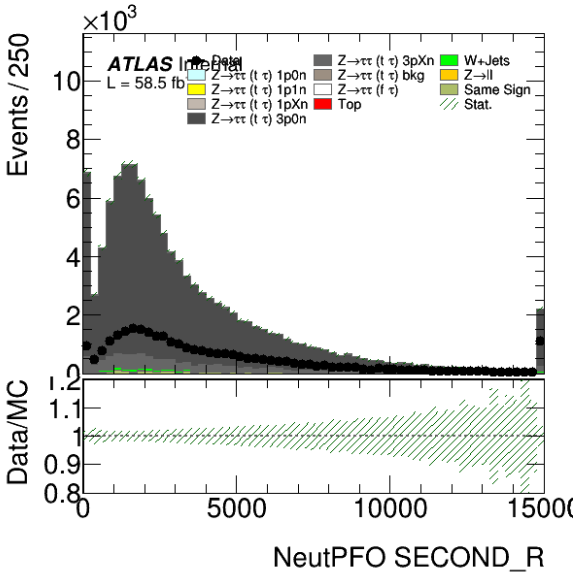
(b) RNN score == 0



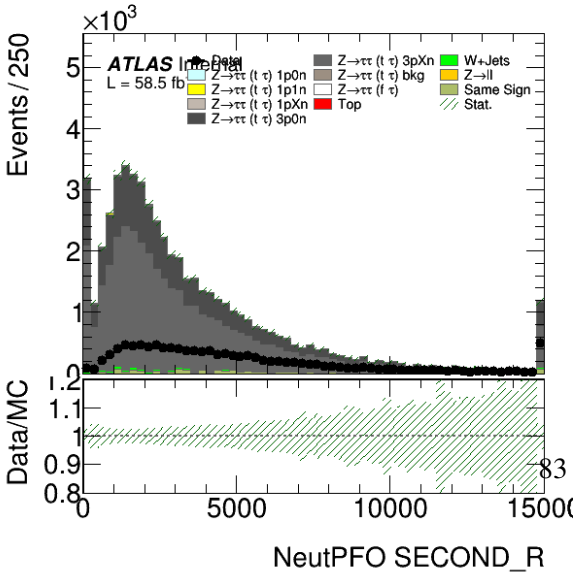
(c) RNN score == 1



(d) RNN score == 2

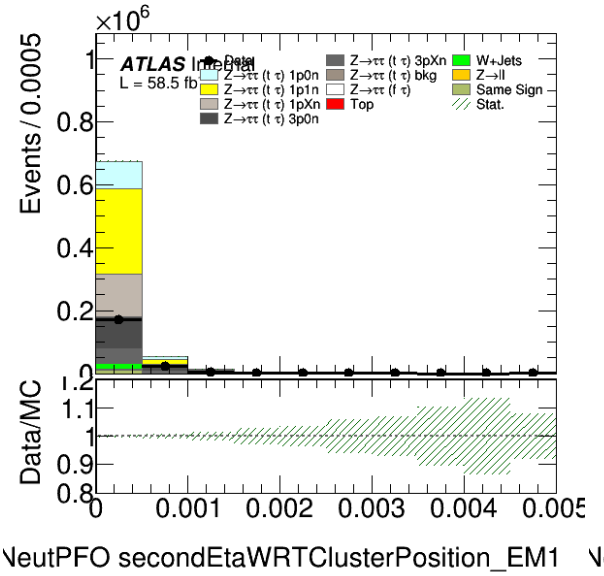


(e) RNN score == 3

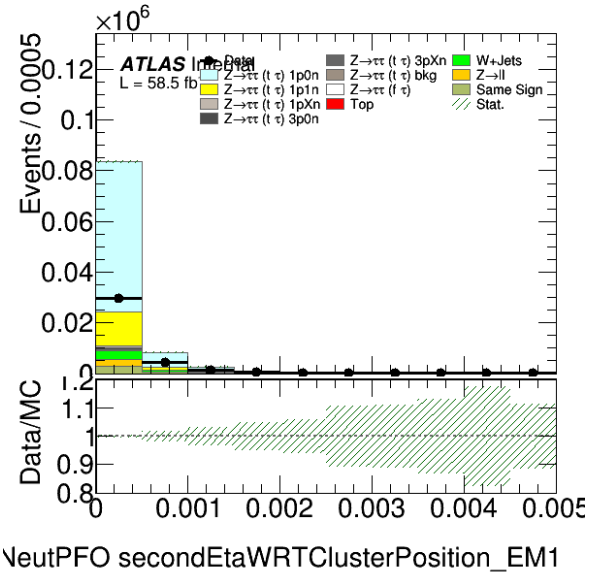


(f) RNN score == 4

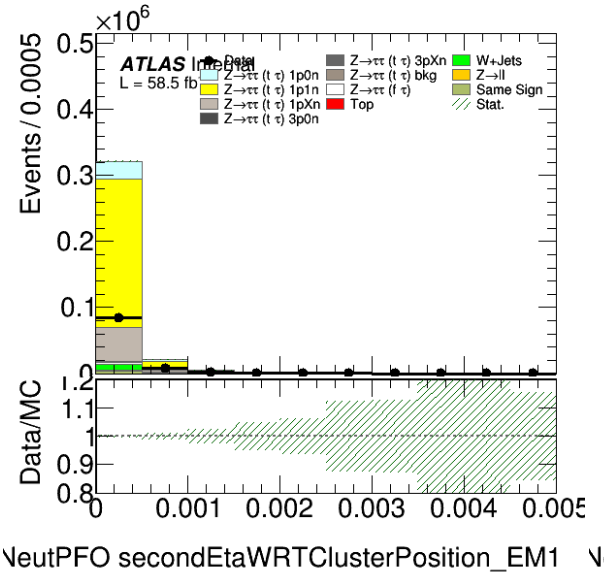
Figure 9.23: Neutral PFO: SECOND_R



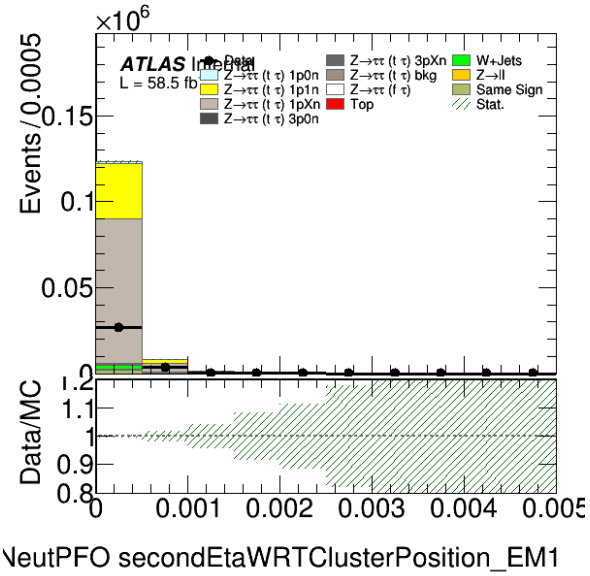
(a) no cut on RNN score



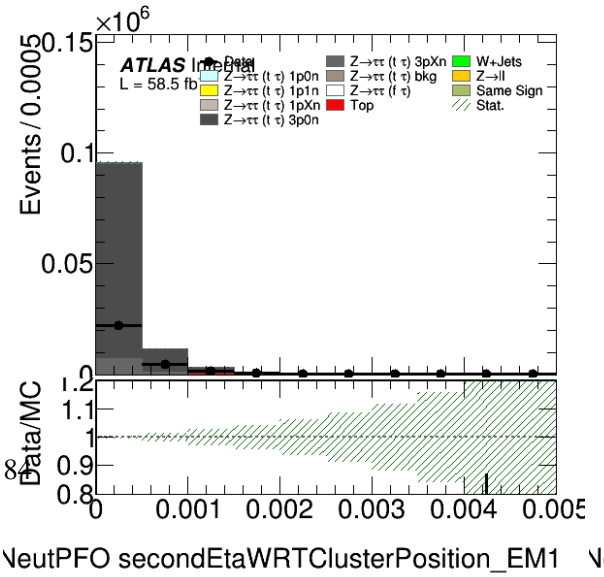
(b) RNN score == 0



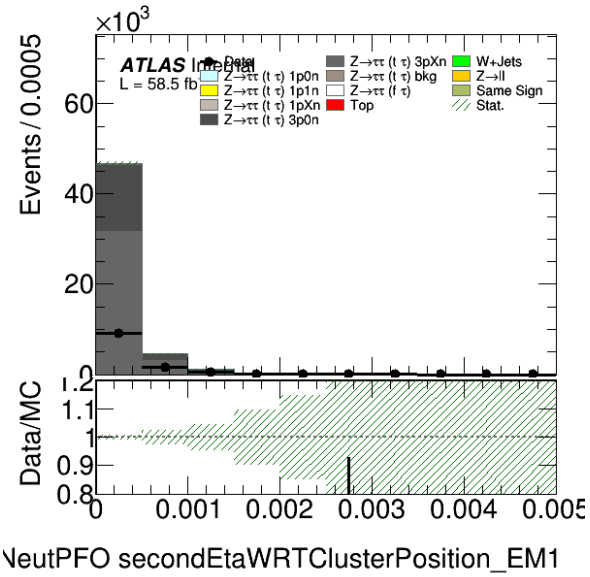
(c) RNN score == 1



(d) RNN score == 2

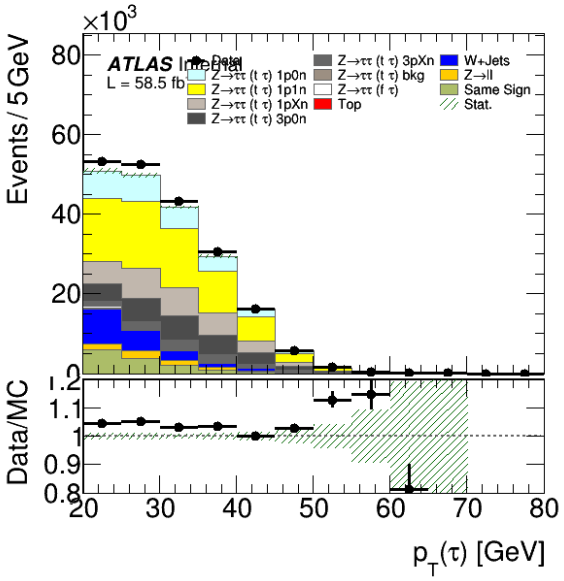


(e) RNN score == 3

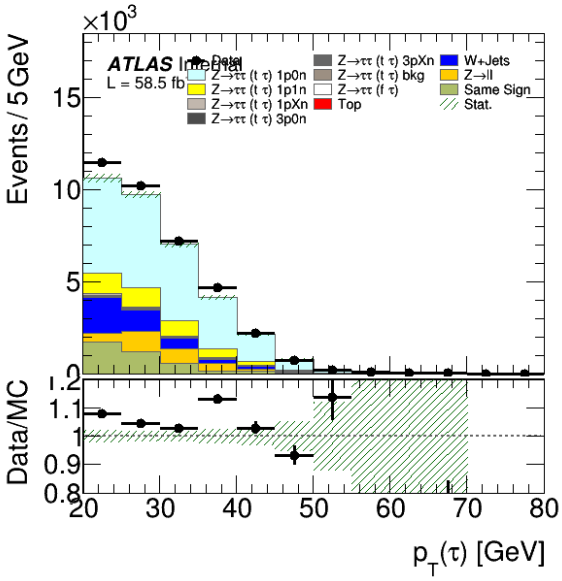


(f) RNN score == 4

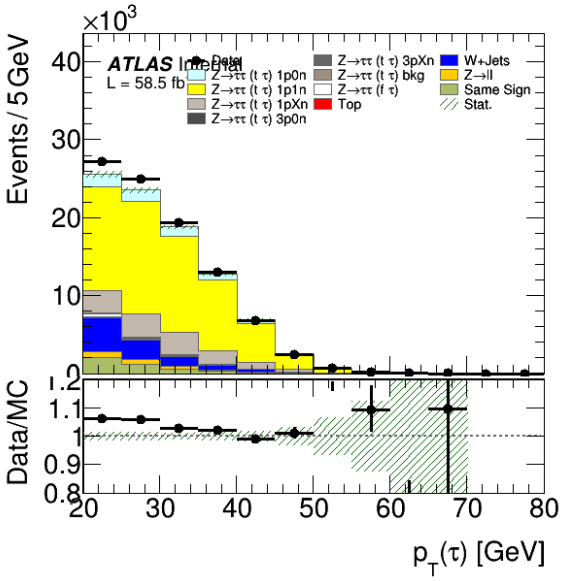
Figure 9.24: Neutral PFO: secondEtaWRTClusterPosition_EM1



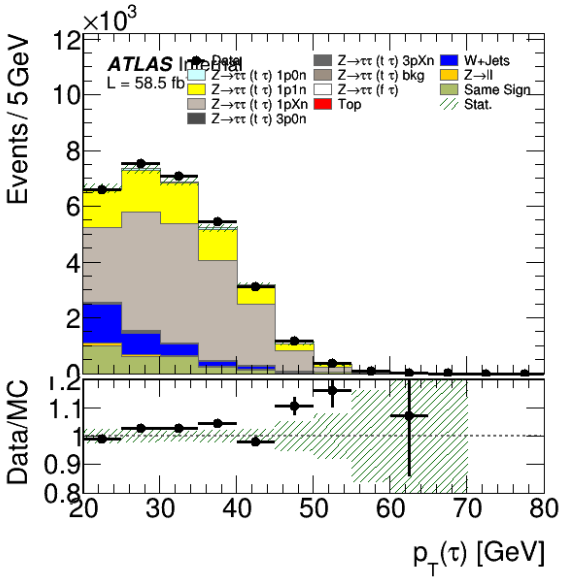
(a) no cut on RNN score



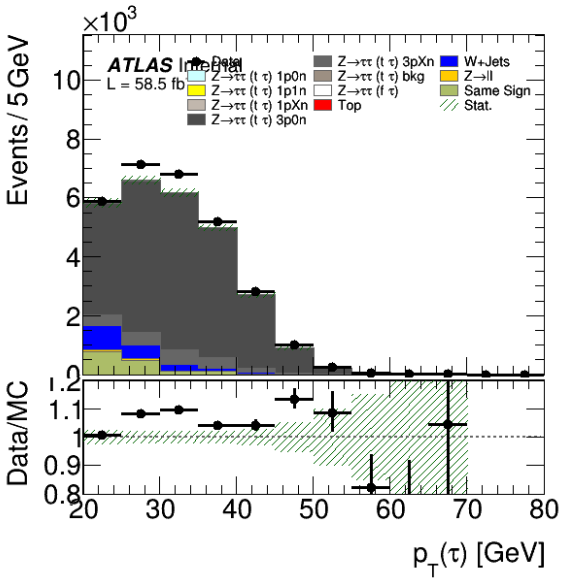
(b) RNN score == 0



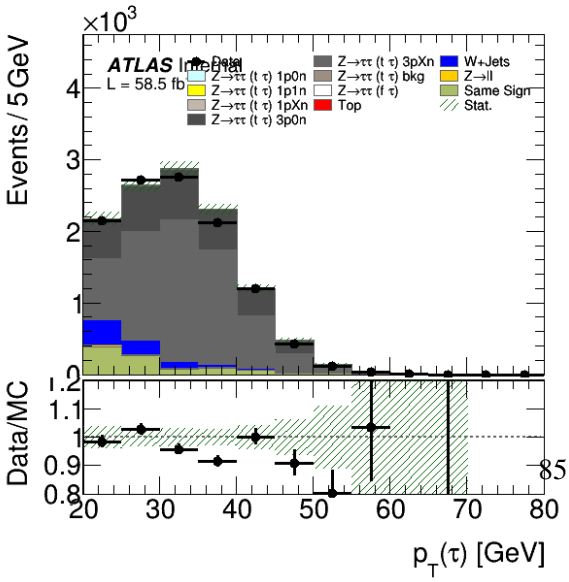
(c) RNN score == 1



(d) RNN score == 2

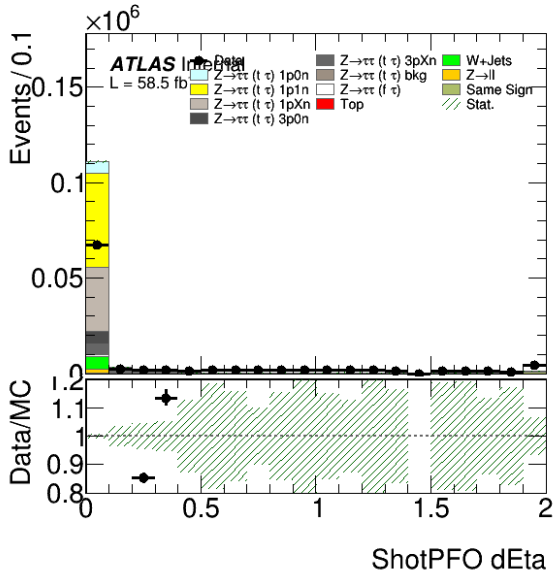


(e) RNN score == 3

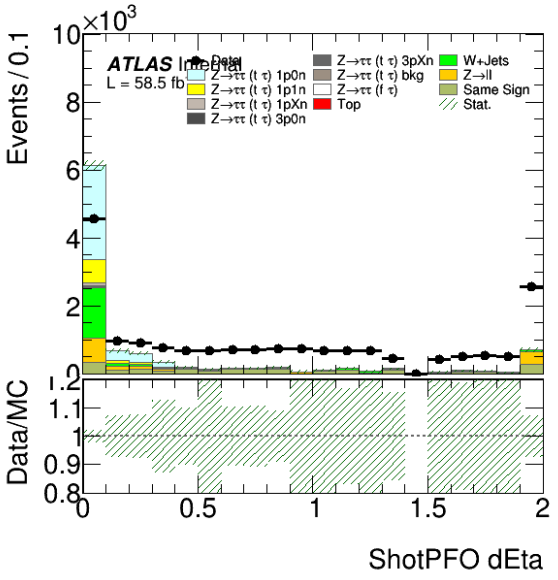


(f) RNN score == 4

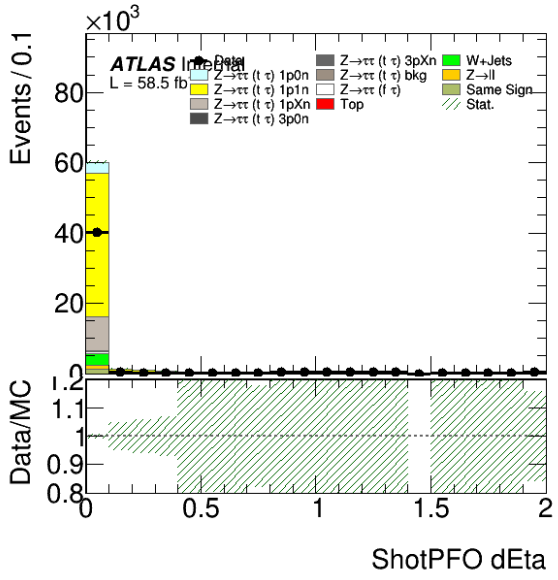
Figure 9.25: Tau_Pt



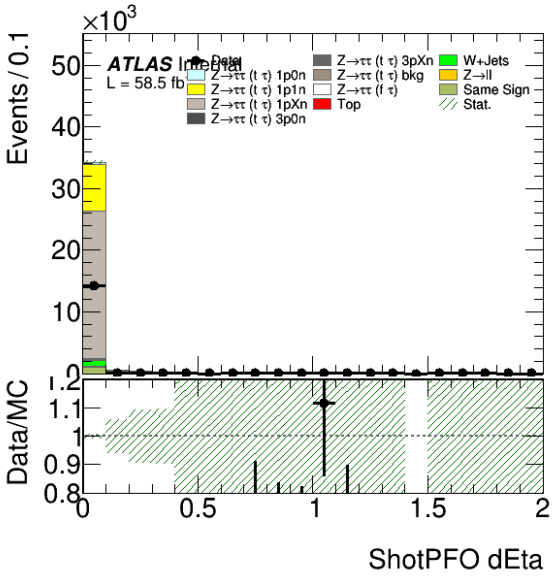
(a) no cut on RNN score



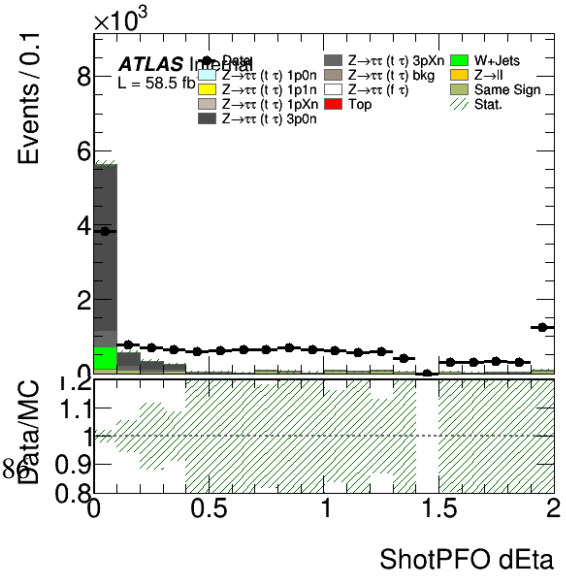
(b) RNN score == 0



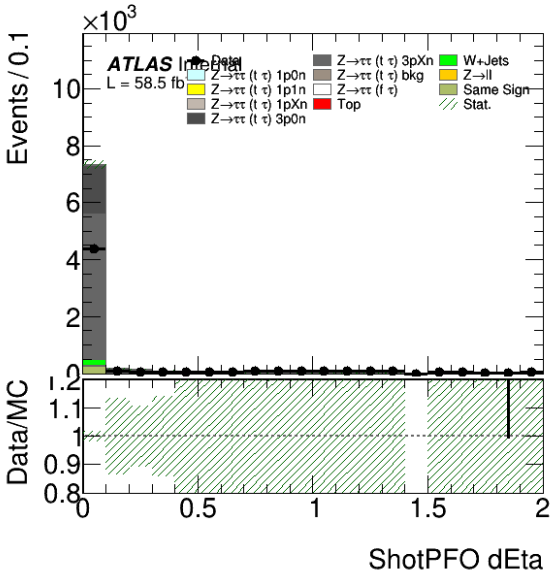
(c) RNN score == 1



(d) RNN score == 2

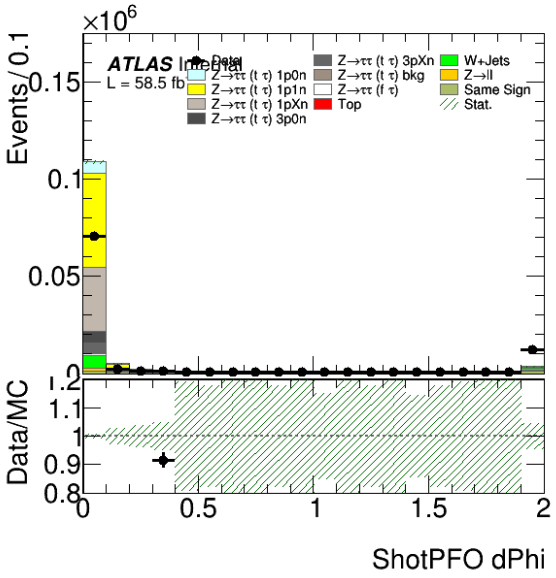


(e) RNN score == 3

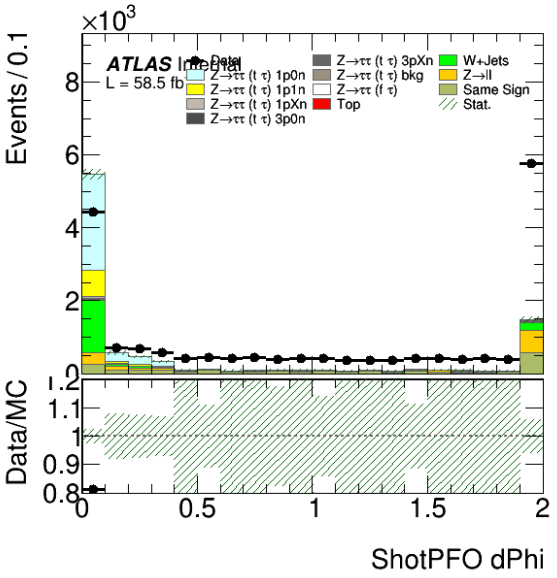


(f) RNN score == 4

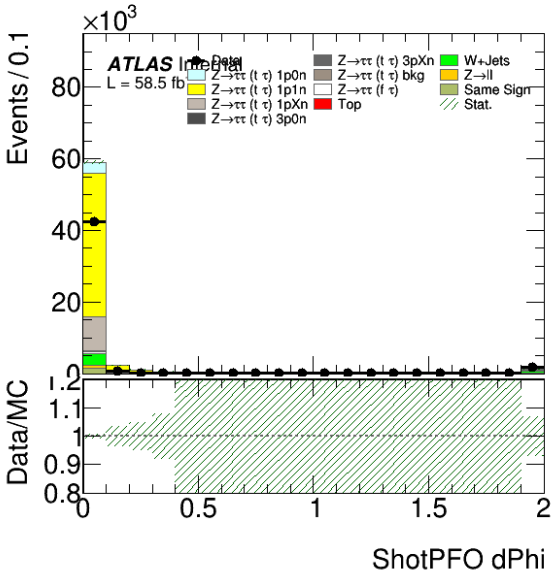
Figure 9.26: Photon shots: dEta



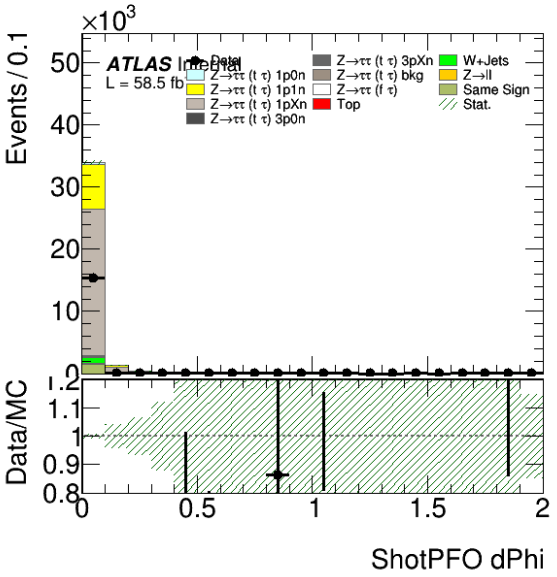
(a) no cut on RNN score



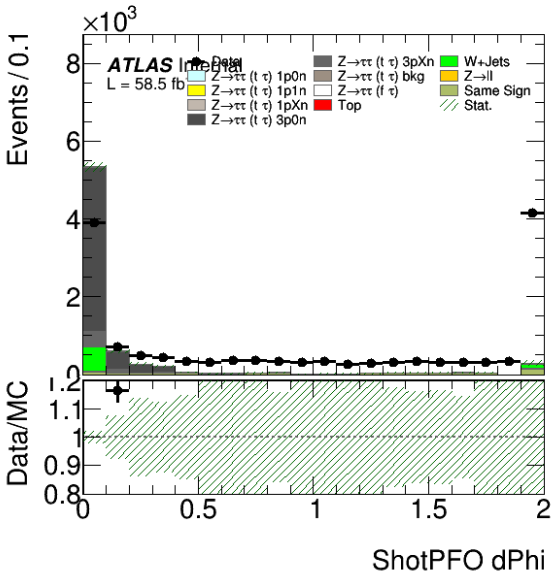
(b) RNN score == 0



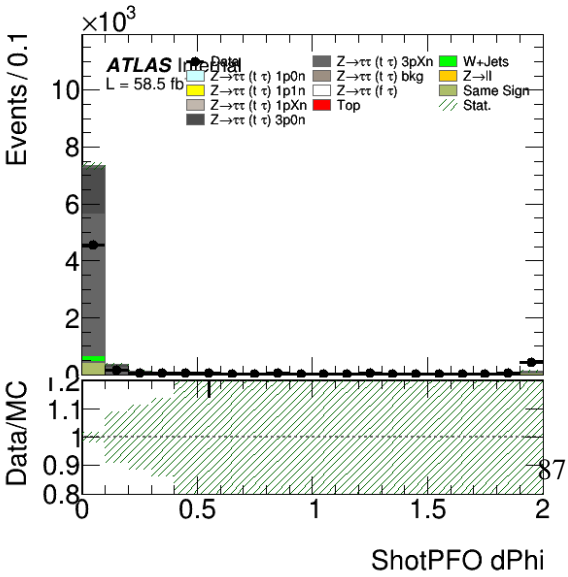
(c) RNN score == 1



(d) RNN score == 2

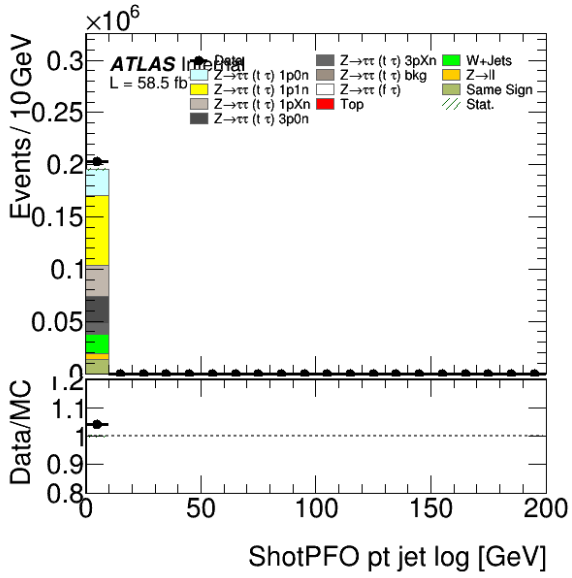


(e) RNN score == 3

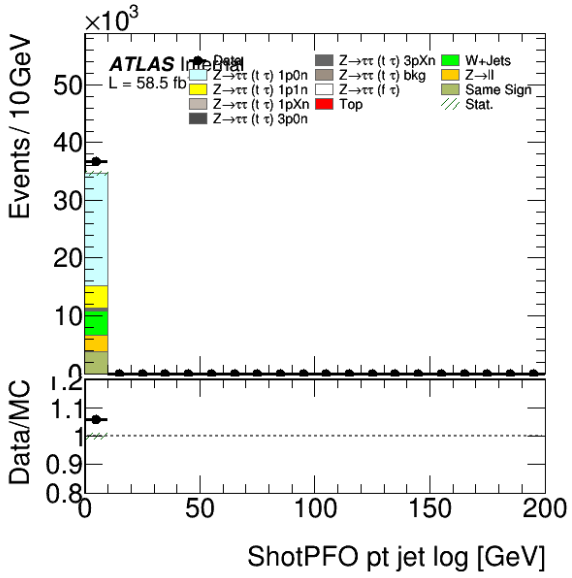


(f) RNN score == 4

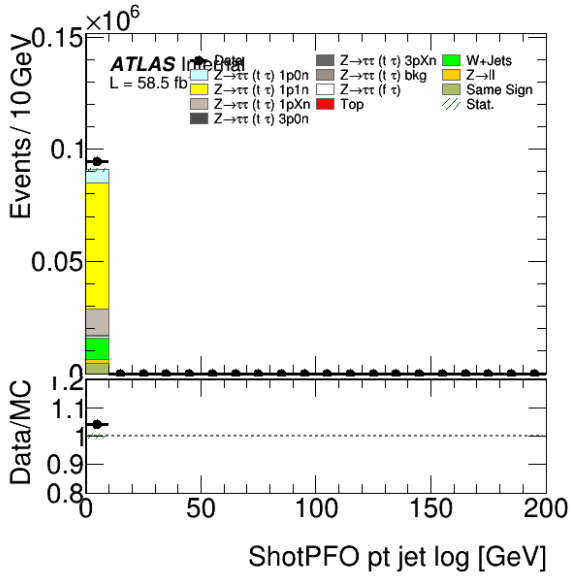
Figure 9.27: Photon shots: dPhi



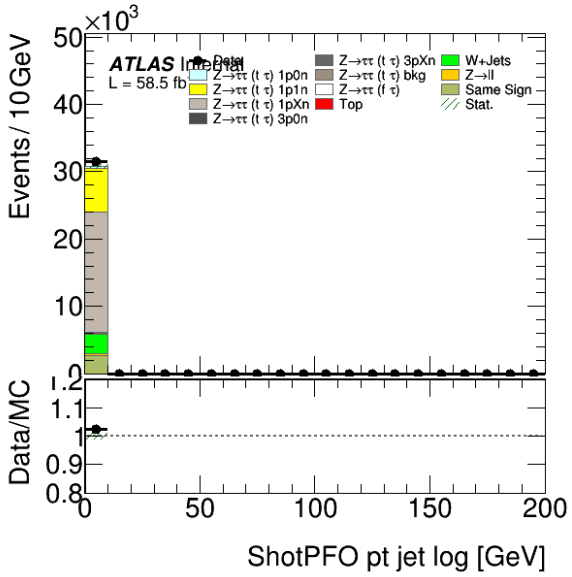
(a) no cut on RNN score



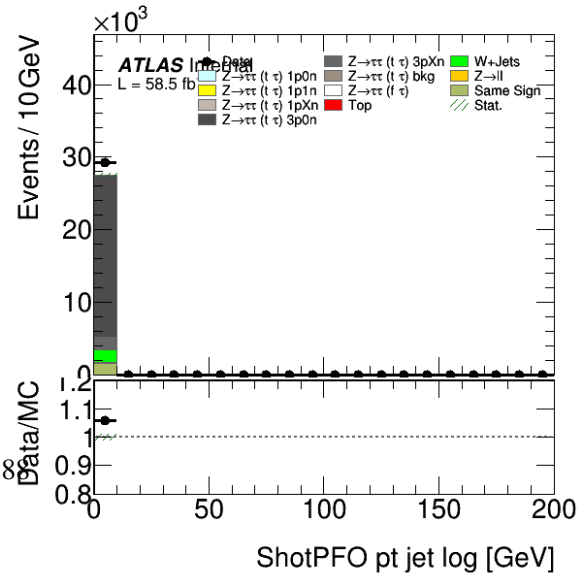
(b) RNN score == 0



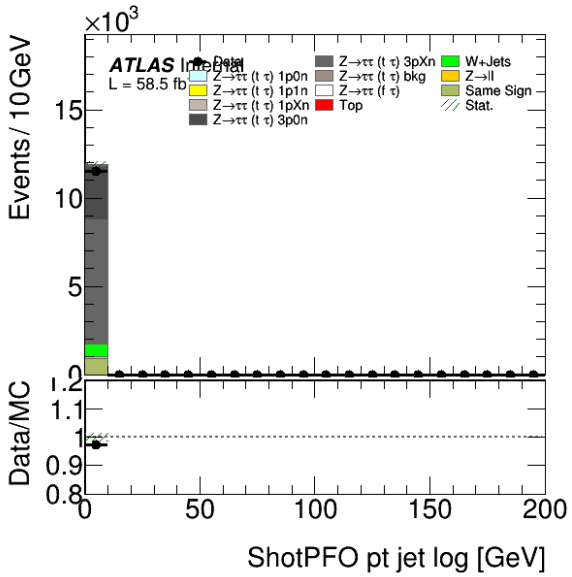
(c) RNN score == 1



(d) RNN score == 2

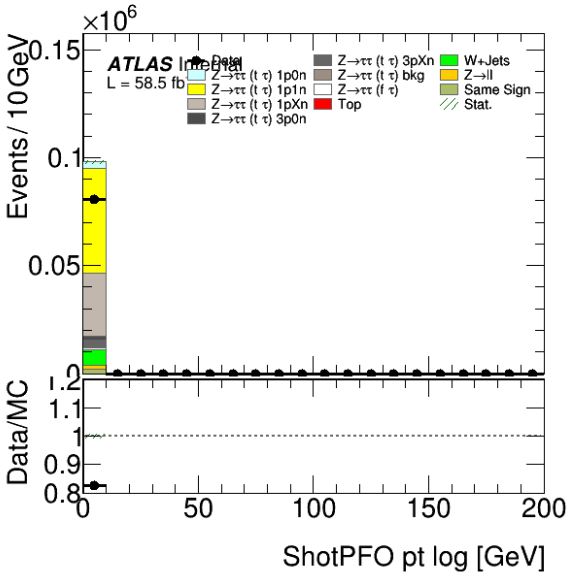


(e) RNN score == 3

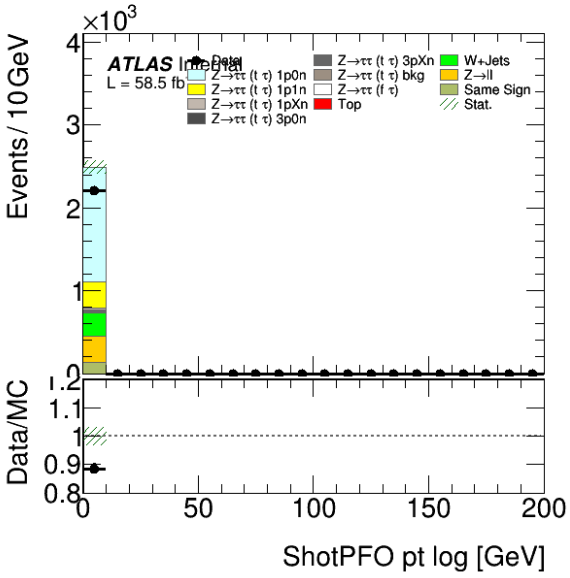


(f) RNN score == 4

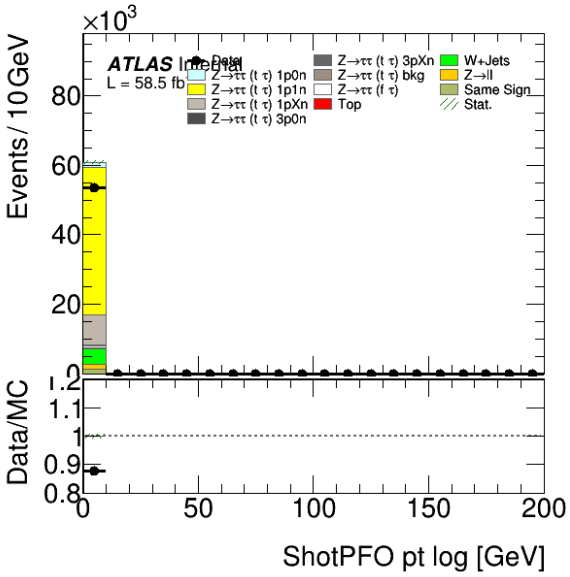
Figure 9.28: Photon shots: ptjetlog



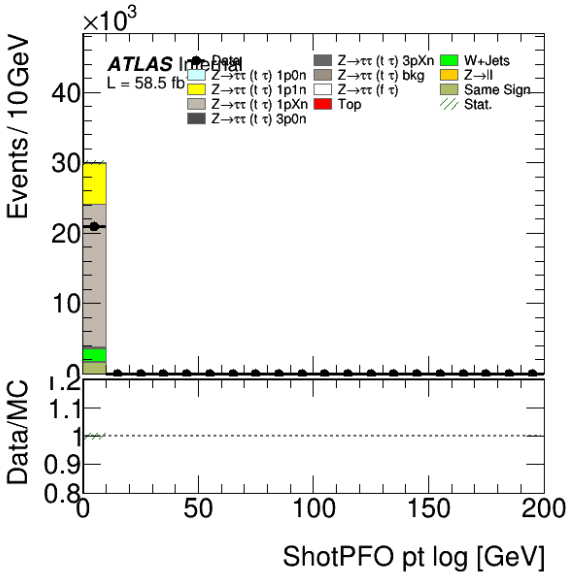
(a) no cut on RNN score



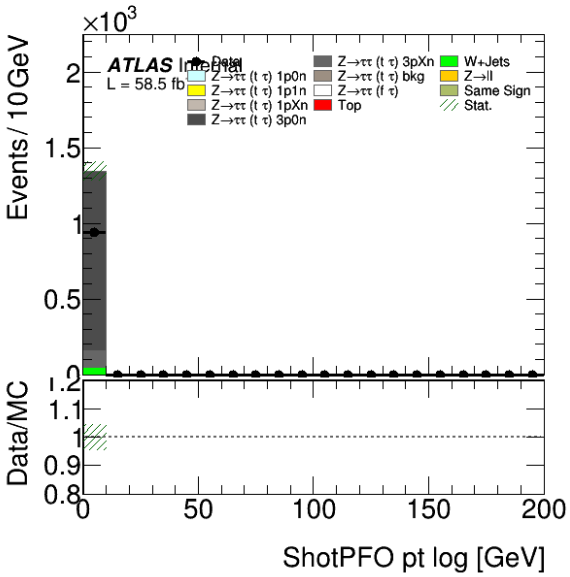
(b) RNN score == 0



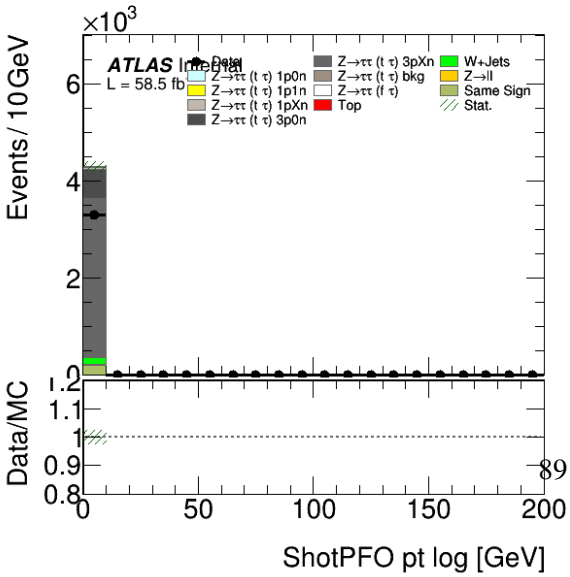
(c) RNN score == 1



(d) RNN score == 2



(e) RNN score == 3



(f) RNN score == 4

Figure 9.29: Photon shots: ptlog

List of Figures

List of Tables