Radboud University Nijmegen

Faculty of Science

Nikhef

Transformer regressing the mass of boosted Higgs bosons decaying to $b\bar{b}$

Master Thesis

MSc Physics and Astronomy

Particle and Astrophysics

Author:
Wouter Morren

Supervisor:
dr. Frank Filthaut

Second reader:
dr. Harm Schoorlemmer

Institute for Mathematics, Astrophysics and Particle Physics
Department of High Energy Physics

November 2023

Abstract

A transformer-based Machine Learning algorithm is developed to predict the Higgs jet mass from boosted Higgs bosons decaying into bottom-quark pairs, $H \to b\bar{b}$. It does so using data of simulated proton-proton collision at a centre-of-mass energy of $\sqrt{s} = 13$ TeV, containing jet, track and subjet information. Two target variables are investigated: the true jet mass (reconstructed from truth particles ending up in the large-R jet in the detector) and the kinematic mass (the invariant mass of the bottom pair). Furthermore, selections of the input variables are tested as well as different architectures to optimise the predictions. The model trained on the kinematic mass provided the best predictions, i.e. the strongest Higgs peak, the least affected p_T and $|\eta|$ distribution and the best resolution at high p_T. The Higgs peak using the regressed masses is described by $\mu = 128$ GeV and $\sigma = 10$ GeV, whereas the reconstructed large-R jet mass peak is described by $\mu = 125$ GeV and $\sigma = 22$ GeV. In terms of the RMSE, this is an improvement of factor 2.

Acknowledgements

During this project, I have had the pleasure of working with many people from different collaborations. They helped me get past difficult parts and provided me with code and data that significantly boosted this project. The majority of the code and data was beneficial to many of my colleagues as well, yet I appreciate everyone's effort as without them the results could not have been as good as they are now. After completing my bachelor's thesis during the Corona pandemic, it was great to experience during my master's thesis what it is like to perform research amongst so many motivated physicists. This, along with many other reasons, motivates me to continue my career in experimental particle physics.

I would like to take a moment to thank some people individually for their help and support during this project. First of all, I would like to sincerely thank Frank Filthaut for providing this amazing opportunity and for being my supervisor. I truly appreciate all your time and commitment to this project, whilst having such a busy schedule and travelling all across Europe. You really helped keep the momentum going by contacting the right people at CERN, while creating awareness of the potential of this project. This was very motivational to me and much appreciated.

Secondly, I would like to thank Geoffrey Gilles for all his contributions to this project. You helped me greatly with the technical setup and in getting familiar with the transformer model. Also, you provided me with different datasets and explained their contents. Without you, I would not have been able to produce so many interesting results.

Next, I would like to thank Jackson Barr, Samuel Van Stroud and Dmitrii Kobylianskii for their impressive efforts on the Salt code and the training data that was produced for the boosted Hbb/cc tagger and of great use to me in this project. Furthermore, I would like to acknowledge everyone else from the Boosted Hbb/cc Tagger Task Force for their interesting contributions to the tagger, which helped me place this project into context.

I would also like to give a shout-out to Marion Missio and Oliver Rieger for helping me out with all the problems I encountered with VSCode, SSH keys and GitLab. Also, I would like to acknowledge Osama Karkout for helping me deal with Umami while using the data that he provided me. Then, to everyone from the FTAG group and the Higgs Coffee meetings: your enthusiasm and feedback were much appreciated. Last but not least, to the entire Nikhef ATLAS group, thank you for making this year a success, I have really enjoyed it. I look forward to meeting you more often during my upcoming time as a PhD candidate in Amsterdam.

Contents

Chapter 1

Introduction

Back in 1964, the Higgs boson was first hypothesised by Peter Higgs. [1] It would explain how the particles in the old version of the Standard Model could be massive without violating the symmetries in the theory. It took 48 years to prove its existence, as it was discovered in 2012 by ATLAS and CMS at CERN. [2, 3] This particle finally completed the theory now well known as the Standard Model (SM) of particle physics. It is able to describe particle interactions well, which for instance take place at CERN. However, some parameters of the model are free, meaning that we do not have a derivation for them and they can only be fitted using data. Moreover, some experiments show results that cannot be described by the SM or that slightly deviate from theoretical predictions. Take for example the potentially deviating magnetic moment of the muon [4], unpredicted neutrino oscillations [5] and unexplained dark matter [6]. Even more unsatisfying is the fact that we cannot yet describe gravity and particle physics using one theory. Hence the search for new physics continues.

A key element in this search is the Higgs boson, because still little is known about this particle. For instance, the measurement of the Higgs self-coupling constant is yet inconclusive. Therefore, we do not know if nor why we live in a meta-stable universe. [7] Besides, the coupling of the Higgs boson to other elementary particles is related to the Higgs boson mass. The corresponding coupling constants are expected to be proportional to the particle masses. Any deviations can then be used to probe physics beyond the SM. In other words, many open questions are linked to the Higgs boson and the ability to answer them depends on the resolution of related measurements.

Right now the Higgs mass is measured to be 125.11 ± 0.11 GeV/c^2. [8] This value is found using $H \to \gamma\gamma$ and $H \to ZZ^* \to 4l$ decays.[1] However, $H \to \gamma\gamma$ has a branching ratio of about 0.2% and $H \to ZZ$ of about 3%. [9] Meanwhile, the decay $H \to b\bar{b}$ has a branching ratio of about 57%. Hence using these events in physics analyses would greatly increase the statistics. However, the production rate of the Higgs boson is low, resulting in only a small signal on top of a large multi-jet background. Distinguishing this specific process from the background is a challenging task as multiple processes produce similar decay products. The mass, which is reconstructed from decay products in the detector, can be a useful discriminating variable, as the mass of the decaying particle is very characteristic of a process. Therefore, we aim to improve the resolution of the reconstructed mass. This could then aid in selecting $H \to b\bar{b}$ events using jet taggers.

[1] The asterisk in Z^* denotes an off-shell particle.

Currently, the mass of a decaying particle is derived using traditional (though state-of-the-art) algorithms. When the decay products are easily distinguishable, e.g. in four-lepton events, the invariant mass can easily be calculated. In hadronic decays, the resolution of the mass is reduced, because signal and background particles may end up in the same jet. In this thesis, we are interested in the boosted regime, since it is more sensitive to physics beyond the SM. In boosted Higgs decays, i.e. when the Higgs bosons have high momenta, the two b jets overlap in the detector. This complicates the reconstruction of the jet and the objects inside of it, as well as the discrimination of background particles. Currently, in such decays, the reconstruction algorithms aim to remove background particles in the detector, correct the measured values of the remaining particles and construct the mass from that. However, sometimes too few or too many particles are removed causing the subsequent adjustments to be incorrect. This is where Machine Learning (ML) algorithms could play an important role, as they could potentially perform this complicated task better. Therefore, we aim to improve the mass resolution by performing a regression task using a transformer-based ML algorithm.

In Chapter 2, the ATLAS detector is described, as well as the physics behind the process of interest, i.e. $H \to b\bar{b}$. In Chapter 3, the simulation and contents of the training and physical SM evaluation samples are described, after which selections on the input variables and objects are made. Furthermore, the possible architectures of the ML models and their properties are described. In Chapter 4 the ML models trained on the two different target variables are optimised, presented, tested and discussed. Finally, in Chapter 5 we conclude our findings.

Chapter 2

Particle physics

2.1 The Standard Model

The Standard Model of particle physics is a model that contains a set of particles and that describes the interactions between them. The particles can be divided into two groups: *fermions* having half-integer spins and *bosons* having integer spins. Fermions are the building block of matter and can be divided into *quarks* and *leptons*. All particles have an antiparticle, which has the same mass and spin, but opposite charge. Neutrinos do not have an electrical charge, but they are expected to have antiparticle variants. The Z boson and the photon, on the other hand, are their own antiparticles. All particles of the SM and their properties are shown in Figure 2.1.

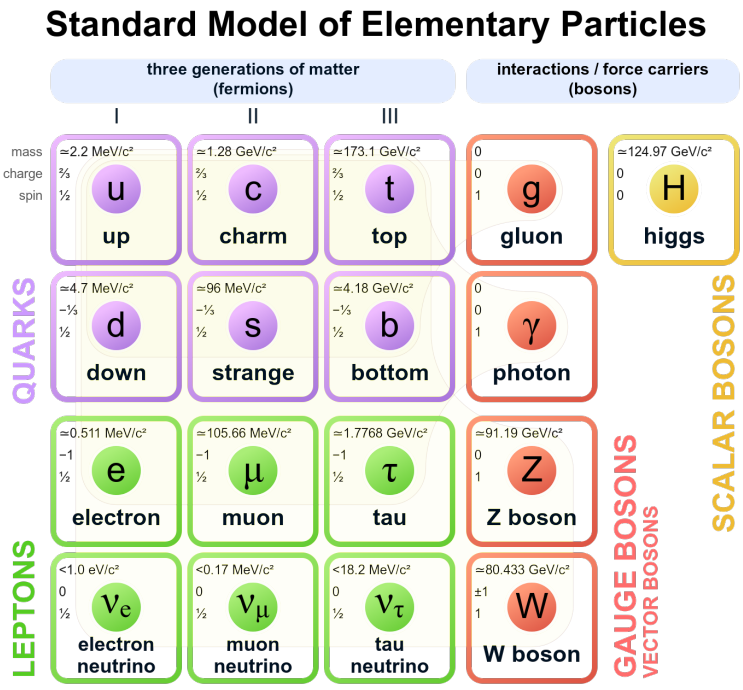


Figure 2.1: The Standard Model of particle physics. [10]

Quarks (q), unlike leptons, have a so-called colour charge, causing them to only appear in colourless combinations of quarks called *hadrons*. Hadrons with one quark and one antiquark ($q\bar{q}$) are called *mesons* and those with three quarks (qqq) are called *baryons*. Those two are the most common kinds of hadrons. The best-known hadrons are the proton (udd) and the neutron (uud). An example of a meson is the B meson containing a bottom antiquark and either an up, down, strange or charm quark ($q\bar{b}$). [11]

Leptons (l) can be divided into charged leptons (ℓ), having charge $\pm 1e$, and chargeless leptons, called neutrinos (ν_ℓ or simply ν). The antileptonic counterpart of a charged lepton is denoted with a plus instead of a minus, e.g. e^+ instead of e^-, whereas the antineutrino is denoted with a bar, just like the quarks, i.e. $\bar{\nu}$.

The bosons can be divided into four gauge bosons, which are the force carriers in the SM, and one scalar boson, which is the Higgs boson. The gluon (g) is responsible for the strong force, binding quarks into hadrons. It carries a colour charge and therefore only couples to coloured particles, i.e. quarks and other gluons. All processes governed by gluon interactions can be described by Quantum Chromodynamics (QCD). The photon (γ) only couples to charged particles and it mediates the electromagnetic force. The related processes are described by Quantum Electrodynamics (QED). The $W^\pm$ and Z^0 bosons are called the weak bosons and they carry the weak force. They couple to all fermions, including the neutrino. Hence the neutrino only couples to the weak bosons. The weak bosons and the photon as well as their interactions can all be described by the Electroweak force. [12]

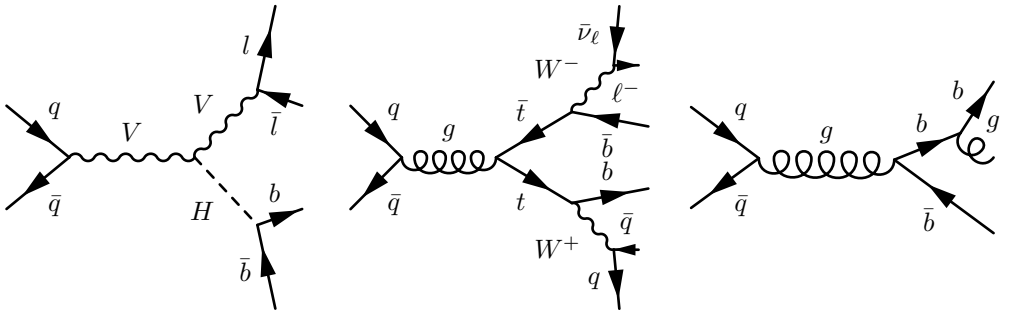
The Higgs boson (H) is the particle that gives mass to all other particles in the SM via the so-called Higgs mechanism. The coupling strength between the Higgs boson and other particles is proportional to the mass of those particles. Hence heavier particles are more likely to emerge in Higgs decays, provided that the sum of the masses of the decaying particles is not larger than the Higgs mass. For instance, the decay $H \to t\bar{t}$ would not be kinematically possible, unless one of the top quarks is a virtual particle decaying to another lighter real particle.

2.2 Higgs decay channels

In the ATLAS detector, millions of protons collide every second with enormous energies, allowing for the creation of new particles. Very rarely an on-shell Higgs boson is produced, but due to its high mass, it quickly decays into other particles. The most likely decay mode of the Higgs boson is $H \to b\bar{b}$, which is sometimes denoted as $H \to bb$, without the bar above the b. This decay mode is hard to extract from the detector data, since many processes produce a $b\bar{b}$-pair and thus similar jets in the detector. Moreover, after the decay the b quarks start to hadronise, meaning that two B hadrons[1] (a b quark with one or more other quarks) are created. These hadrons can then decay into numerous other particles again, which complicates selecting the right jets.

To be able to discriminate $H \to b\bar{b}$ events more efficiently from background processes, it is useful to look at the decay mode in which a weak boson is associated with the Higgs boson as depicted in Figure 2.2a. One can then select events with 0, 1 or 2 charged leptons ending up in the detector, which reduces the relative background contribution in the 0-, 1- or 2-lepton channel respectively. Meanwhile, the neutrinos remain undetected as they barely interact with matter. Below we list a few interesting signal events, as well as some background processes that could spoil the measurements.

[1] We also imply their antiparticles.



(a) $H \to b\bar{b}$ **(b)** Top **(c)** QCD

Figure 2.2: Feynman diagrams of (a) the general weak associated $H \to b\bar{b}$ decay, i.e. $q\bar{q} \to V \to VH \to l\bar{l}b\bar{b}$, as well as of example background processes involving (b) a top pair, i.e. $q\bar{q} \to g \to t\bar{t} \to \ell\bar{\nu}_\ell q\bar{q}b\bar{b}$ and (c) QCD multijets, i.e. $q\bar{q} \to g \to gb\bar{b}$. In the latter process, the rightmost gluon is able to produce even more jets. All processes result in a bottom pair and a certain number of leptons and jets.

Signal events:

- $Z \to ZH \to \ell\ell bb$ (2-lepton channel)
- $W \to WH \to \ell\nu bb$ (1-lepton channel)
- $Z \to ZH \to \nu\nu bb$ (0-lepton channel)

Background events:

- QCD, which includes any process that involves gluons, but no Higgs boson, decaying into a bottom pair, e.g. $g \to b\bar{b}$.
- Top, which denotes any process in which a top pair is produced, which then decays to a bottom pair via W bosons. The two W bosons could then decay into a pair of quarks or to a lepton-neutrino pair. Therefore, this process can contribute to the 0-, 1- and 2-lepton channels.

2.3 ATLAS

The ATLAS detector [13] at the Large Hadron Collider (LHC) is a particle detector that can be used in numerous fields of research. It has a cylindrical geometry that is forward-backward symmetric and it covers nearly the entire solid angle around the collision point. A cut-away view of the detector can be found in Figure 2.3. The coordinate system used in ATLAS is right-handed. Its origin is at the centre of the detector, i.e. at the so-called nominal interaction point (IP). The z-axis is aligned along the beam, the x-axis points from the IP to the centre of the LHC ring, and the y-axis points upwards.

Points in this coordinate system are denoted using the coordinates r, ϕ and η. The cylindrical coordinates r and ϕ are used in the transverse plane, with $\phi \in [0, 2\pi)$ the azimuthal angle around the z-axis. The *pseudorapidity*, denoted by η, is defined in terms of the polar angle $\theta \in [0, \pi]$ as $\eta = -\ln\tan(\theta/2)$. Hence $\eta \in (-\infty, \infty)$, but typically most values lie within $\eta \in [-3, 3]$. Angular distance is defined by $\Delta R \equiv \sqrt{(\Delta\eta)^2 + (\Delta\phi)^2}$.

To understand the contents of the data that we will be using, we will briefly describe the components of the detector. The detector consists of an inner tracking detector, electromagnetic and hadronic calorimeters, and a muon spectrometer. The inner tracking detector, or just the Inner Detector (ID), is surrounded by a thin superconducting solenoid providing a 2 T axial magnetic field. This provides the reconstruction of

charged-particle tracks in the range $|\eta| < 2.5$.

The ID system consists of three subsystems: a Pixel Detector, a Semiconductor Tracker (SCT), and a Transition Radiation Tracker (TRT). The Pixel Detector's inner radius is 34 mm and it can detect small energy deposits using silicon pixels. It contains an insertable barrel layer (IBL), which is the innermost pixel layer (out of four) covering the vertex region. The Pixel Detector is surrounded by the Semiconductor Tracker consisting of silicon strips. Both subsystems usually provide four 2D space points per track. The surrounding Transition Radiation Tracker ($|\eta| < 2.0$) containing drift tubes provides radially extended track reconstruction and electron identification information.

The ID is surrounded by electromagnetic (EM) calorimeters, which measure showers of electrons, photons and to some extent hadrons. They are again surrounded by hadronic calorimeters, which measure the energy deposits from hadrons passing through the EM calorimeters. Both calorimeters cover $|\eta| < 4.9$. The EM calorimetry is provided by barrel and endcap high-granularity lead/liquid-argon (LAr) calorimeters ($|\eta| < 3.2$). Hadronic calorimetry is provided by three steel/scintillating-tile barrel calorimeters ($|\eta| < 1.7$), and two copper/LAr endcap calorimeters ($1.5 < |\eta| < 3.2$). Finally, the forward regions ($3.2 < |\eta| < 4.9$) are covered by copper/LAr and tungsten/LAr calorimeters, optimised for EM and hadronic showers respectively.

The muon spectrometer consists of triggering and high-precision tracking chambers. The muon trigger system ($|\eta| < 2.4$) is composed of resistive plate chambers in the centre region and thin gap chambers in the endcap regions. The separate precision chamber ($|\eta| < 2.7$) measures the deflection of muons in a magnetic field that is generated by three superconducting toroidal magnets. It contains three layers of monitored drift tubes and, in the forward region, it contains cathode strips. [14]

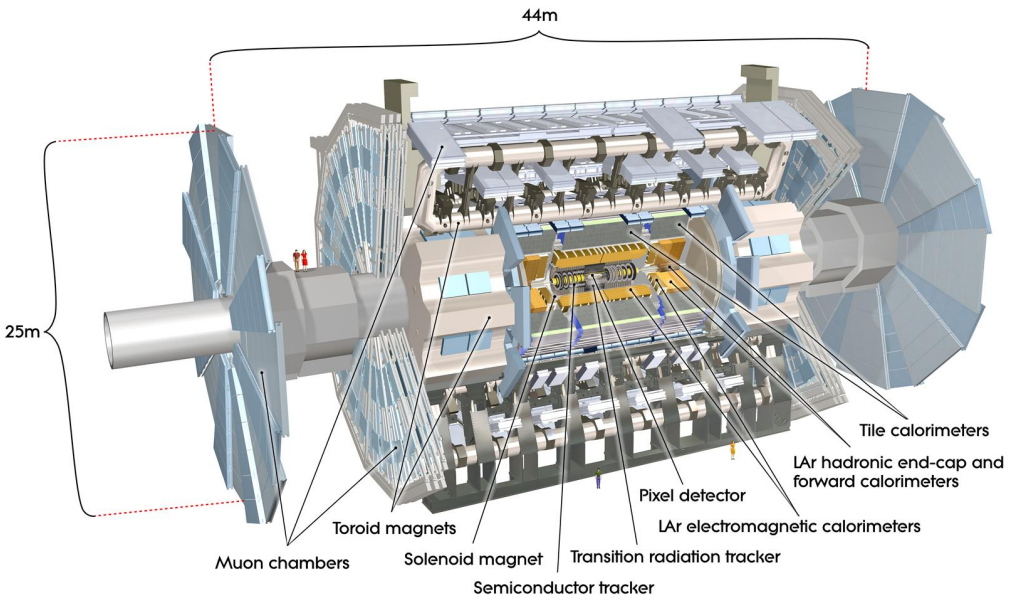


Figure 2.3: A cut-away view of the ATLAS detector. [13]

Chapter 3

Methods

In this chapter, the necessary definitions and properties related to the simulated data and the ML algorithm are explained. In this project, a transformer-based ML model was used to improve the resolution of the reconstructed jet mass. For the training of this model, a suitable dataset is generated and a representative SM dataset is used for the evaluation of the model. From the signals in the different parts of the detector, many different objects and variables can be defined and constructed. We will be using the variables describing tracks, jets and subjets as input for the ML model. This chapter will explain these objects and variables and from them, a suitable selection is made for the training. After that, the transformer model is explained, as well as related ML hyperparameters that need to be investigated and optimised.

Note that the data, model and code are similar to those used by the Boosted Hbb/cc Tagger Task Force. This group trains classification ML algorithms, also called taggers, like the GN2X tagger. It aims to filter boosted $H \to b\bar{b}$ and/or $H \to c\bar{c}$ events from the background using a transformer-based neural network. [15] Since we are aiming to improve the mass resolution of similar events, which could then aid these taggers, it makes sense to use a similar setup. However, different input objects and variables will eventually be used. Also, the architecture is adjusted and optimised for our task and the code is adjusted to be able to perform regression as a main task rather than classification.

3.1 Data generation

Simulation data at ATLAS can be generated for many different purposes, for example for calibrations, physics analyses, or - like in our case - for training ML models. This data is generated using Monte Carlo (MC) simulations. Using probability density functions, these simulations aim to deliver events that together provide the desired distributions, e.g. distributions as predicted by the SM.

The data used in this thesis consists of jets that are simulated from proton-proton (pp) collisions at a centre-of-mass energy of $\sqrt{s} = 13$ TeV. Four processes are considered, namely $H \to b\bar{b}$, $H \to c\bar{c}$, top and QCD and they are simulated separately. The specific decay chains that are included are detailed in tables 3.1 and 3.2. Only $H \to b\bar{b}$ and $H \to c\bar{c}$ jets that are associated with a Z boson (ZH) are considered.

The training and evaluation data are simulated in two different ways. The training data containing ZH production is generated with a biased phase-space sampling using an

Table 3.1: Simulation details of the training data with corresponding generator versions, tunes and PDF sets.

Jet	Process	Generator	Tune	NNPDF
$H \to b\bar{b}$	$q\bar{q} \to ZH, Z \to \mu^+\mu^-$	PYTHIA 8.306 [17]	A14 [18]	2.3 LO [19]
$H \to c\bar{c}$	$q\bar{q} \to ZH, Z \to \mu^+\mu^-$	PYTHIA 8.306	A14	2.3 LO
Top	$Z' \to t\bar{t}$	PYTHIA 8.235	A14	2.3 LO
QCD	Multijet	PYTHIA 8.235	A14	2.3 LO

Table 3.2: Standard Model simulation details of the evaluation data with corresponding generator versions, tunes and PDF sets. Here $Z \to \ell\bar{\ell}/\nu\bar{\nu}/q\bar{q}$, where $\ell = e, \mu$.

Jet	Process	Generator	Tune	NNPDF
$H \to b\bar{b}$	$q\bar{q}/gg \to ZH$	POWHEG V2 + PYTHIA 8.212 [20]	AZNLO [21]	3.0 NLO
$H \to c\bar{c}$	$q\bar{q}/gg \to ZH$	POWHEG V2 + PYTHIA 8.212	AZNLO	3.0 NLO

artificially wide Higgs decay width of 400 GeV. Then selecting only kinematic masses between 50 and 200 GeV results in a nearly uniform mass distribution within that range. [16] Having a uniformly distributed target variable, rather than a peaked one, prevents an ML model from solely predicting the mean value, i.e. the artificially set Higgs mass. This would make the model insensitive to physics beyond the Standard Model (BSM) and background processes. The final training dataset does not contain a completely flat kinematic mass distribution due to for instance the selection criteria in the jet reconstruction algorithms. Similarly, top events are created with a hypothetical Z' boson to obtain a flatter p_T (transverse momentum) distribution.

The evaluation data is a simulation of certain SM events, where the Higgs mass is set to 125 GeV. In the evaluation data, only $H \to b\bar{b}$ and $H \to c\bar{c}$ events are considered. Throughout this thesis, we will only present the results on $H \to b\bar{b}$ events. In the appendix, the predictions on $H \to c\bar{c}$ events and background processes, i.e. QCD and top, are presented as well.

The exact generation methods for the training and evaluation data are detailed in Table 3.1 and 3.2 respectively, together with the versions of the generator, tunes and parton distribution functions (PDF). The specific tune determines how the data is adjusted, and it aims to better match the properties and distributions of the processes with the experimental data. The PDFs describe the probability distribution of the momentum and energy carried by quarks and gluons within a proton or other hadrons involved in a collision. In the training data, the decays of bottom and charm hadrons are modelled by EVTGEN V1.6.0, and in the evaluation data EVTGEN V1.2.0 is used. [22]. The events are then passed through the ATLAS full detector simulation [23]. This simulation takes into account the effect of multiple pp interactions per bunch crossing, as well as the effect on the detector response due to interactions from previous and upcoming bunch crossings. [15]

In this thesis, two different datasets are considered. One dataset was originally produced by the Boosted Hbb/cc Tagger Task Force to train for instance the GN2X tagger. [15] This dataset contains only one of the two target variables that we want to train our ML on, i.e. the true jet mass. Therefore, we will call this sample the true jet mass sample. The other dataset is generated for this project and it contains the other target variable, i.e. the kinematic mass. Therefore, we will call this sample the kinematic mass sample.

3.2 Jet definition

When the proton beams collide in the detector, particles are produced and emitted in all directions. These particles sometimes have high momenta, hence when they decay their decay products somewhat end up in the same direction. Moreover, particles hitting the calorimeter create electromagnetic or hadronic showers, creating localised energy deposits. In other words, particle decays can create structures in the detector. This can help us identify and reconstruct the actual process that took place during a collision.

In boosted $H \to b\bar{b}$ decays, all the particles resulting from the Higgs boson belong to one so-called *jet*, which is a somewhat cone-shaped object in which all the particles end up. The next decaying particle, i.e. the b hadron, could create a jet on its own, which usually results in a slightly smaller jet in the detector. That is because the daughter particle is less energetic, causing it to decay into fewer particles. Also, it decays slightly further away from the interaction point, which prevents the decay products from spreading as much before they hit the detector. Different kinds of cone-shaped jets can therefore be defined and reconstructed.

3.2.1 Large-R jets

Jets in the ATLAS detector are reconstructed using FastJet [24] with the anti-k_t algorithm [25] and a specific radius parameter R denoting the typical angular dimension of the jet. This reconstruction allows for measurements on the particles within the jet, while excluding much of the background. When the primary decaying particle does not have a high momentum, i.e. in the *resolved* case, we are more likely to have multiple smaller separated jets ending up in the detector, rather than one big jet. Such jets are best described by small-radius jets or *small-R* jets, which have $R = 0.4$.

With the aim of finding BSM physics, we are interested in the so-called *boosted* regime, in which the decaying Higgs boson has a high transverse momentum of about 250 GeV or higher. The angular separation between its decay products can be approximated by $R \sim 2m_H/p_T$, with m_H being the mass of the Higgs boson and p_T its transverse momentum, i.e. $p_T^2 = p_x^2 + p_y^2$. For p_T of about 250 GeV, this opening angle is approximately 1. For higher momenta the decay products of the two bottom quarks will overlap, making it impossible to correctly describe two small-R jets with the correct constituents. Therefore, in the boosted regime it is more useful to consider large-radius jets, or *large-R* jets with $R = 1.0$.

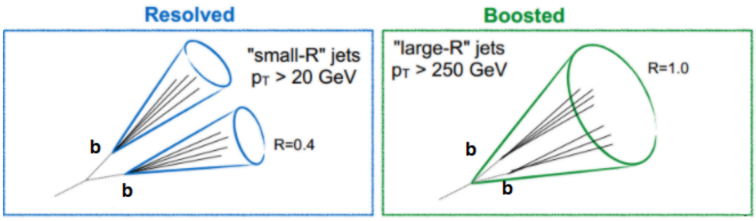


Figure 3.1: An illustration of the small-R jet in the resolved case versus the large-R jet in the boosted case. [26]

3.2.2 UFO jets

The large-R jet can be built from different kinds of objects, each having its advantages and disadvantages. Besides the information from localised energy deposits in

the calorimeter, tracker information can be used to define different detector objects. Particle-Flow (PFlow) Objects, or PFOs for short, best describe low p_T jets. Track-CaloClusters (TCCs) on the other hand best describe high p_T jets. These objects can be used as input for the jet reconstruction algorithm. Unified Flow Objects (UFOs) take advantage of PFOs and TCCs to achieve optimal performance across a wide kinematic range. [27] The data used in this thesis therefore consists of large-R jets constructed from UFOs.

The resulting UFO large-R jets are only angular selections on the η,ϕ-plane and therefore are still able to for example contain particles piling up from the underlying event. The variables that can be calculated from the large-R jet constituents will thus be slightly off. To improve the jet definition, many grooming techniques have been developed, like Trimming, Pruning and Soft-Drop. The UFO large-R jets in this thesis are adjusted with the Soft-Drop algorithm [28] along with Constituent Subtraction [29] and SoftKiller [30] in order to remove background from the same event and pile-up from other simultaneous collisions. Soft-Drop (SD) is a technique for removing soft and wide-angle radiation as depicted in Figure 3.2. Constituent Subtraction (CS) mainly aims to remove pile-up by uniformly adding ghost particles, which represent background particles, and removing them from their nearest neighbouring particle. After that SoftKiller (SK) applies a certain p_T cut onto all jets to remove soft radiation. The application of these three algorithms is motivated in [27].

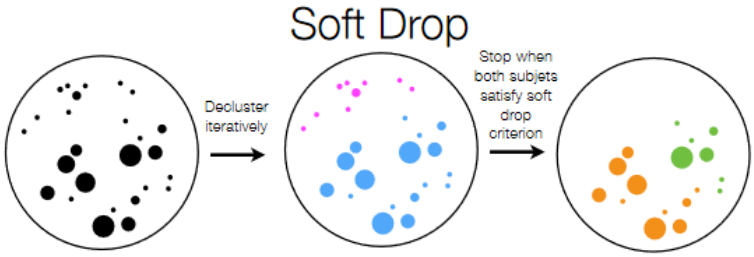


Figure 3.2: An illustration of the Soft Drop grooming method. [31]

Besides large-R jets, we will be making use of *subjets*. These are jets with a variable radius (VR) defined as $R = \rho/p_T$, where p_T is the transverse momentum of the subjet and $\rho = 30$ GeV. The minimal radius is 0.02 and the maximum is 0.4. The subjets are reconstructed with the VR anti-k_t algorithm. [32] The VR subjets are associated with the large-R jet, they are required to have $p_T > 7$GeV. These subjets are to be the jets resulting from hadronisation and nuclear fragmentation, which result in a number of final-state hadrons or leptons, as well as from secondary decays, e.g. the decays of the B hadrons.

3.2.3 Jet mass

Now that we have defined a jet, we can look into different definitions of the mass. First, a few definitions will be presented, after which we will discuss the target variable of the ML algorithm.

The kinematic mass

The *kinematic mass* (m_{kin}) is the mass that can be calculated using all the truth particles resulting from the Higgs decay within the simulation. That is at generator level in the Monte Carlo (MC) simulation of the event, in which all the truth information about

the particles is available. Using all the momenta and energies of the truth particles, the kinematic mass can be calculated using the energy-momentum relation as follows: [33]

$$m^2_{\text{kin}} = \left(\sum_i E_i\right)^2 - \left(\sum_i \vec{p}_i\right)^2, \text{ where } i \in \text{truth} \tag{3.1}$$

In the SM the Higgs decays would result in a sharp kinematic mass peak centred at the Higgs mass, as the decay width of the Higgs boson is expected to be only about 4.1 MeV. [34] In the MC simulation an artificially wide decay width of 400 GeV is chosen to create a flatter mass distribution.

The kinematic mass is hard to approximate from detector level information, since quite some information is missing. For instance, muons and neutrinos carry away energy that does not (completely) end up in the tracker or the calorimeter. Moreover, truth particles might not end up inside the large-R jet, reducing the mass of the jet. Yet it is the most informative variable as it can be directly related to the theory.

The true jet mass

It may be easier to choose a different target variable, like the *true jet mass*, denoted by m_{true}. This is the mass that is reconstructed and calculated from all the truth particles that end up inside the ungroomed large-R jet at detector level. This means that the energies and momenta as measured by the detector in the MC simulation are used to calculate the true jet mass:

$$m^2_{\text{true}} = \left(\sum_i E_i\right)^2 - \left(\sum_i \vec{p}_i\right)^2, \text{ where } i \in \text{truth} \cap \text{jet} \tag{3.2}$$

To simplify the problem even further, we also include the leptons that decay from the associated weak boson and end up in the large-R jet. This has the advantage the ML algorithms do not have to discriminate between situations in which the jet from the weak boson ends up in the large-R jet or not. This gives the model more space to learn other aspects that are needed to calculate the mass. This also reduces the need to use muon chamber information, which we will not be using. The disadvantage is that this affects the minimal achievable resolution of the Higgs peak. This approach would also reduce the necessity of using the object-based transformer network, which could use separate tracks as input. Still, this architecture could be beneficial if it were to compensate for e.g. pile-up effects.

Note that in this definition of the large-R jet, no grooming techniques have been applied, as this would remove some truth particles. Since all the truth-level information in the MC simulation is available, it is not needed to try and remove background particles.

The reco mass

Along with the latter definition, it is very useful to include the *reco mass* in the ML model. The reco mass is the mass reconstructed from the groomed UFO large-R jet as defined in Section 3.2.2. This mass, denoted as m_{UFO}, is not reconstructed at truth level and can therefore be used as an input variable. This variable may include background and besides that, some truth particles may be removed while creating the large-R jet. Moreover, the precision of the energy/momentum reconstruction of the individual particles is limited due to background particles. Hence this mass variable will be slightly different from the true jet mass, but at least it is closely related, making this variable very informative.

$$m^2_{\text{UFO}} = \left(\sum_i E_i\right)^2 - \left(\sum_i \vec{p}_i\right)^2 \text{, where } i \in \text{jet} \tag{3.3}$$

Target variable

In this project, we are aiming to sharpen the $H \to b\bar{b}$ mass peak in the SM signal sample by improving on the reco mass. The reco mass can be very useful for filtering signal from background events as the mass of a particle is a very characteristic property. Improving the resolution of the jet mass therefore ideally improves the classification performance and thus the signal strength.

When using ML models, this task is easier when training on a target variable that is closely related to m_{UFO}, since in that case using m_{UFO} alongside some other variables can be very beneficial. This method also allows us to more effectively use other variables that can be calculated from the large-R jets. The latter might not be as effective when training on the kinematic mass, which in itself is a complicated task as much information is missing. Yet, being able to predict the kinematic mass would be more informative as it is directly related to the theory. Both target variables will be investigated in this thesis.

3.3 Jet variables

The enormous amounts of low-level data collected in the ATLAS detector can be converted to high-level variables that describe the properties of the large-R jet. Some are designed to discriminate between different kinds of processes, which is useful for taggers. Sometimes taggers suffer from *mass sculpting*, which is any unphysical distortion in the mass distribution. This could for instance occur when jets have masses close to the mass of a certain particle, since then these events may be predicted to originate from that particle, irrespective of the actual origin of the jet. Consequently, background processes can be wrongly classified around that mass, which would result in deviating mass distributions. To prevent this effect, some variables have been defined that should be independent of the mass, which is undesired in our case. In this section, we will list and explain the available jet variables, after which we will test their usefulness and their relation to the mass. From this, a selection of variables will be made for each target variable.

First, the variables in the true jet mass sample will listed and discussed [1]:

- p_T
- η
- $|\eta|$
- m_{UFO}
- τ_1
- τ_2
- τ_3
- τ_4
- τ_{21}
- τ_{32}
- τ_{42}
- ECF_1
- ECF_2
- ECF_3
- C_1
- C_2
- D_2
- $\sqrt{d_{12}}$
- $\sqrt{d_{23}}$
- $\sqrt{d_{34}}$
- z_{12}
- z_{23}
- z_{34}
- $k_t\Delta R$
- τ_a
- P

The first four variables are relatively simple variables that are based on the constituents of large-R jets. The jet p_T represents the transverse momentum of the large-R jet, which is the sum of all the transverse momenta inside the groomed UFO large-R jet. The variable η is the pseudorapidity of the centre of the large-R jet and $|\eta|$ is its absolute value. The variable m_{UFO}, denoted by `mass` in the code, is again the reco mass.

[1] For the notation as used in the code see Table A.1.

3.3.1 Jet substructure variables

To increase the amount of data and insightful information on which the ML can learn, we could include more high-level variables called *jet substructure variables*. They can be computed using the lower level data as measured in the detector or slightly higher level data that is derived from the aforementioned reconstruction algorithms. We aim to include potentially more useful combinations of the available data. Some substructure variables turn out to be correlated with the true jet mass and hence they may be able to improve the predictions. In the rest of this section, the available substructure variables are briefly described.

N-subjettiness

The *N-subjettiness inclusive jet shape*, or τ_N, tries to define some sort of probability that a certain jet has N subjets. We know that more energetic particles are more likely to produce more jets and hence the values of τ_N for different N together may indicate how energetic the jet was and thus how heavy the decaying particle was. We define:

$$\tau_N = \frac{1}{d_0} \sum_i^K p_{T,i} \min\{\Delta R_{1,i}, \ldots, \Delta R_{i-1,i}, \Delta R_{i+1,i}, \ldots, \Delta R_{N,i}\} \tag{3.4}$$

Here d_0 is a normalisation factor defined by $d_0 = \sum_i p_{T,i} R_0$, where R_0 is the radius used in the jet clustering algorithm. The index i sums over all K particles, $\Delta R_{j,i}$ is again the rapidity-azimuthal distance between a candidate subjet j and the particle i. For $N = 1$ this subjet is just the large-R jet and for $N > 1$ the N hardest (highest p_T) VR anti-k_t subjets are used. If $\tau_N \approx 0$ then all the energy ended up in N or fewer subjets, but if $\tau_N \approx 0$ then there are at least $N+1$ subjets. [35] This means that if the average distance to $N+1$ subjets is smaller than to N subjets, then we would obtain $\tau_{N+1} < \tau_N$. This better match means that we are more likely to have $N+1$ rather than N subjets. Since the latter could be useful to know, combinations of these variables called the *N-subjettiness ratios* are defined:

$$\tau_{NM} = \frac{\tau_N}{\tau_M} \tag{3.5}$$

Small ratios would then indicate we are more likely to have N than M subjets. [36] If N or M is more than the number of reconstructed subjets, then $\tau_{NM} = -999$.

Energy Correlator Functions

Since the introduction of τ_N, more variables have been developed for the same purpose. For now, we will mainly focus on C_1, C_2 and D_2. For these variables, we first need to define the *N-point Energy Correlator Function* (ECF) that is suitable for hadron colliders [37, 38]:

$$ECF(N, \beta) = \sum_{i_1 < i_2 \cdots < i_N \in J} \left(\prod_{a=1}^{N} p_{T i_a} \right) \left(\prod_{b=a}^{N-1} \prod_{c=b+1}^{N} R_{i_b i_c} \right)^{\beta} \tag{3.6}$$

Here $R_{ij}^2 = (y_i - y_j)^2 + (\phi_i - \phi_j)^2$, where $y = \frac{1}{2} \ln \frac{E+p_z}{E-p_z}$. For relativistic particles $E \approx |\boldsymbol{p}|$ and hence $y \approx \eta$. The computation time grows exponentially with N and therefore only values up until $N = 3$ are taken into account. For $N = 1$ the right two products of equation 3.6 are equal to 1 and hence $ECF(1, \beta) = \sum_{i \in J} p_{Ti}$. After choosing β, as will be done below, this variable can be dropped to give the simplified notation ECF_N.

Using equation 3.6 one can define the *Energy Correlator Double Ratio*:

$$C_N^{(\beta)} = \frac{ECF(N+1,\beta)ECF(N-1,\beta)}{ECF(N,\beta)^2} \tag{3.7}$$

The parameter β is used to focus on a specific type of jet as it controls how important the wide-angle radiation in the outer regions of a jet is as compared to the collinear radiation in the centre. A value of $\beta = 1$ is used as it focuses on both regions making it the best discriminating value. Similarly to τ_N, C_N should be sensitive to jets with N subjets or N-pronged jets.

The substructure variable D_2 again tries to improve on C_2 and is defined using the normalised Energy Correlator Function:

$$e_N^{(\beta)} = \frac{ECF(N,\beta)}{ECF(1,\beta)^N} \tag{3.8}$$

This function can be used to define the dimensionless variable D_2 [39] as:

$$D_2 = D_2^{(\alpha=1,\beta=1)} = \frac{e_3^{(\alpha)}}{\left(e_2^{(\beta)}\right)^{3\alpha/\beta}} = \frac{e_3^{(1)}}{\left(e_2^{(1)}\right)^3} \tag{3.9}$$

Splitting Measures

The splitting measures, denoted by $\sqrt{d_{ij}}$, represent distances between two clustering objects, which are two subjets that are identified by the anti-k_t algorithm and then merged into one large-R jet. [40] These variables are defined using the *splitting scales*:

$$d_{ij} = \min(p_{Ti}^2, p_{Tj}^2) \cdot \Delta R_{ij}^2 / R^2 \tag{3.10}$$

where p_{Ti} and p_{Tj} are the transverse momentum of the subjets, ΔR_{ij} is the angular distance between the subjets and R is the radius of the large-R jet, which is equal to 1. For reclustered large-R jets with only one subjet, $\sqrt{d_{ij}}$ is equal to zero. The variable d_{12} represents the splitting scale of the last two remaining subjets and thus the hardest splitting. The variables d_{23} and d_{34} represent the splitting scales of the second and third last merged subjets respectively.

The splitting scales represent some angular distance. For high momenta, the opening angle between the two jets that (presumably) result from the Higgs boson can be approximated as $R_{12} \sim 2m_H/p_T$. If the reclustered subjets are indeed b-jets, then these splitting scales can be related to the Higgs mass up to some extent.

Using these splitting scales one can define new variables that are less correlated to the jet and subjet invariant masses, which could therefore be less useful in our case. These variables are called *energy sharing variables* and are defined by z_{ij} as:

$$z_{ij} = \frac{d_{ij}}{d_{ij} + m_{ij}^2} \tag{3.11}$$

where m_{ij} is the invariant mass of the two subjets combined. Hence apart from its dependence on the splitting scale, this variable is also directly related to this mass variable, which thus may be a strong discriminating variable.

KtDR

The variable KtDR, denoted by $k_t\Delta R$, is the angular distance between the last two subjets before reclustering [41], which is simply defined by:

$$(k_t\Delta R)^2 = R_{ij}^2 = (\eta_i - \eta_j)^2 + (\phi_i - \phi_j)^2 \tag{3.12}$$

The indices i and j represent the last two subjets. It aims to describe the jet radius for different kinds of processes and thus different distributions of the energy ending up in the jet. The actual radius of a jet is interesting as again it is related to the opening angle.

Angularity

The variable *Angularity* is denoted by τ_a (not to be confused with the N-subjettiness τ_N) and is defined by:

$$\tau_a = \frac{1}{M}\sum_i E_i \sin^a\theta_i[1-\cos\theta_i]^{1-a} \tag{3.13}$$

where M is the jet mass, E_i is the energy of the ith jet constituent and θ_i is its angle with respect to the centre of the jet. The parameter a can be chosen to emphasise radiation near the edges ($a < 0$) or core ($a > 0$) of the jet. The angularity variables are sensitive to the degree of symmetry in the energy flow inside a jet. Extremely boosted jets will exhibit a higher degree of symmetry as compared to a resolved jet in which the two b-quarks create two distinct subjets inside the large-R jet. The τ_{-2} observable is used as it helps to distinguish QCD jets from boosted particle decays, because of the broader tail expected in the QCD jets. [33]

Planar Flow

The last variable, Planar Flow, is denoted by P and is defined by:

$$P = 4\frac{\det(I_E)}{\mathrm{Tr}(I_E)^2} \tag{3.14}$$

where I_E is a two-dimensional matrix defined by:

$$I_E^{kl} = \sum_i \frac{1}{E_i} p_{i,k}p_{i,l} \tag{3.15}$$

Here E_i is the energy of the i^{th} jet constituent and $p_{i,k}$ and $p_{j,l}$ are the k and l components of its transverse momentum calculated w.r.t. the jet axis, i.e. $k, l \in \{x, y\}$. Collinear energy depositions like two-pronged decays result in $P \approx 0$, whereas isotropic energy distributions like many-body decays result in $P \approx 1$. [42]

3.3.2 Choice of variables

Even though it is possible to train an ML model with all the available variables, it may be more efficient and better to train the model with fewer parameters. In some parts of the ML model, the number of trainable parameters and thus the computation time scales quadratically with the number of input variables. Moreover, when you have too many useless variables, you may confuse your model in the sense that it has trouble finding out what combination of variables to use. Consequently, it may take longer to reach the optimal configuration, increasing the computation time even further. It may

even increase the chance that you end up in a local minimum if the input and thus the model is dominated by less useful variables. Hence if you could ignore some variables safely, this could speed up the optimisation of your model drastically and give better results.

Since one cannot simply test all combinations of input variables[2], you need to make more educated guesses on what variables should be used. One could run a training and then see what variables have been used by looking at the weights, such that you can remove the variables that turned out to be not so useful. However, whether a variable is useful depends on the size and architecture of your ML model and vice versa, the optimal architecture depends on the input variables.

Instead, one could make scatter plots of the target variable vs. the input variables. Any form of structure indicates that the variable could be used to derive the target value. Sometimes the structure is hard to spot, so additionally we plot the correlation between the input variable and the target. These correlation values can give a clear quantitative measure of how useful a certain variable can be, however, uncorrelated variables may still exhibit structures. Hence both methods will be used to decide which variables we should use.

For the scatter plots, we created a resampled dataset with a completely flat mass distribution. This is done to be able to distinguish the structure in the distribution of the target variable from the structure caused by the mass dependence. This flat mass sample is produced using undersampling and it only contains masses between 60 and 200 GeV. Despite the limited range, this will still give us valuable information.

In the scatter plots and correlations matrix, the true jet mass is used as the target variable. As can be seen in Figure 3.3, the reco mass, the ECFs and the splitting scales show clear structural behaviours. Also, p_T, the four subjettiness variables, C_1, C_2, $k_t\Delta R$ and the Angularity show slight structural behaviours.[3] In Figure 3.4 we can see that all their correlation values have values of more than 0.1, but so does τ_{21}. The ratio τ_{21} has a similar goal as τ_{32} and τ_{42}, but the latter two seem to exhibit no structure at all, hence it seems that all N-subjettiness ratios can be ignored relatively safely. The energy sharing variables, D_2 and Planar Flow do not seem to have have much structure or correlation w.r.t. the true mass. Hence they are probably not very informative and therefore they will be left out as well.

Even though $|\eta|$ is only a little correlated to the mass, it can still be informative since it is a variable related to θ, which is used for describing physical processes theoretically. Moreover, $|\eta|$ as well as η could be useful when combining this information with the track or subjet variable $d\eta$, which describes the pseudorapidity distance of the track or subjet w.r.t. the centre of the large-R jet. The angular distance can in some cases be related to the opening angle of the two decaying jets and thus the mass of the decaying particle. Therefore knowing the correct angle per track could be useful and hence η will be included as well.

All in all, this will leave us with the following list of input variables for the training on the true jet mass:

- p_T
- η
- $|\eta|$
- m_{UFO}
- τ_1
- τ_2
- τ_3
- τ_4
- ECF_1
- ECF_2
- ECF_3
- C_1
- C_2
- $\sqrt{d_{12}}$
- $\sqrt{d_{23}}$
- $\sqrt{d_{34}}$
- $K_t\Delta R$
- τ_a

[2] This scales as 2^N, where N is the number of input variables.
[3] The latter two behaviours are better visible when plotting the x-axis on a logarithmic scale.

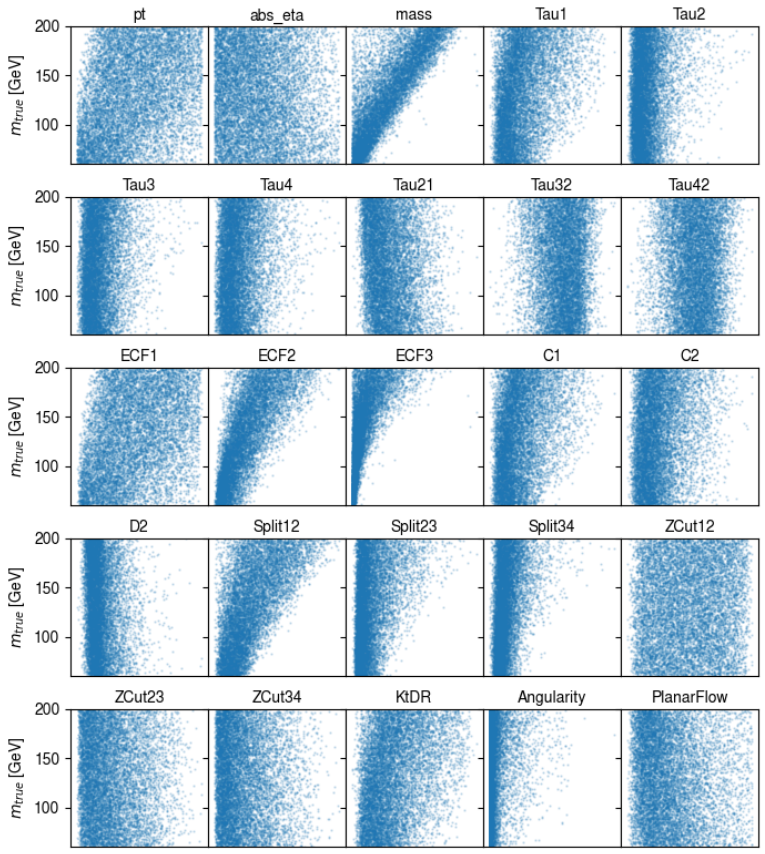


Figure 3.3: A scatter plot of the target mass vs. the different input variables, depicting their relationship.

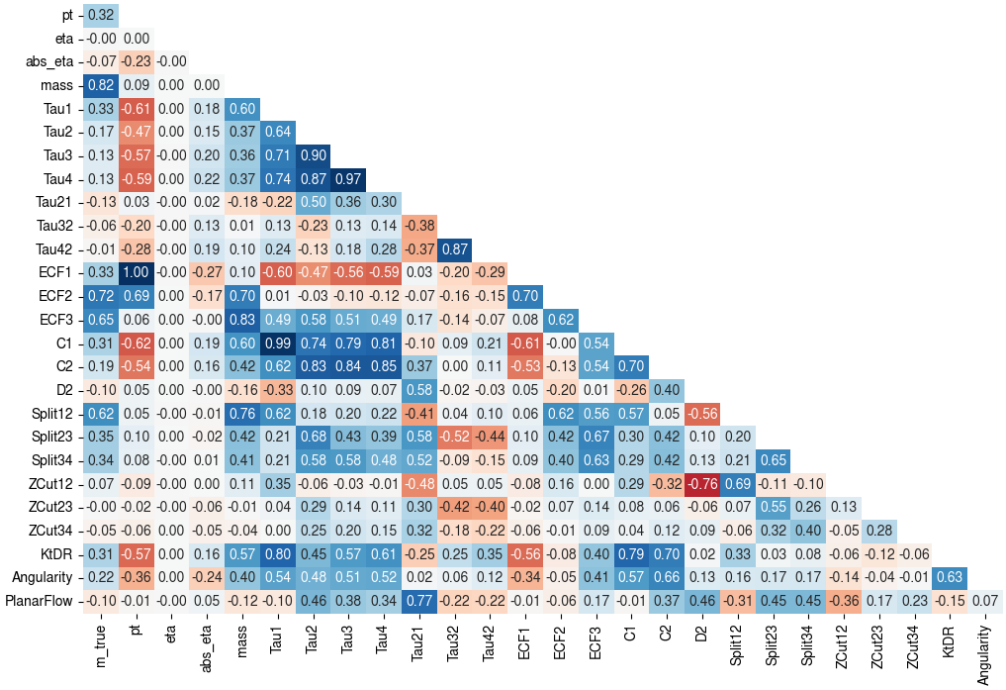


Figure 3.4: The correlation between the different input variables and the target mass.

For the training on the kinematic mass, we are not using the substructure variables. In a certain training on the kinematic mass, we found that including them worsened the results after sufficient epochs. That could be because they are defined using groomed UFO large-R jets and thus it is harder to relate them to the kinematic mass than to the true jet mass. Therefore, they are excluded and not further investigated in the training on the kinematic mass. Furthermore, $|\eta|$ is not available in the kinematic mass dataset, but the energy E is. The energy could perhaps be related to the mass and it indeed improved the results in a certain training, hence it will be included. All in all, in the training on the kinematic mass the jet variables p_T, η, E and m_{UFO} are used.

3.4 Track variables

Only tracks that are associated with the large-R jet are available. This means that some tracks originating from the Higgs decay that end up outside the large-R jet are not present in the data. This limits the maximal achievable resolution, mainly at low p_T. Furthermore, only a maximum of 100 tracks can be stored, but this number is only reached on rare occasions. In such cases, the tracks with the highest impact parameter are left out. Usually, about 20 tracks end up inside the jet. The kinematic mass sample allows for a maximum of 80 tracks per jet, but this would only alter 0.012% of the events.

All the variables that are available for the tracks are listed and explained in Table 3.3 and some are visualised in Figure 3.5. The first 11 out of 21 variables describe pixel-related information. This is very low-level information that could potentially outperform the high-level information as relatively few assumptions have been made by using reconstruction algorithms. However, it is much harder to convert these variables usefully to describe the jet mass as compared to using more physics-related variables like the latter 10 track variables.

When training on the true jet mass the pixel information did not show any improvements as the algorithm more heavily relies on the substructure variables that are better related

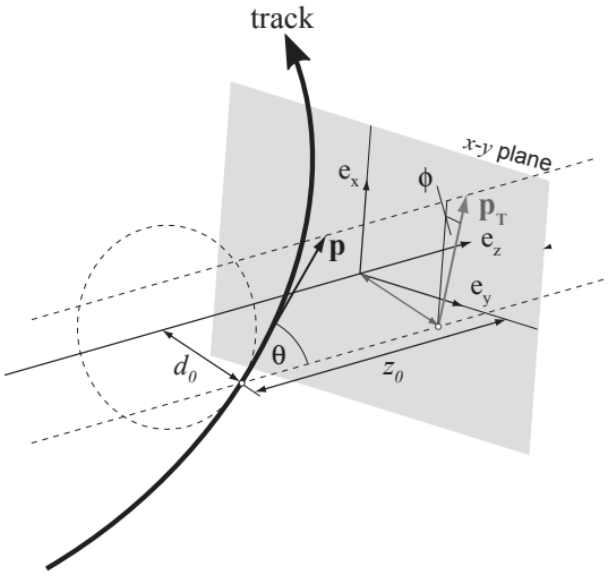


Figure 3.5: A visualisation of the track variables θ, ϕ, p_T, d_0 and z_0. [44]

Table 3.3: Track variables with their name, notation and description. The sign in the last two variables denotes the direction w.r.t. the beam axis. [43] The Insertable B-Layer (IBL) is the innermost pixel layer and the SemiConductor Tracker (SCT) surrounds the pixel detector. Split hits are hits identified as being created by multiple charged particles during the ambiguity solver stage at pattern recognition level. Shared hits are hits contributing to the track fit and to another track.

Name	**Description**
`numberOfInnermostPixelLayerHits`	Number of hits in the IBL $\in \{0, 1, 2\}$.
`numberOfNextToInnermostPixelLayerHits`	Number of hits in the next-to-innermost pixel layer $\in \{0, 1, 2\}$.
`numberOfInnermostPixelLayerSharedHits`	Number of shared hits in the IBL.
`numberOfInnermostPixelLayerSplitHits`	Number of split hits in the IBL.
`numberOfPixelHits`	Combined number of hits in the pixel layers (incl. IBL).
`numberOfPixelHoles`	Combined number of crossed active modules where no hit was found in the pixel layers (incl. IBL).
`numberOfPixelSharedHits`	Number of shared hits (not marked as split hit) in the pixel layers (incl. IBL).
`numberOfPixelSplitHits`	Number of split hits in the pixel layers (incl. IBL).
`numberOfSCTHits`	Combined number of hits in the SCT layers divided by two.
`numberOfSCTHoles`	Combined number of crossed active modules where no hit was found in the SCT layers.
`numberOfSCTSharedHits`	Number of shared hits in the SCT layers divided by two.

Name	**Notation**	**Description**
`qOverP`	q/p	Charge of the track particle divided by its momentum magnitude.
`deta`	$d\eta$	Pseudorapidity between the track and the jet.
`dphi`	$d\phi$	Azimuthal angle between the track and the jet.
`d0`	d_0	Transverse impact parameter (IP), closest transverse distance of the track to the primary vertex point.
`z0SinTheta`	$z_0 \sin(\theta)$	Longitudinal IP projected onto the direction perpendicular to the track.
`qOverPUncertainty`	$\sigma(q/p)$	Uncertainty on the track q/p.
`thetaUncertainty`	$\sigma(\theta)$	Uncertainty on the track θ.
`phiUncertainty`	$\sigma(\phi)$	Uncertainty on the track ϕ.
`IP3D_signed_d0_significance`	$s(d_0)$	Signed transverse IP significance $(d_0/\sigma(d_0))$ from the IP3D algorithm.
`IP3D_signed_z0_significance`	$s(z_0 \sin\theta)$	Signed longitudinal IP significance $(z_0/\sigma(z_0))$ from the IP3D algorithm.

to the target variable. A training excluding the substructure variables and including the pixel information showed a worse performance. Therefore the pixel information is left out when training on the true jet mass. When training on the kinematic mass the pixel information did show improvements, so they will be included in that case. As much research has been performed on which track variables to use, all of the latter 10 track variables will be included in both models.

As mentioned earlier, we would like to combine the jet and the track information to be able to use combinations of these variables as input of the transformer, like $\eta + d\eta$. These angular combinations could potentially be related to the opening angle and thus the Higgs mass. We will combine the jet and track variables by concatenating to each track all the jet variables. This will give us 28 variables for each track when training on the true jet mass and 25 variables when training on the kinematic mass. As mentioned earlier, in some parts of the transformer the number of trainable parameters and thus the computation time scales quadratically with the number of input variables. The reduction of 47 to 28 parameters thus cuts down the computation time to 35% in some parts, without losing much valuable information.

3.5 Subjet variables

All the variables that are available for the subjets are listed and explained in Table 3.4. How exactly these variables will be used is discussed in Section 3.7.7. Again the jet information will be concatenated to each subjet. Only events with at least two subjets are considered and at most three subjets can be stored per jet.

Table 3.4: Subjet variables with their name, notation and description.

Name	Notation	Description
`pt`	p_T	Transverse momentum of the subjet.
`eta`	η	Pseudorapidity of the subjet.
`mass`	mass	Invariant mass of the subjet.
`deta`	$d\eta$	Pseudorapidity between the subjet and the jet.
`dphi`	$d\phi$	Azimuthal angle between the subjet and the jet.

For both target variables, we found that adding subjets improved the predictions. For a model trained on the kinematic mass, the improvement in terms of RMSE was about 3% and it was mainly visible for low masses. When testing on the true jet mass the improvement was more significant, i.e. about 15%. The improvement was mainly visible for high masses, but not so much for masses around 125 GeV. The amount of improvement is of course dependent on the architecture of the model, but any improvement shows that the model is able to learn from the subjets. Therefore the subjets are added as input in both models.

3.6 Preprocessing

3.6.1 Resampling

The true jet mass data was originally produced to train taggers, i.e. classification algorithms. [15] To make sure that the taggers are less biased to choose one class over another, the different classes have been resampled such that each process occurs approximately equally often. The four different processes that have been used have characteristic distributions in certain kinematic variables. QCD events for instance

have a falling p_T distribution, causing relatively more events with low p_T values to be QCD events. Similarly, the $H \to b\bar{b}$ and $H \to c\bar{c}$ samples have a slight m_{UFO} peak at around 75 GeV. This may result in QCD events being classified as $H \to b\bar{b}$ or $H \to c\bar{c}$ too often when having a m_{UFO} value around the peak value, resulting in mass sculpting. To prevent this, the classes have been kinematically resampled as a function of jet p_T, η and m_{UFO}. The so-called 3D resampling requires the selection of kinematic bins with enough statistics for each process. Therefore only values of $250 < p_T < 1300$ GeV, $-2 < \eta < 2$ and $50 < m_{\text{UFO}} < 300$ GeV remain. This results in a slight reduction of statistics, but still 59 million events are left containing about 14.5 million $H \to b\bar{b}$, 14.5 million $H \to c\bar{c}$, 8 million top and 22 million QCD events.

The data containing the kinematic mass was not 3D resampled, as it was not produced for a tagger, but solely for this project. It contains values of $p_T > 150$ GeV, $-2 < \eta < 2$, $m_{\text{UFO}} > 50$ GeV and $50 < m_{\text{kin}} < 200$ GeV. The cut on m_{kin} has been performed to obtain an even flatter mass distribution as outside this interval we slowly run out of statistics. Values of $m_{\text{UFO}} < 50$ GeV were available, but this would include relatively many QCD events that are misidentified as $H \to b\bar{b}$ in the simulation. These are events in which the truth particles at detector level from QCD events mimic $H \to b\bar{b}$ large-R jets, causing these b-jets to be falsely identified as originating from a Higgs boson by the tagging algorithm. These events complicate the training and are therefore removed. Removing $m_{\text{UFO}} < 50$ GeV also makes the comparison to training on the true jet mass fairer. We chose a wider cut on p_T, partially to increase statistics and partially because at high momenta we hope to find signs beyond the SM.

All in all, the kinematic mass sample ends up with 3 million events. Since $H \to c\bar{c}$ events give comparable jets and have very similar kinematic distribution, these events are used to increase statistics. Including them indeed slightly improved the predictions on $H \to b\bar{b}$ events and naturally on $H \to c\bar{c}$ events. Together, the $H \to b\bar{b}$ and $H \to c\bar{c}$ datasets provide us with 6.8 million events. Top and QCD events are not considered when training on the kinematic mass.

3.6.2 Training, validation and test data

The data is split into three independent datasets: the training, validation and test datasets. The *training* data is used to train the ML algorithm and update its weights in each training step. The *validation* data is used to independently validate the training each epoch to for instance see if the ML algorithm is overtraining or training too fast or slow. Finally, the *test* data is used to independently test the predictions after the training. The training data must be big enough to gain the maximum achievable resolutions in the predictions. The validation data must be big enough to represent all the characteristics of the data, such that the loss is representative. The test data must be big enough to perform all the tests on how well the model was able to predict the target values.

Since 14.5 million $H \to b\bar{b}$ events provide a lot of statistics for the true jet mass model, only $H \to b\bar{b}$ events are used and relatively many events can be part of the training data. Therefore a splitting is applied resulting in about 12.3 million training, 1.2 million validation and 1 million test events. The kinematic mass dataset is split into about 4.5 million training, 1.1 million validation and 1.1 million test events.

3.6.3 Feature scaling

The used input variables, also called features, have vastly different ranges of values and different orders of magnitudes. In an ML algorithm, weights determine how important a

certain value is and how much it contributes when calculating the next variable. At the beginning of a training, the weights are not yet optimised, which means that variables with large values dominate in the calculations. Therefore they determine what variables the model is initially mainly looking at. Smaller though more important variables are only uncovered later in the training, which slows down the optimisation of the model, or perhaps not at all, causing the model to end up in a local minimum of the loss space.

To prevent these issues all the input variables are scaled. Since the variables could contain largely deviating values as well as small but important relative differences[4], the appropriate scaling scheme is *z-score normalisation* or *standardisation*. This feature scaling scheme per feature x is defined by:

$$x' = \frac{x - \mu}{\sigma} \tag{3.16}$$

Here μ is the mean value of the feature and σ its standard deviation. The same scaling is performed on the target values.

3.7 Machine Learning

Machine Learning algorithms can play an important role in improving on the reco mass, as they try to optimise the reconstruction of their target value. They do so completely differently as compared to conventional algorithms, hence potentially much better masses could be predicted. They could also deal with much more complicated events, like extremely boosted decays or events with pile-up from simultaneous collisions, causing particles to overlap in the detector. In future runs and detectors, dealing with the increasing luminosity is a challenging task. Here ML models could be very useful as well.

Nevertheless, they might also produce predictions with different and sometimes even unphysical properties. Hence it is important to understand the steps that ML models take and what their consequences are. This can also help to find the optimal model and improve the predictions. An ML model has many parameters that it can optimise on its own, but there are also parameters describing the ML model, called hyperparameters (HP), that need to be optimised by hand. In this section, all the necessary aspects of ML models are described as well as how the input data is used, after which the HPs are described.

3.7.1 Neural Network

Neural Networks (NN) form the basis of ML algorithms. The simplest neural network is called a Multilayer Perceptron (MLP), which consists of an input layer, at least one hidden layer and one output layer. Let us first start with a single *perceptron*, which can be defined by the function:

$$y = \phi\left(\sum_i w_i x_i + b\right) \tag{3.17}$$

In this function, the inputs x_i are multiplied by their corresponding weights w_i and added to a bias b. The activation function ϕ of a perceptron [45] is defined as:

$$\phi(x) = \begin{cases} 1 & \text{for } x > 0 \\ 0 & \text{else} \end{cases} \tag{3.18}$$

[4]This makes Min-Max scaling undesired.

If the activation function is linear, like $\phi(x) = x$, then adding the outputs of two perceptrons would effectively be the same as having one perceptron. Therefore, to increase the complexity and learning capacity of the NN the activation function must be nonlinear.

3.7.2 Activation function

Depending on the problem to be solved, one can choose from different activation functions. In classification problems you want the output to be some sort of probability denoted by values between 0 and 1, hence a sigmoid activation function could be used. However, in regression models, the output could in principle be any value.

A common activation function in regression models is the rectified linear unit or *ReLU activation function*, which is defined as:

$$\phi(x) = \begin{cases} x & \text{for } x > 0 \\ 0 & \text{else} \end{cases} \tag{3.19}$$

For $x < 0$ any combination of weights producing $x < 0$ will give the same output of $\phi(x)$. Finding the optimal set of weights in such cases is hard, because you need a gradient between two outputs corresponding to two sets of weights to be able to tell which set is better.

To help mitigate this so-called vanishing gradient problem, one might use the *Leaky ReLU activation function*, which is defined by:

$$\phi(x) = \begin{cases} x & \text{for } x > 0 \\ ax & \text{else} \end{cases} \tag{3.20}$$

Here $a = 0.01$, but it can in principle be any value. When using the Parametric ReLU or PReLU, the optimal value for a is learnt by the NN as well. [46]

A more complex activation function and a smooth approximation of ReLU is the sigmoid linear unit or *SiLU activation function*, which is defined by:

$$\phi(x) = x \cdot \sigma(x), \text{ where } \sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.21}$$

This function has the advantage that the switch of x going from positive to negative values or vice versa is smoother, which would result in more stable learning. This function is also used by the transformer-based GN2X tagger, which is trained on the same data as we are using. [47] Therefore it could be useful in our model as well. Many more smooth approximations of ReLU can be defined, but for now, we will only focus on this one.

3.7.3 Dense layer

When all inputs are used to calculate the output of a perceptron, you have a so-called *dense layer* of dimension one. You could use the same inputs to calculate different outputs using different perceptrons in parallel, which would create a higher dimensional dense layer. If you would represent each weight by a line connecting the input with the output of the activation function (both depicted by nodes), then you would obtain a so-called fully connected layer as depicted in Figure 3.6.

In this example, the two dense layers are hidden layers of an MLP. Each layer could in principle be made arbitrarily large and you could use an arbitrary number of dense

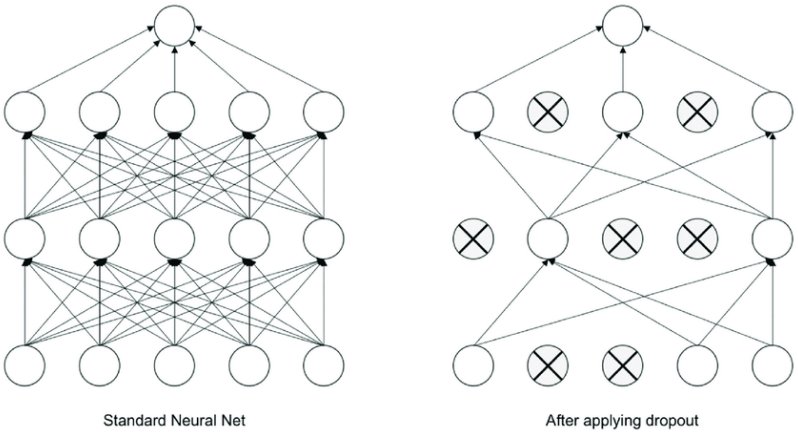


Figure 3.6: On the left, a schematic representation of dense layers as fully connected layers in a Multilayer Perceptron containing two hidden layers. Each line contains a weight and each node implicitly contains the summation and activation function, except for the input nodes that just represent the input. [48] On the right, a schematic representation of dropout in dense layers. [49]

layers. This increases the number of weights and therefore the complexity of the model, which ideally enhances its ability to predict the output. However, at some point, the number of weights is so large that in principle you can make any prediction and you could theoretically create a bijection of the input data to the target values of the training data. This would mean that your model is able to perfectly predict all target values provided in the training data, but not those in the independent validation and test data. This is called *overtraining*.

In order to prevent this problem without (much) reducing the number of weights and complexity of the model, one could use *dropout* regularisation. During every training step, a dropout layer randomly prunes away some of the nodes as depicted in Figure 3.6. This means that some of the nodes cannot be used anymore to calculate the value of the next node and therefore its value will be slightly different. This causes the predictions of an MLP to deviate from the target value even more, making it impossible to find the set of weights that matches just the training data and not the validation data. Instead, the model is forced to predict the target value for many possible configurations of the nodes, which effectively means that it needs to be able to predict the target value for different kinds of incomplete and noisy inputs. This helps *generalising* the model as it becomes able to make predictions in more situations. Therefore, dropout may help to get the model out of a local minimum.

A dropout rate of 0.2 randomly prunes away 20% of the nodes per layer during the training. It does so by temporarily setting these nodes to 0 while leaving the weights unchanged. However, during validation and testing, this is not done in order to achieve the best predictions. Since in that case, more nodes are used, which would result in larger perceptron outputs, the weights are temporarily adjusted accordingly. In this example, they would be multiplied by 0.8.

3.7.4 Metrics

In order to optimise the weights of a NN, a metric is needed to determine how well the model performed. This metric is called a loss function, which calculates some kind

of error between the predictions and the target values. Three loss functions will be addressed.

The *Mean Squared Error* (MSE) is the most common and well-known error function as it maximises the likelihood function. It is defined by:

$$\mathrm{MSE}(y,t) = \frac{1}{N}\sum_{i}^{N}(y_i - t_i)^2 \tag{3.22}$$

Here y_i denotes the predicted value, t_i the true or target value and N the number of data points. The advantage of this loss function is that it punishes predictions farther away from the true value relatively much as compared to values closer by. The disadvantage, however, is that it is scale-dependent. This means that for equal relative errors, the deviation at higher target values is punished harder than that at lower target values.

An alternative is the *Mean Absolute Error* (MAE), also known as the L1 loss function, which is defined as:

$$\mathrm{MAE}(y,t) = \frac{1}{N}\sum_{i}^{N}|y_i - t_i| \tag{3.23}$$

The advantage of this loss function is that it is not scale-dependent and it is better at ignoring bad predictions as a result of deviating input data, that should not be learnt from. This allows the model to focus more on learning one pattern belonging to the majority of the data, rather than learning two patterns less accurately. The disadvantage of this loss function is that it usually takes longer to converge to the optimal solution as compared to using the MSE.

The *Huber Loss function* tries to take the best of MSE and MAE by defining a piecewise function:

$$\mathrm{HuberLoss}_{\delta}(y,t) = \begin{cases} \frac{1}{N}\sum_{i}^{N}\frac{1}{2}(y_i - t_i)^2 & \text{if } |y_i - t_i| < \delta \\ \frac{1}{N}\sum_{i}^{N}(\delta|y_i - t_i| - \frac{1}{2}\delta^2) & \text{else} \end{cases} \tag{3.24}$$

By default $\delta = 1$, which makes this function equivalent to the Smooth L1 Loss. For $|y_i - t_i| > \delta$ this loss function behaves like the MAE and otherwise, it behaves like the MSE, which makes the function smooth around zero. The Huber Loss function should be able to ignore outliers like the MAE while still converging to the optimal solution quickly, especially during later epochs. [50]

One could also choose to optimise w.r.t. the *Root Mean Squared Error* (RMSE) as it has the correct unit, however, the optimal solution is the same for the MSE, while the convergence takes longer. Therefore using the MSE during training is more useful than using the RMSE. Nonetheless, testing the predictions after the training will be done with the RMSE to get the correct unit.

For an unbiased estimator, where $\bar{y} = \bar{t}$ in each bin, the RMSE equals its standard deviation σ, which describes the dispersion of a distribution. In a normal distribution σ describes the resolution of the peak. The standard deviation is calculated w.r.t. to the mean of the distribution.

$$\sigma(y) = \frac{1}{N}\sum_{i}^{N}(y_i - \bar{y})^2 \tag{3.25}$$

Therefore it is in general not equal to the RMSE. When approximating the distributions of the different mass variables as normal distributions, one could try to improve on resolution of the reco mass by minimising σ. However, this allows for a deviating mean,

i.e. a bias, which you want to minimise as well. The RMSE minimises both the bias and variance at once and is therefore used to determine how well the model is able to predict the target value after the training. When determining the resolution of the signal in the SM samples, the mean and standard deviation are investigated as well.

3.7.5 Weight optimisation

The loss function defines a loss space that is dependent on the weights. Starting at some initial position denoted by a set of weights, we would like to update the weights such that we end up in the global minimum of the loss space. These updates (also named training steps) are performed during so-called *epochs*, during which the entire dataset is used once to update the weights. After each epoch parameters like the learning rate may be updated to improve the optimisation.

Multiple weight optimisation schedules have been developed to optimally optimise the weights. The most frequently used optimiser is the Adam optimiser, which is based on multiple simpler optimisers. These are Gradient Descent, Momentum and RMS Propagation. When they have been elaborated, the Adam optimiser as well as a slightly more complex variant called the AdamW optimiser will be described. The latter will be used in our model.

Gradient Descent

The simplest optimiser is called Gradient Descent (GD), which updates the weights as:

$$w^{\tau+1} = w^{\tau} - \eta \nabla L(w) \tag{3.26}$$

Here w is a certain weight vector, τ the timestamp, $L(w)$ the loss function given the weights and $\eta > 0$ is called the *learning rate*. The learning rate (LR) can be tuned to achieve efficient weight updates and optimal performance. A small LR can give more accurate predictions, but takes longer to converge, whereas a large LR is faster, but it may diverge from a solution as depicted in Figure 3.7.

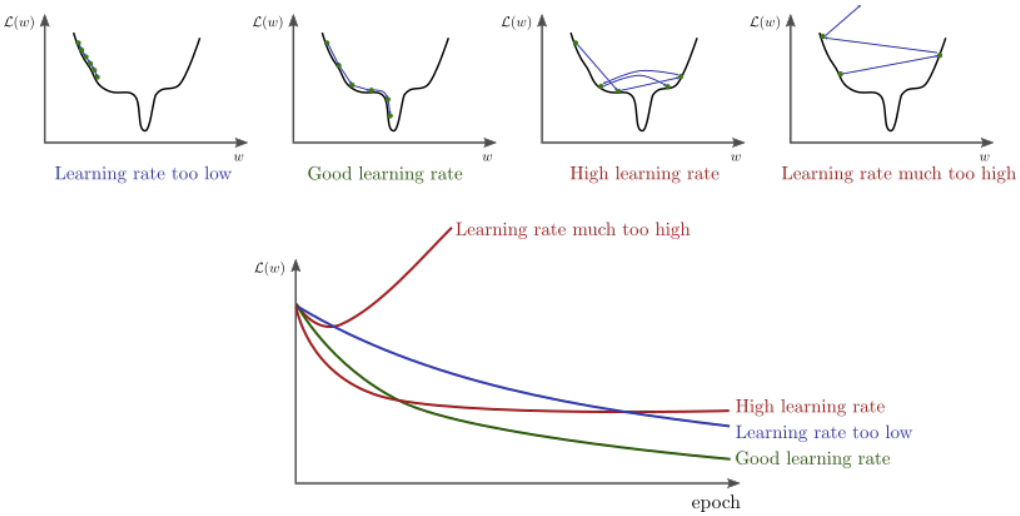


Figure 3.7: A visualisation of different learning rates and their consequences. [51]

Mini-batch learning

The gradient $\nabla L(w)$ could be calculated using the entire training dataset, but this means that updating the weights once is computationally very expensive. Alternatively, you could update the weights after each data point, which is called Stochastic Gradient Descent (SGD), resulting in quick learning. However, the update would be very inaccurate and different data points could even update the weights in opposite directions. Therefore it is desired to use the best of both worlds: *mini-batch learning*. Updating the weights each timestamp is done using parts of the data called a *batch*. Ideally, this batch is big enough to give representative and accurate updates, but small enough for quick convergence. The batch size is again a tunable hyperparameter of the model.

Momentum

Momentum can be used to stabilise and speed up the training. It is defined using a moving average of gradients from multiple steps. Using this gradient to update the weights reduces the randomness in the gradients of different batches and it forces the weights to continue updating in the same direction. Moreover, Momentum can help the model get "pulled" out of a local minimum. The momentum is defined as:

$$V^\tau = \beta V^{\tau-1} + \eta \nabla L(w) \tag{3.27}$$

Here V is the momentum and $\beta \in (0, 1)$ is a new hyperparameter that determines how much previous momenta should be used and thus the strength of the momentum. Updating the weights is then defined by:

$$w^{\tau+1} = w^\tau - V^\tau \tag{3.28}$$

AdaGrad

Rather than taking a constant LR, a different LR per weight could be set using AdaGrad. Some weights do not get updated frequently, because the type of data points that should do so is not very common. Other weights do so, because their effect on the gradient is little, resulting in small updates. AdaGrad tries to compensate for that, which boosts the convergence speed.

Root Mean Square Propagation

RMSprop improves on AdaGrad by not only looking at the last time step but by also looking at previous time steps, similar to Momentum. The weights are then updated according to:

$$w^{\tau+1} = w^\tau - \frac{\eta}{\sqrt{v^\tau + \epsilon}} \nabla L(w) \tag{3.29}$$

Here ϵ is a smoothing term to avoid division by zero (10^{-8} by default) and v^τ is defined as:

$$v^\tau = \beta v^{\tau-1} + (1 - \beta)(\nabla L(w))^2 \tag{3.30}$$

Adam

Finally, the Adaptive Moment Estimation (Adam) optimiser is a combination of mini-batch learning, RMSprop and Momentum. It first calculates two moving averages m^τ and v^τ each time stamp, which are defined as:

$$m^\tau = \beta_1 m^{\tau-1} + (1 - \beta_1) \nabla L(w) \tag{3.31}$$

$$v^\tau = \beta_2 v^{\tau-1} + (1 - \beta_2)(\nabla L(w))^2 \tag{3.32}$$

Here β_1 and β_2 are hyperparameters, which are by default 0.9 and 0.999 respectively. Since these averages are biased to 0 for $\beta_1, \beta_2 \approx 1$, bias-corrected variants are defined:

$$\hat{m}^\tau = \frac{m^\tau}{1-(\beta_1)^\tau} \tag{3.33}$$

$$\hat{v}^\tau = \frac{v^\tau}{1-(\beta_2)^\tau} \tag{3.34}$$

Here the τ in the denominators denotes an exponent, which equals the time step. [12] The weights are then updated by:

$$w^{\tau+1} = w^\tau - \eta \left(\frac{\hat{m}^\tau}{\sqrt{\hat{v}^\tau + \epsilon}} - \lambda w \right) \tag{3.35}$$

The term $-\lambda w$ is added in the AdamW optimiser, where $\lambda = 0.01$ by default. It outperforms Adam joint with L2 regularisation, but apart from that it does a similar job. [52] L2 regularisation is an example of Weight Decay that prevents overfitting by penalising big weights. It does so by adding the term $-\lambda||w||^2$ to the loss function, but this affects the loss space and its minimum. In the AdamW optimiser Weight Decay is decoupled from the loss function. It is only applied when updating the weights using the derivative of w^2 w.r.t. w (the derivative of the L2 loss term) as an alternative to the L2 regularisation.

Backpropagation

The weights cannot be updated all at once. Instead, backpropagation is applied in the optimiser, which means that updating the weights starts at the end of the model and ends at the beginning. To illustrate this, a simple feed-forward network is considered:

$$a_j = \sum_i w_{j,i} z_i \tag{3.36}$$

$$z_j = \phi(a_j) \tag{3.37}$$

Here ϕ is an activation function, z_i is called an activation and $w_{i,j}$ is the i^{th} weight of the j^{th} layer. For a given set of weights, all activations can be calculated starting from the beginning of the model. This is called forward propagation.

The loss belonging to a specific weight can then be calculated using the chain rule:

$$\frac{\partial L}{\partial w_{j,i}} = \frac{\partial L}{\partial a_i} \frac{\partial a_i}{\partial w_{j,i}} = \delta_j z_i \tag{3.38}$$

The first partial derivative is named δ_j and it is known in the last layer when for instance the MSE loss function is used:

$$\delta_j = \frac{\partial L}{\partial a_i} = \frac{\partial L}{\partial z_i} \frac{\partial z_i}{\partial a_i} = (y_k - t_k)\phi'(a_j) \tag{3.39}$$

Here $\phi'(a_j)$ is the derivative of the action function. This equation can then be used to work backwards through the model to update the other weights using equation 3.38 and the δ of the next layer [53]:

$$\delta_j = \frac{\partial L}{\partial z_j}\phi'(a_j) = \left(\sum_k \frac{\partial L}{\partial a_k} \frac{\partial a_k}{\partial z_j} \right) \phi'(a_j) = \left(\sum_k \delta_k w_{j,k} \right) \phi'(a_j) \tag{3.40}$$

Learning rate scheduler

The learning rate η, as mentioned before, can be optimised to obtain the desired performance and convergence time. Instead of assigning one value to it, this value could be adjusted to match the desired value as a function of the epoch. In our model, we apply the *OneCycleLR scheduler*. [54] At the end of the training, when the model is close to a minimum loss value, a smaller LR is desired to allow for a closer approach to the minimum loss value. In the beginning, it is useful to have a rather big learning rate to quickly move relatively close to the minimum. However, when going too quickly at the beginning, the minimum might already be traversed. Hence the OneCycleLR scheduler first starts at some initial LR and increases to its maximum LR within for instance 10% of the number of epochs. Then during the rest of the epochs, it decreases again to its final LR. It does so in both intervals via a cosine-shaped curve.

3.7.6 Attention mechanisms

A key element of the architecture that we will be using is the self-attention mechanism. Attention mechanisms deduce the relations between the elements in a sequence, after which it determines how much and what information of each element is used or worth paying attention to.

Input data

The input that we will be using to explain the attention mechanisms is defined as $x \in \mathbb{R}^{L\times d}$, which is a sequence of length L, consisting of embeddings of dimension d. The input could for example be the track data belonging to an event, containing L tracks and d features. Instead of using these specific features, the embeddings could also consist of a certain number of combinations of these features created by dense layers. The row vector $x_i \in \mathbb{R}^{1\times d}$ denotes the i^{th} element in the sequence, e.g. a single track.

Attention mechanism

In an attention mechanism, the input x is used three times. To see which (and how much of these) elements must be used to produce the output, a set of weights will be defined. These weights are called the attention scores. To calculate the attention scores an $L \times L$ matrix is produced, consisting of the elements $z_{ij} = x_i x_j^T$. The i-th row is defined by the vector $\mathbf{z} = [x_i x_j^T]_{j\in[1,L]}$. These row vectors are put into the softmax function:

$$\text{softmax}(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{L} e^{z_k}} \tag{3.41}$$

This softmax function gives us normalised weights, which sum up to one per row. These weights can be put into a matrix form giving us the attention scores:

$$a_{ij} = \text{softmax}(x_i x_j^T) = \text{softmax}([x_i x_k^T]_{k\in[1,L]})_j = \frac{e^{x_i x_j^T}}{\sum_{k=1}^{L} e^{x_i x_k^T}} \tag{3.42}$$

This $L \times L$ matrix is then multiplied again by the input to give for each track i the vector:

$$A_i = \sum_{j=1}^{L} a_{ij} x_j \tag{3.43}$$

This new vector is the so-called context-aware vector, which can be put into a neural network to give the desired output. [55]

Self-attention mechanism

In our algorithm, however, we will be making use of a self-attention mechanism, which is slightly more involved than the previously defined attention mechanism. The difference lies in the way the input is used. Instead of simply using the same input x three times, we will use three different kinds of inputs called the query $Q = xW^Q$, the key $K = xW^K$ and the value $V = xW^V$. Effectively one passes the input through three separate Linear layers. In each kind, the features are multiplied by a weight matrix to create certain combinations of these features that aim to fulfil their tasks, which are explained below.

Query: The query is a vector that describes what part of the input we need to pay attention to. [56] So the i-th track-like vector $Q_i = x_i W^Q$ is a weighted combination of the features of interest. The dimension of the query is defined as d_q, hence $Q \in \mathbb{R}^{L \times d_q}$ and $W^Q \in \mathbb{R}^{d \times d_q}$.

Keys: For each query, we have a key which is a vector of dimension d_k. This vector tells us which combination of features we must pay attention to given the query, such that we can determine weights that will select the tracks we will need for the output. In other words, it tells us what the key features are that relate the important information of the query to the other elements of the sequence. Since the key and query will be multiplied $d_k = d_q$. Again $K \in \mathbb{R}^{L \times d_k}$ and $W^K \in \mathbb{R}^{d \times d_k}$.

Values: For each track, we produce a vector of dimension d_v consisting of the combinations of features that will be passed on to the rest of the model to produce the desired output. Now $V \in \mathbb{R}^{L \times d_v}$ and $W^V \in \mathbb{R}^{d \times d_v}$.

Just as before we will compute an attention score, but this time by using the formula:

$$a_{ij} = \text{softmax}\left(\frac{Q_i K_j^T}{\sqrt{d_k}}\right) \tag{3.44}$$

To prevent small gradients at large d_k, the input of the softmax function is divided by $\sqrt{d_k}$. [57] Multiplying the attention scores with the values then gives us a matrix $A \in \mathbb{R}^{L \times d_v}$ containing the most useful tracks containing the most useful combination of features. These operations are depicted in Figure 3.7.6 and can be summarised by:

$$A(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V \tag{3.45}$$

Using these three different kinds of inputs allows us to extract more useful and more thoroughly selected tracks and features as compared to a regular attention mechanism, as more accurate attention scores and context-aware vectors can be created.

We can summarise the process as follows: The query determines what combinations of features we must attend to. From this, the key determines how the features of the track are related to the other tracks by finding the key features. The resulting attention scores create tracks that are aware (and combinations) of the other tracks. The value provides the most useful combination of features that will then be passed on to the model to produce the output. The output could thus be a list of (generated) tracks that are likely to correspond to the decay, each containing the required or corrected information that is needed to calculate the mass.

Multi-Head Attention

A more advanced version of the self-attention mechanism is the Multi-Head Attention (MHA) mechanism. An MHA layer runs through a self-attention mechanism several

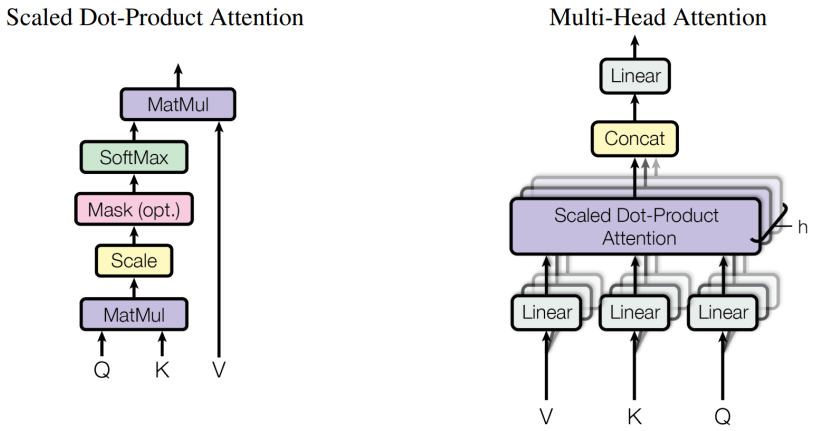


Figure 3.8: On the left, a Scaled Dot-Product Attention layer with three different inputs, i.e. query, key and value. On the right, a Multi-Head Attention layer, consisting of several attention layers running in parallel. It has three inputs, which could be the same or different objects, that are linearly transformed to form the queries, keys and values per head. [57]

times in parallel. The independent outputs are concatenated onto one array and then linearly transformed into the required dimension. [58] Multiple attention heads allow for different outputs by attending differently to the same tracks. Each head may for instance focus on extracting information from different subjets or cones, if that is the path the model takes.

The *number of heads*, denoted by h, determines how many times the algorithm should perform the attention mechanism. The result of the MHA then is:

$$MHA(Q,K,V) = [H_1,\dots,H_h]W_0, \text{ where } H_i = A(xW_i^Q, xW_i^K, xW_i^V) \tag{3.46}$$

Here $i = 1,\dots,h$. The input x is again linearly transformed, but now differently for each head using the trainable weight matrices $W_i^Q \in \mathbb{R}^{d\times d_k}$, $W_i^K \in \mathbb{R}^{d\times d_k}$, $W_i^V \in \mathbb{R}^{d\times d_v}$. This will result in different queries, keys and values per head. One could create three different input objects with different dimensions d by having three separate linear projections or dense layers before the MHA. But since x is already linearly transformed in each head, this may not be really necessary and therefore this is not done. The last linear transformation $W^O \in \mathbb{R}^{hd_v\times d_{\text{out}}}$ reshapes the output of the different heads of the MHA into the desired dimension, but this step is optional. [56] The steps of a MHA are depicted in Figure 3.8.

MHA in series

Besides using multiple heads in parallel, one could create a deeper neural network by placing MHA layers in series. This allows the ML algorithm to learn to more accurately filter the tracks by first creating an MHA that only removes a part of the tracks using some looser selection criterion. This may of course remove tracks that are needed for the next selection criterion, hence the track information x is passed along with the MHA to the next layer via $x' = x + MHA(x)$.

To make sure that the numbers do not grow exponentially large after a few layers, which causes the activation function to operate differently, a normalisation layer is applied. This is called Layer Normalisation, which is defined by:

$$\text{LayerNorm}[x] = \frac{x - \mathbb{E}[x]}{\sqrt{\text{Var}[x] + \epsilon}} \cdot \gamma + \beta \tag{3.47}$$

By default $\epsilon = 10^{-5}$, which prevents negative roots due to numerical deviations. Here β and γ are trainable parameters. [59] Layer Normalisation is applied as defined by Normformer. [60] This prevents the gradients of earlier layers from being much bigger than later layers, which adds stability to the training. The input and output of the MHA as well as x' will go through this Layer Normalisation, such that we can define a self-attention layer by:

$$y[x] = \text{LayerNorm}[x + \text{LayerNorm}[MHA(\text{LayerNorm}[x])]] \tag{3.48}$$

The number of self-attention layers is again a hyperparameter. The combination of self-attention layers with input and output layers together creates an ML model called a transformer.

3.7.7 Hyperparameter optimisation

There are quite some free parameters in the MHA layer that need to be fine-tuned to optimise the output. Finding the optimal parameters for an ML model is partially a matter of trial and error, but we can at least make a well-educated guess as a starting position. Some parameters can be set by simplifying the problem a bit and others should preferably be set as big as the model allows it to be. In this section, some simplifications and a choice of architecture will be made, after which the leftover HPs are listed. The effects of the leftover HPs are described, investigated and set in the results.

Problem simplification

To prevent information loss we could choose the embedding dimension of the input of the MHA d to be at least equal to the number of features f present in the tracks. Two methods to choose d are described below:

1. Each head will have the same dimension as the track dimension, such that the MHA can focus on different parts of the sequence without information loss, i.e. $d_k = d/h = f$. In that case, we can choose an arbitrary number of heads if we define the embedding dimension to be $d = h \cdot f$.
2. The input will be divided among the different heads and hence $d_k = d/h \geq f/h$, meaning that d must be a multiple of h. To make sure that all information is used, the embedding dimension must potentially be increased to the next nearest multiple of h. This way of splitting speeds up the optimisation, without losing information, however, this reduces the complexity of each head.

The second option is more conventional as a transformer is designed for language models, which usually deal with data that is described with large embedding dimensions. Therefore splitting this embedding dimension across different heads allows for looking at a sentence differently, without having to reduce the embedding dimension to acquire a reasonable computation time. Since the embedding dimension in our case does not necessarily need to be that large, the first option will be used as it gives more freedom in choosing d and h. If desired d can always be set larger than $d = h \cdot f$.

Again to prevent information loss, we set $d_v = d_k$, such that $d_q = d_k = d_v$. In that way, it is possible to interpret the output of the MHA as a selection of track-like objects with track-aware or corrected information.

Choice of architecture

As mentioned before, to be able to use both the jet and track information (as well as combinations of them) inside the MHA, the jet info is concatenated to each of its

associated tracks. Then this information will be transformed to a certain embedding to allow for these combinations using one or more dense layers. The embedding dimension as well as the number of hidden dense layers and their dimensions are HPs of the model.

Since we are including subjets, it would be useful to be able to use both the track and subjets information alongside the jet information inside the MHA layers. This can be achieved by first concatenating the jet and subjet information and then transforming the jet+subjet objects via a dense layer to the same dimension as the embedding dimension of the jet+track objects. Then the n embedded tracks can be appended to the 3 embedded subjets to obtain a $(n+3) \times d$ dimensional object describing an event. This embedded object can then be used as a query, key and value in the MHA layer.

The embedded object will then pass n self-attention layers each consisting of the same number of heads h. The concatenated output may then be transformed to the same embedding of the input of the self-attention layers. Afterwards, again a certain number of hidden layers can be used to predict the target value, using a dense layer of dimension one.

Hyperparameters

To clearly state what tests have been performed to optimise the two ML models a list of HPs and options is given, which are tested and optimised in the results.

- The effect of certain cuts on the data.
- The loss function.
- The activation function.
- The dropout rate in the different dense layers.
- The learning rate.
- The number of epochs.
- The batch size.
- The number of hidden layers and their dimension before the MHA.
- The embedding dimension d.
- The number of heads h.
- The number of self-attention layers n.
- The usage of the output projection.
- The number of hidden layers and their dimension after the MHA.

Chapter 4

Results

In this chapter, we present and argue the optimal models belonging to the two different target masses, i.e. the true jet mass and the kinematic mass. After that, their predictive abilities are investigated. This will first be done using the test samples, after which the two models will be compared side by side using the SM sample.

4.1 Optimisation

In this section the dependence of the predictions on the architecture will be discussed, allowing us to choose the HPs of the two models. First, some results and choices are described that are independent of (or similar for) the two target masses. After that, architectures are investigated and chosen separately for the two targets.

4.1.1 General architecture settings

Hidden layers

Even though hidden dense layers are commonly used in ML algorithms, they did not show any benefits. When training on relatively few epochs, any bias that showed up was enlarged by hidden layers. When training for many epochs the model with hidden layers seemed to converge to give the same results as the same model without the hidden layers. Therefore it seems as if the only dense layer transforming the input data to the embedding dimension before the MHA is sufficient. Possibly, the input features themselves are already very informative, or at least for the MHA, meaning that many dense layers are unnecessary. Similarly, any hidden layer after the MHA seemed to degrade the results as well, implying that the MHA layer performs the most important data transformation.

Multiple architectures for the hidden layers have been tested. For instance, the encoder architecture in which the output of the MHA is pushed through dense layers having a dimension that is half the dimension of the previous layer. This results in a triangle-shaped architecture. Different numbers of layers have been tested, including the case in which the dimension is halved until it has dimension 1. This is then the output layer predicting the mass. Also, dense layers having a dimension of at least twice the embedding dimension of the MHA have been considered. In some cases the model was even unable to converge, indicating that the maximum number of trainable parameters has been overshooted. This means that no gain could be obtained from increasing the

size and number of the dense layers even further. Furthermore, applying the optional linear transformation on the output of the MHA to obtain a lower-dimensional track-like object after the MHA turned out to be disadvantageous in terms of RMSE.

Taking these observations into account, we choose a model in which the input is pushed through one dense layer to the embedding dimension of the MHA. The output of the MHA is then transformed via one dense layer to the output layer of dimension 1, meaning that this last layer is like a single perceptron.

The activation function

Changing the activation did not have any noticeable effect on the quality of the predictions. Even the loss curve of the training and validation sets showed no difference, not even at the start of the training. This could be because we are not using a deep NN with many dense layers, which are the only layers using an activation function. Therefore the effect of using different activation functions is negligible, which is why we choose to use the activation function as applied in the GN2X tagger, i.e. the SiLU activation function.

Dropout

As mentioned earlier, dropout can help a model generalise, which prevents it from ending up in a local minimum. It does so by pruning away certain nodes, which may contain valuable information. Therefore, the model effectively trains on data with a slightly lower quality, which reduces the quality of the predictions on the training data. However, if the model can generalise well, this could mean that the model ended up at a lower minimum, which could compensate for the reduced quality of the data.

The latter is probably not what happened in our case. Any addition of dropout worsened the predictions in terms of RMSE. This may also indicate that the model does not contain enough parameters, such that it can not learn to generalise. Increasing the number of parameters naturally improves the predictions, but at some point, dropout did not seem to have any effect anymore, even when using a dropout rate of 0.5. This could mean that multiple nodes acquired the same goal, hence if some nodes are pruned away the results remain similar.

Alternatively, it could mean that the model is already able to generalise well, such that adding dropout is not able to improve the model. This could well be the case when training on the true jet mass as the training data contains 12.3M events. Hence the different patterns could already be recognisable and learned. The increase from 1M to 12.3M showed only little improvement. Therefore, we expect that not many more different kinds of patterns can be learnt and that using this setup there is no way the model can generalise even further. In that case, choosing a dropout rate of 0 allows for better predictions.

Learning rate

The initial and final learning rates in the OneCycleLR scheduler are set to be 10^{-8}. This allows for a smooth start and smooth settling at the end. The LR is set to reach its maximum after 10% of the epochs. The optimal maximum learning rate for both target variables seemed to lie around 10^{-4}.

Any value higher than 10^{-4} caused to model to learn quickly indeed, after which barely any progress is observed. This is visible in the validation loss as a near vertical line downward followed by a near horizontal line. Such a loss curve prevents the model from

accurately learning a pattern and instead, the loss oscillates around the minimum for quite a while. This prevents the model from optimally using the lower LR during later epochs (for learning subtle patterns) as Momentum is not able to direct the model to a minimum very well.

Any value lower than 10^{-4} caused the loss curve to be smoother, however, even after 50 epochs the loss was still higher than the model with a max LR of 10^{-4}. This means that the model did not have enough time to approach the minimum of the valley before the OneCycleLR scheduler decreased the LR again.

Loss function

It is to be expected that using MSE as the loss function would minimise the RMSE, yet other loss functions may produce predictions that are more desired. As mentioned before the MSE loss function is scale-dependent, meaning that the relative error at high target values contributes more to the loss than similar relative errors at low target values.

When applying the MAE loss function then as expected the predictions at low target values become better and worse at high target values. This is interesting for W and Z boson calibration studies as in that case a good resolution is desired around a mass of 80 and 90 GeV.

In the scope of this project, using the MAE was mainly interesting as it is better at neglecting outliers. Such an outlier could be a situation in which the reco mass is much higher, because it accidentally used a background particle to reconstruct the jet mass. Alternatively, the reco mass could be much lower, because some truth particles ended up outside the large-R jet. This would also mean that fewer tracks are associated with the jet and thus available as input. Similarly, a neutrino could result in a reduced reco mass and so do misidentified QCD events. This is what results in a tail in the m_{UFO} distribution, especially for low m_{UFO}. At high values for m_{UFO} the tail is somewhat compensated by the grooming algorithms, which aim to remove background particles.

Such outliers, however, are very common, meaning that it is more efficient to be able to compensate for these outliers as well, rather than focusing on improving on the m_{UFO} masses that are already predicted relatively well. Moreover, using the MAE does not show a significant difference in the distribution of the predictions, indicating that outliers are not suddenly clearly ignored. Therefore these "outliers" should be improved on as well and thus not be considered as outliers. In that case, using the MSE loss function is more desirable as it better takes into account different kinds of events and it better minimises the RMSE. The latter is indeed observed, therefore the HuberLoss and MAE are not used in the rest of the results.

4.1.2 Optimising for the true jet mass

In this section, we will optimise the model trained exclusively on the true jet mass. At the end, a summary of the model will be presented.

Batch size & number of epochs

Since the true jet mass dataset contains 14.5 million events, it is desired to use relatively large batches as otherwise, the model has already reached its minimum before the entire dataset could have been used. Preferably a dataset is used multiple times to update the weights. Using large batches would also give the most accurate updates on the weights.

The batch size is limited by the available memory. In a previous version of the code, it was only possible to load the entire dataset at once. This resulted in an even lower limit on the batch size, the number of events or tracks that could be used, as well as on the model size and thus its complexity. This motivated the use of the package Salt [61] as it uses data loading procedures from Pytorch. It describes the entire dataset using pointers and it only loads the data into the memory when a certain batch is called. In that case, only the batch size in combination with the model size is limited by the available memory. Of course, the optimal value of an HP is dependent on the values of the other HPs. This means that the HPs, including the batch size, should be optimised altogether, for instance via a grid search.

To obtain a reasonable computation time and precision on the weight updates a batch size of 1000 is applied. This value also allows for the investigation of relatively big models. Using 10 epochs in that case already allowed the model to converge with a smooth validation loss curve, but to obtain better results the final model was trained with 50 epochs, which resulted in slightly better results.

MHA hyperparameters

Increasing the number of self-attention layers n, the number of heads h and the embedding dimension d naturally improves the predictions. That is until overtraining occurs, but when using 14.5 million events this was not a problem. The limiting factor again is the available memory, which only allows for a limited number of combinations for n, h and d. Every increase of a factor two of these variables improves the predictions differently. Per variable, every next multiplication of two gives a smaller improvement than the last one.

Increasing the embedding dimension did not improve the predictions as much as increasing the other parameters. Therefore $d = h \cdot f_{jt}$ will be chosen as the embedding dimension, where f_{jt} is the number of features per track+jet object as they contain more variables than the subjet+jet objects. Here f_{jt} is equal to the number of track features f_t plus the number of jet features f_j, i.e. $f_{jt} = f_j + f_t = 28$.

When training on the true jet mass it turned out that increasing n had more effect than increasing h. At some point increasing n did not have much effect anymore, after which increasing h becomes desired again.

Final model

In the end, multiple models seemed to give very similar results. Some gave a better RMSE, but others gave a better RMSE around the Higgs mass and worse elsewhere. At some point, the differences were so small that conclusions on what should be improved could barely be made. One of the best models is described below and used in the rest of the results.

- Number of heads h: 8
- Embedding dimension d: 224 (28 per head)
- Number of self-attention layers n: 4
- Number of trainable parameters: 621.154
- No hidden layers
- Dropout rate: 0.0
- Batch size: 1000
- Number of epochs: 50 (used epoch 47)
- Activation function: SiLU

- Loss function: MSE
- Optimiser: AdamW
- Scheduler: OneCycleLR (initial & final LR: 10^{-8}, max. LR: 10^{-4})

4.1.3 Optimising for the kinematic mass

Since this dataset is a bit smaller, it is desired to increase the number of epochs. However, using for instance 50 epochs instead of 10, or increasing the model size, resulted in a better performance on the test sample, but in a worse performance on the SM sample. This indicated some sort of overtraining w.r.t. the SM sample. To be more precise, in the SM sample, the Higgs peak shifted to the right towards 140 GeV, while becoming smaller and wider. Any form of dropout and configuration of the hidden layers was unable to compensate for this. This complicated the optimisation.

Therefore, we decided to pick the model with the best loss for the SM sample instead of the validation sample. The predictions on the test sample will thus be suboptimal. The optimal model was already found in the first few epochs. Hence, the batch size was increased to 4000 and the number of epochs was set to 40, such that a smoother loss curve could be obtained. Already around epochs 7 to 10, the best models were found after which the loss curve started to increase again, illustrating the overtraining on the SM sample. The larger batch size then limited the complexity of the model, but again larger models showed worse performance on the SM sample as they became better suited for describing the training data. The best model was found to have 2 heads and 4 MHA layers. Again $d = h \cdot f_{jt}$ is used, with $f_{jt} = 25$. A summary of this model is given below:

- Number of heads h: 2
- Embedding dimension d: 50 (25 per head)
- Number of self-attention layers n: 4
- Number of trainable parameters: 33.402
- No hidden layers
- Dropout rate: 0.0
- Batch size: 4000
- Number of epochs: 40 (used epoch 7)
- Activation function: SiLU
- Loss function: MSE
- Optimiser: AdamW
- Scheduler: OneCycleLR (initial & final LR: 10^{-8}, max. LR: 10^{-4})

4.2 Predictions on test sample

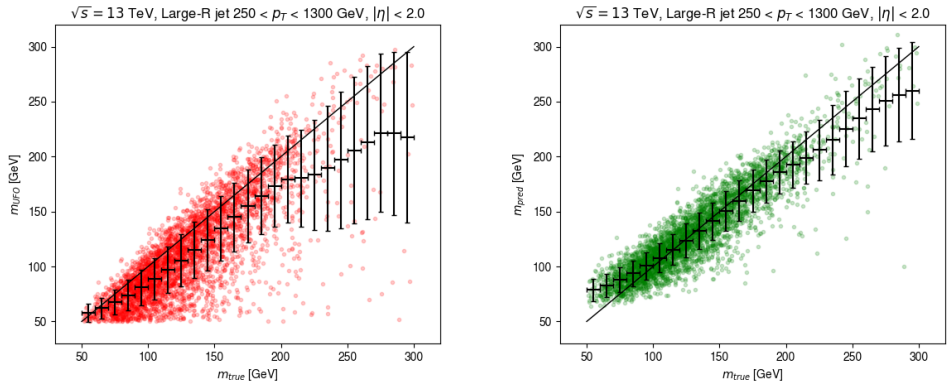
In this section, we present the results of the two models on the test samples. We use m_{UFO} as a reference to determine how much the ML model is able to improve w.r.t. solely using the conventional mass reconstruction techniques.

4.2.1 True jet mass

First, the results are presented and discussed that are obtained using the aforementioned model trained on the true jet mass.

Reco mass vs. predicted mass

In Figure 4.1a we plot the reco mass vs. the true jet mass together with the mean of m_{UFO} per m_{true} bin and its standard deviation. Here we clearly observe the lower cut of $m_{\text{UFO}} > 50$ GeV resulting from the 3D resampling. Also, we observe that the reco mass is often too low w.r.t. the true jet mass, whereas it is often not much too high. This causes the standard deviation in the lowest m_{true} bin to be artificially small. For higher values of m_{true}, we see the bias increasing towards lower values of m_{UFO}, especially beyond $m_{\text{true}} > 200$ GeV.



(a) **(b)**

Figure 4.1: (a) The reco mass m_{UFO} and (b) the predicted mass m_{pred} vs. the true jet mass m_{true}, together with their mean and standard deviation per m_{true} bin.

In Figure 4.1b we plot the predicted mass against the true jet mass. We observe that the standard deviations in each of the m_{true} bins have decreased, except in the lowest few bins, where there was less room for improvement. Moreover, in these bins the bias is significant. To explain this, two theories will be stated, one about shifting and one about clipping.

Firstly, this bias may be explained by assuming that the predictions are heavily influenced by m_{UFO}, which is likely due to the high correlation between the reco mass and the target mass. In that case *shifting* up all reco masses by a bit automatically reduces the bias and thus the RMSE of the entire dataset, but the consequence is that the reco masses in the low m_{true} bins have been shifted up too much. Including values of $m_{\text{UFO}} < 50$ GeV could cause this bias to vanish, or at least partially, whilst increasing the bias in m_{UFO} at low m_{true}. However, such values are not present in the data and thus this hypothesis could not be tested for the true jet mass. For the kinematic mass this was possible, but adding values of $m_{\text{UFO}} < 50$ GeV did not show a significant shift of the bias, indicating that this theory is not the main contributor to the bias. Even though the target variable and the results are slightly different, the results showed a similar sigmoid-shaped bias profile. This indicates that the bias might be a consequence of the same process, e.g. clipping.

Clipping means that the ML model pushes all the predicted values to within the interval of the target values. The reco mass only contains values between 50 and 300 GeV and the target variable is correlated to the reco mass. Therefore the target distribution is also somewhat limited to values between that interval, which allows for clipping to occur. Clipping can be beneficial when minimising the MSE loss as otherwise, for instance at low m_{true}, the predicted masses could be predicted to be above the 300 GeV. Since the

MSE scales quadratically, this would result in a large error contribution. Clipping then reduces this error at the cost of introducing some bias at the ends of the distribution.

As previously mentioned, a completely flat mass sample has been produced using undersampling. It uses true jet masses between 60 and 200 GeV, such that the statistics are not reduced too much. To test the effect of clipping a model was trained on this sample and it then predicted values of about 200 GeV for any m_{true} value beyond 200 GeV. This strengthens the expectation that clipping is mainly responsible for the bias at low and high values for m_{true}.

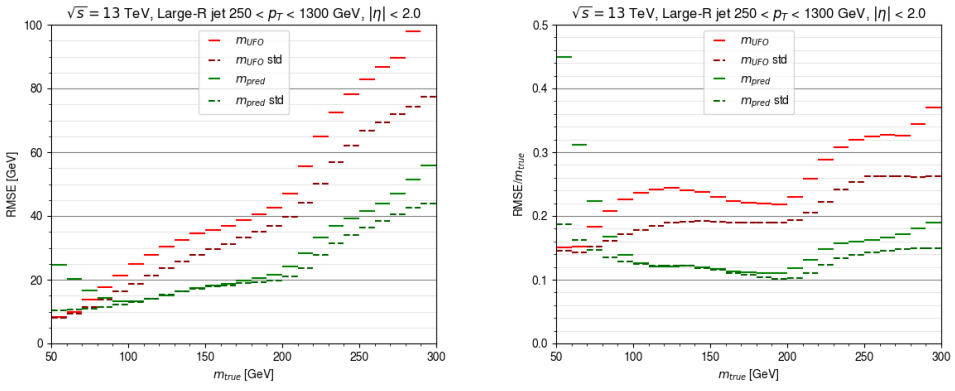
Clipping could be a sign that the model is not able to generalise well, at least not beyond the interval of the target values. However, applying a dropout rate of 0.5 was not able to solve the problem. To be more precise, it did not have any effect at all. This would indicate that there is no other more general pattern to be learnt given the model and the data. Smaller models did show differences, but then dropout mainly resulted in a degraded performance.

Error dependencies

In Table 4.1 we can see that the model is able to improve on the reco mass by a factor of 1.8 in terms of the RMSE. When looking at the Higgs region, i.e. within $100 < m_{\text{true}} < 150$ the improvement is 2.0, which is slightly larger as this cut excludes most of the biased data resulting at the ends of the m_{true} distribution.

Table 4.1: The RMSE of the reco mass and the predictions w.r.t. the true jet mass, and the improvement of the predicted mass w.r.t. the reco mass.

Selection	**RMSE [GeV]**		**Improvement**
	$\boldsymbol{m_{\text{UFO}}}$	$\boldsymbol{m_{\text{pred}}}$	
None	33	18	1.8
$100 < m_{\text{true}} < 150$ GeV	30	15	2.0



(a) **(b)**

Figure 4.2: (a) The RMSE of the reco mass m_{UFO} and the predicted mass m_{pred} per true jet mass m_{true} bin, alongside their standard deviation (std) w.r.t. their mean. (b) Similarly, the relative RMSE, i.e. the RMSE divided by the centre value of the m_{true} bin. (right)

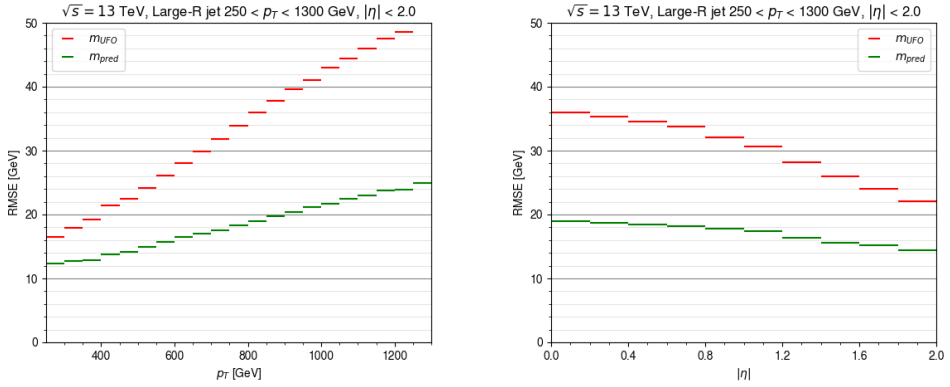
In Figure 4.2 we plot the RMSE and relative RMSE as a function of the true jet mass as well as the standard deviation per m_{true} bin. When the standard deviation coincides

with the RMSE, this means that there is no bias. Within 90 and 200 GeV the bias of the predicted values is minimal, as can also be seen in Figure 4.1b. For low m_{true} the bias contributes significantly to the RMSE, but for high m_{true} the bias contribution to the RMSE is smaller since the standard deviation is already relatively large. In Figure 4.2b we observe that the relative RMSE remains nearly constant between 100 and 200 GeV. At the Higgs mass of 125 GeV, the error is 15 GeV giving a relative error of 0.12.

As we can see in Figure 4.3, the predicted masses are better than the reco masses in all p_T and $|\eta|$ bins. The results are best at low p_T and high $|\eta|$, which is where the reco mass is best as well. This could partially be explained by the fact that at low p_T the decaying particles are further separated in the detector. This makes it easier to distinguish and remove background particles before determining the reco mass. Also, it causes signal particles to overlap less, hence their properties can be determined more accurately.

Since $|\eta|$ and p_T are slightly negatively correlated, as can be seen in Figure 3.4, we automatically obtain a decreasing RMSE as a function of $|\eta|$. Selecting only low p_T bins[1] causes the $|\eta|$ dependence of the RMSE to vanish, indicating that in that regime the $|\eta|$ dependence is indeed due to this correlation. Selecting only high p_T bins causes the dependence to remain. This could be because the b hadrons are able to travel further before they decay, allowing the decay vertex at high $|\eta|$ to be positioned inside the detector and thus better determined. Meanwhile, at low $|\eta|$ the b hadrons could have already passed the first few layers before they decayed, causing the track reconstruction to be less accurate.

The biggest improvements can be observed at high p_T and low $|\eta|$ as there was more room for improvement due to the higher RMSE of the reco mass. This is a promising observation, because boosted events are hard to reconstruct and because they are more sensitive to BSM physics. It underlines the potential that ML models have in such complex situations.



(a) **(b)**

Figure 4.3: The RMSE of the reco mass m_{UFO} and the predicted mass m_{pred} w.r.t. the true jet mass m_{true} per (a) p_T and (b) $|\eta|$ bin.

[1] This can be seen in Figure A.5.

Mass distributions

In Figure 4.4a the mass distributions are plotted. Here we observe that the mass distribution of the target, and thus the reco mass, is far from flat. Multiple aspects could be responsible for that. Firstly, a cut on the kinematic mass has been performed at generator level to only include values between 50 and 200 GeV, which explains the lack of events at high masses. Secondly, the energies and momenta of the particles, which are used to calculate the true jet mass, are smeared due to the limited resolution of the detector. This results in a smooth fall-off around the edges of the distribution. Thirdly, the cut on the reco mass causes another reduction of true jet mass values around the edges. Fourthly, missing particles like neutrinos and particles outside the large-R jet cannot be used to calculate the true jet mass and the reco mass. Last but not least, certain selection criteria in the jet reconstruction algorithms determine which jets are used, which can again alter the mass distributions. Hence the target distribution is not uniform anymore, which affects the predictions.

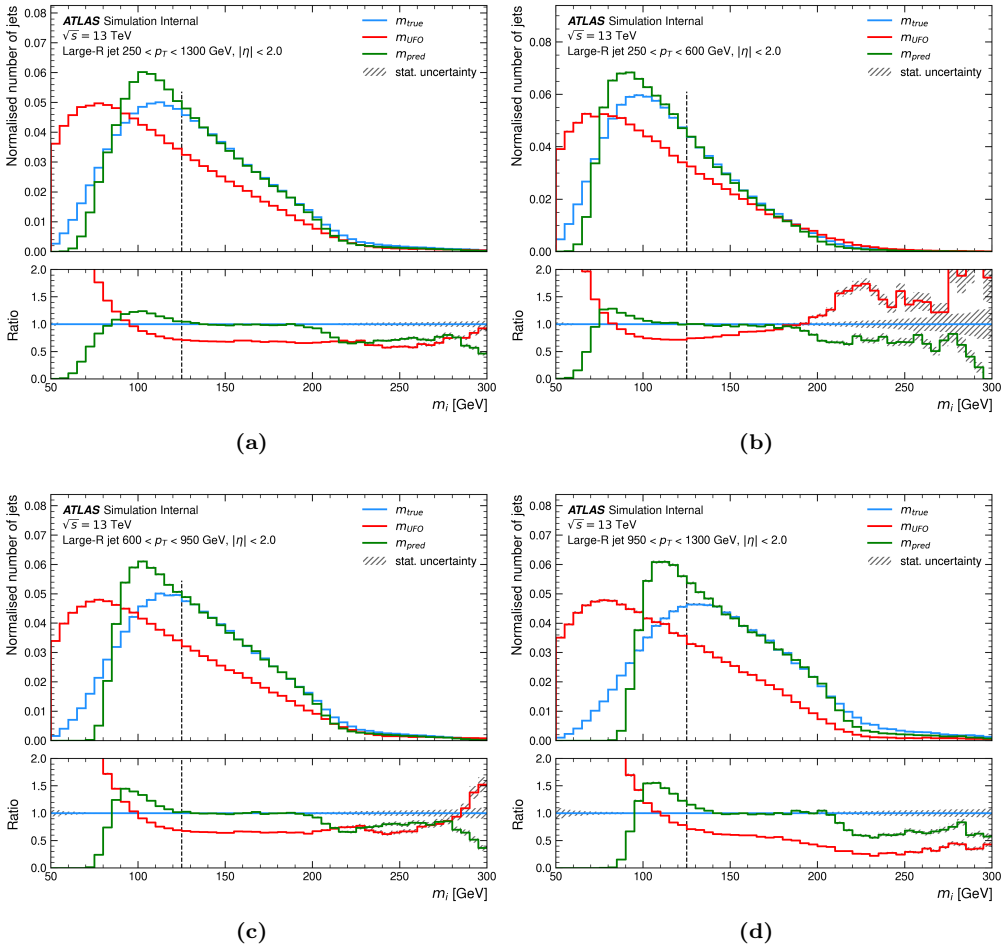


Figure 4.4: (a) The mass distribution of the true jet mass, reco mass and predicted mass and (b,c,d) these distributions for different p_T cuts.

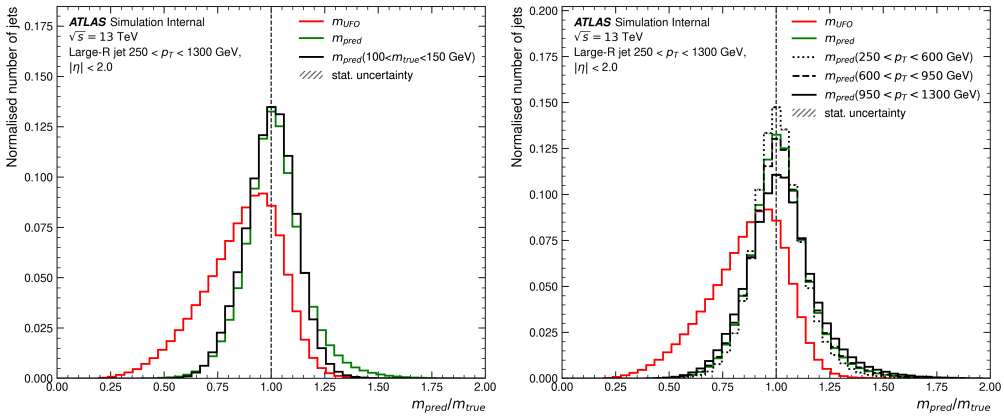
We observe that the predicted mass distribution better follows the true jet mass distribution as compared to the reco mass distribution. Still, we observe some biases that can be explained by clipping. At low masses, we observe a drop in the m_{pred} distribution. This is because at low m_{true} all predicted masses are too high, causing the absence of

low m_{pred} values. Similarly, a drop is present at the high end of the m_{pred} distribution, i.e. between $280 < m_{\text{pred}} < 300$ GeV, which is better visible in the ratio panel.

Due to this shift upwards at low m_{true} we observe an excess of m_{pred} values around 100 GeV. This excess towards 100 GeV could also be partially due to the fact that the target distribution contains a peak, despite the efforts for creating a relatively flat mass sample using a large decay width. Such a peak may cause the predictions to be pulled towards that peak. This automatically improves the RMSE by a little bit.

To test this hypothesis, the previously mentioned flat mass sample was used in a training with the same number of jets. As expected the excess around the peak was reduced, but the clipping effects became more significant. That could be because now the target variable has a steep cut-off at the edges of its distribution. This resulted in a worse RMSE. However, this could partially be due to the reduced statistics around the Higgs mass. To strengthen this hypothesis about the peak, we can also look at the mass distribution for different p_T cuts in figures 4.4b-4.4d. Then we see that the excess around the m_{pred} peak somewhat follows the peak of its target variable. It moves to the right for higher p_T. This shows the effect of the presence of the peak in the target values.

Furthermore, it seems that the model is relatively much influenced by the reco mass and possibly the other jet substructure variables. This is not unexpected since to each track and subjet all jet variables were concatenated. Firstly, the reduced abundance of predicted masses between 200 and 280 GeV seems to reflect the sudden change in the reco mass distribution at higher reco masses up to some extent. Secondly, when looking at high p_T, the steep cutoff in the m_{UFO} distribution seems to be reflected in the m_{pred} distribution, though at a different mass. Meanwhile, the true jet mass distribution clearly differs from the reco mass distribution. This explains why the predictions become worse at high p_T in terms of RMSE. However, around masses of 200 GeV, the m_{pred} distribution better adapts to the slight bulge in the m_{true} distribution for different p_T cuts. This indicates that the model is able to move away from the m_{UFO} distribution up to some point.



(a) **(b)**

Figure 4.5: The relative mass distributions, i.e. the reco and predicted mass divided by the true jet mass, for (a) different m_{true} cuts and (b) different p_T cuts.

In Figure 4.5 the relative mass distribution is plotted for different m_{true} and p_T cuts. Here we observe that the predicted masses create a stronger, better-centred and more symmetric peak as compared to the reco masses. When performing a cut on the true

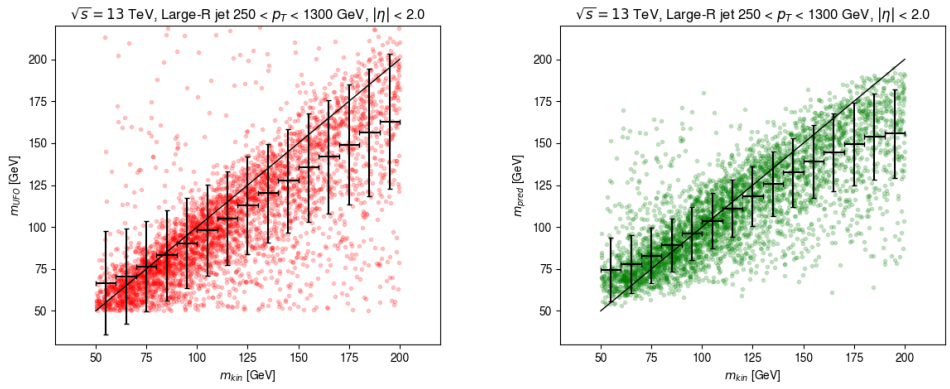
jet mass to minimise the clipping effect at the borders we mainly observe fewer too-high predicted masses, resulting in an even better peak. That is because relatively many events have low m_{true} values, which is exactly where the predictions are often too high. When cutting on p_T, we observe that for lower p_T the peak becomes stronger, as expected from Figure 4.3a. Also, fewer too-high predicted masses remain, making the peak even better-centred.

4.2.2 Kinematic mass

In this section, the results are presented and discussed that are obtained using the aforementioned model trained on the kinematic mass.

Reco mass vs. predicted mass

In Figure 4.6 both the reco and the predicted mass are plotted as a function of the kinematic mass. The kinematic mass is a variable that is less related to the reco mass as compared to the true jet mass. Therefore the deviations of the reco masses w.r.t. the kinematic mass can be much larger. This time outliers in the reco masses are visible both above and below the target values, which causes the standard deviation of the reco mass at low m_{kin} to not be artificially small. Still at low m_{kin} the cut in the reco mass causes the mean to be artificially high. The outliers are also visible in the predicted masses, which causes the effect of clipping to be more clearly visible. All predictions are now confined to form a rectangle-like figure. At low m_{kin} no predictions below the target values are found, but only above them. This automatically causes the mean to be too high at low m_{kin} and vice versa too low at high m_{kin}.



(a) **(b)**

Figure 4.6: (a) The reco mass m_{UFO} and (b) the predicted mass m_{pred} vs. the kinematic mass m_{kin}, together with their mean and standard deviation per m_{kin} bin.

To test if - besides clipping - the lower cut on the reco mass could be responsible for this bias at low m_{kin}, we included masses of $m_{\text{UFO}} < 50$ GeV during a training. However, the bias was still present, because in this dataset a cut on the kinematic mass has been performed to only include $50 < m_{\text{kin}} < 200$ GeV. This indirectly results in a similar though less steep decline in the reco mass distribution around $m_{\text{UFO}} = 50$ GeV. Moreover, including $m_{\text{UFO}} < 50$ GeV introduces a second m_{UFO} peak around 20 GeV that mostly consists of misidentified QCD background. The corresponding kinematic mass is thus incorrect and hard to predict. The model simply shifts these masses up to around the mean of the target distribution, to reduce the MSE loss. Meanwhile, we

found that clipping is not reduced. That may be because this shifting of low m_{UFO} values is a similar operation as clipping, as again the predictions are pushed to within the interval of the target distribution. Therefore, excluding values of $m_{\text{UFO}} < 50$ GeV during the training improves the predictions.

Including $m_{\text{UFO}} < 50$ GeV after the training does not reduce the bias from clipping either. Instead, the jets belonging to these reco masses are predicted to all have values somewhere between about 50 and 100 GeV. Conclusively, the bias is a consequence of a clipping model and not so much due to the absence of low reco masses in the test sample.

The bias in the mean of the predictions in Figure 4.6b is a bit exaggerated due to clipping since for instance at low kinematic masses only too high predictions remain, increasing the mean artificially. Yet the bias is still clearly recognisable as a sigmoid-shaped curve. This means that only the predicted masses away from the edges, for example within $100 < m_{\text{kin}} < 150$ GeV, are relatively physical. In the SM sample $m_{\text{kin}} = 125$ GeV, hence this model still has predictive power for $H \to b\bar{b}$ events.

As compared to the reco masses, the standard deviation of the predicted masses per m_{kin} bin is reduced in all bins. Moreover, the mean is better centred in the middle to high kinematic mass bins. In the bin including the Higgs mass (120-130 GeV), the bias in the mean of the predictions is reduced as compared to that of the reco mass. The bias around the Higgs mass will be better quantified using the SM sample. At low kinematic masses, the mean of the predictions again deviates more than the mean of the reco masses. However, the relatively accurate mean of the reco mass is again a consequence of the artificial lower cut of m_{UFO}. Including smaller reco masses would cause the mean of the reco masses to be much too low in all bins. Hence the predicted mass improves on the reco mass both in terms of the mean and the standard deviation.

Error dependencies

In Table 4.2 we can see that the model is able to improve on the reco mass by a factor of 1.4. When looking at the Higgs region ($100 < m_{\text{kin}} < 150$) the improvement is increased to 1.6 as this mass cut reduces the effects of clipping.[2]

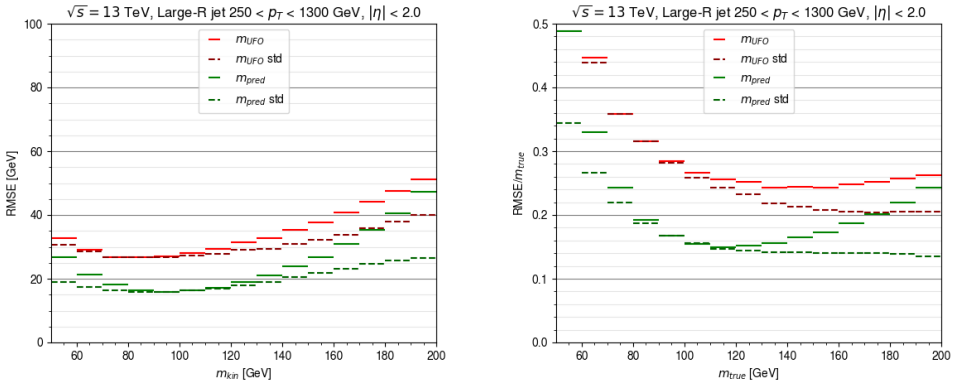
Table 4.2: The RMSE of the reco mass and the predictions w.r.t. the kinematic mass, and the improvement of the predicted mass w.r.t. the reco mass.

Selection	**RMSE [GeV]**		**Improvement**
	$\boldsymbol{m_{\text{UFO}}}$	$\boldsymbol{m_{\text{pred}}}$	
None	33	24	1.4
$100 < m_{\text{kin}} < 150$ GeV	31	19	1.6

In Figure 4.7a we plot the RMSE as a function of the kinematic mass. Here we see that the predicted mass improves on the reco mass in all bins. The biggest improvement and best RMSE can be found between 80 and 130 GeV. In the bin containing the Higgs mass (120-130 GeV), the RMSE is 19 GeV and the relative RMSE is 0.15.

A discrepancy between the RMSE and the standard deviation again displays a bias. This time the bias is only negligible in a relatively small region, i.e. within 80 and 140 GeV, whereas the training on the true jet mass showed a negligible bias within 100 and 200 GeV. This may be because the kinematic mass is confined to a smaller interval, i.e.

[2]Larger models allow for an improvement of 1.5 and 1.8 resp. at the cost of a worse performance on the SM sample, which is why they are not considered.

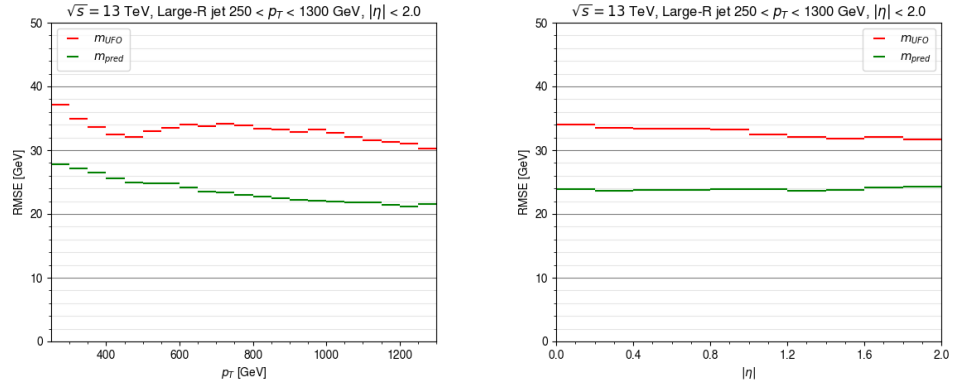


(a) **(b)**

Figure 4.7: (a) The RMSE of the reco mass m_{UFO} and the predicted mass m_{pred} per m_{kin} bin, alongside their standard deviation (std) w.r.t. their mean. (b) Similarly, the relative RMSE, i.e. the RMSE divided by the centre value of the m_{kin} bin.

50-200 GeV rather than 50-300 GeV. Again at the ends of the mass range, the bias is found to be maximal, which coincides with the findings from Figure 4.6b about clipping. This time the contribution of the bias to the RMSE is biggest at high masses, which is best visible in Figure 4.7b. At high masses, the relative spread in the predictions remains constant, while the relative RMSE increases again, similarly to m_{UFO}.

In Figure 4.8 we plot the RMSE as a function of p_T and $|\eta|$. Again we observe that the predicted masses have a lower RMSE than the reco mass in all p_T and $|\eta|$ bins. The RMSE is nearly independent of $|\eta|$, but the best improvement is visible at low $|\eta|$. The RMSE and the improvement thereof are best at high p_T. Again, at high p_T the particles of the b jets overlap in the detector. One of the goals of using an ML algorithm is to show that they are able to improve on conventional reconstruction algorithms, especially in these hard situations. This observation does exactly that, which is promising for BSM searches.



(a) **(b)**

Figure 4.8: The RMSE of the reco mass m_{UFO} and the predicted mass m_{pred} w.r.t. the kinematic mass m_{kin} per (a) p_T and (b) $|\eta|$ bin.

Since the kinematic mass data is not 3D resampled, p_T values beyond the range of 250-1300 GeV are available, i.e. values of $p_T > 150$ GeV. For $p_T > 1300$ GeV the RMSE of both the predicted and the reco mass slowly starts to increase again. Meanwhile, the improvement starts to decrease, as fewer statistics are available at high p_T. Still, the predicted masses are better than the reco masses in all p_T bins.

Mass distributions

In Figure 4.9a we plot the mass distributions of the reco, predicted and kinematic mass. Here we see that the predicted mass follows the kinematic mass distribution better than the reco mass distribution does, albeit with a different shape. Again the effect of clipping is well visible as for low and high kinematic masses there is a sudden lack of predicted masses, whereas in the middle too many events are predicted.

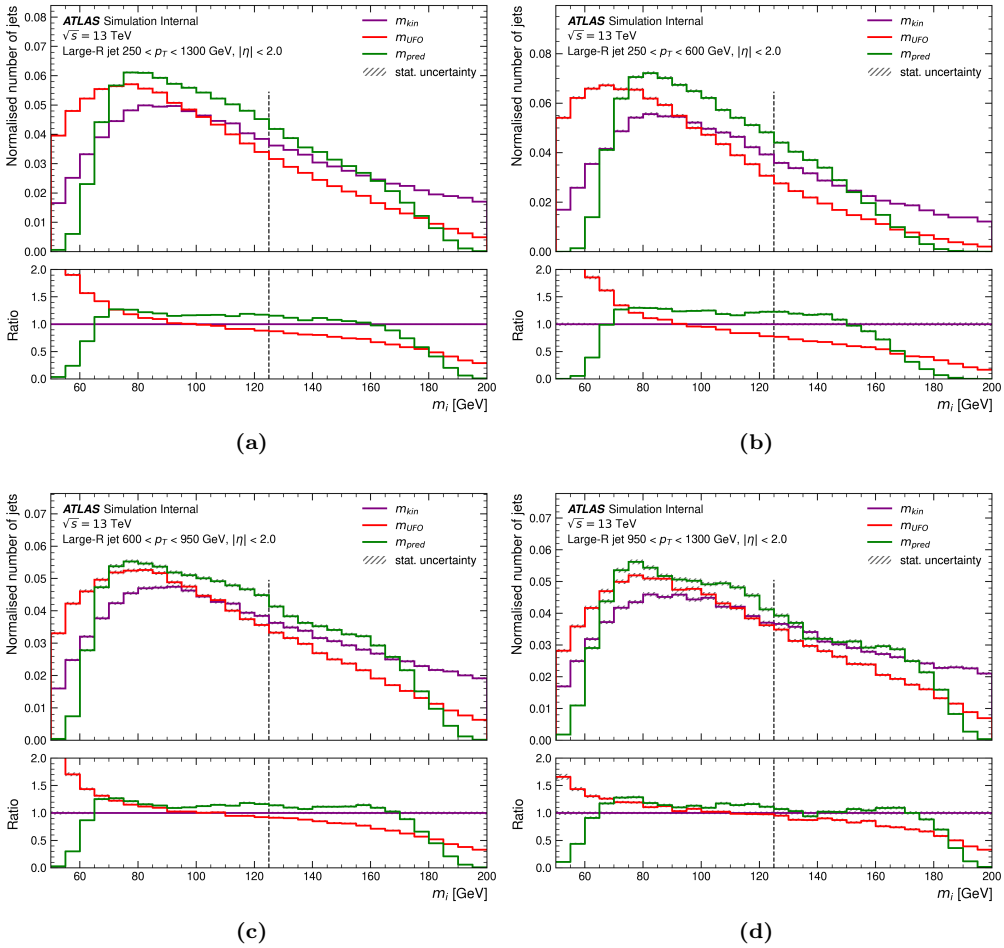
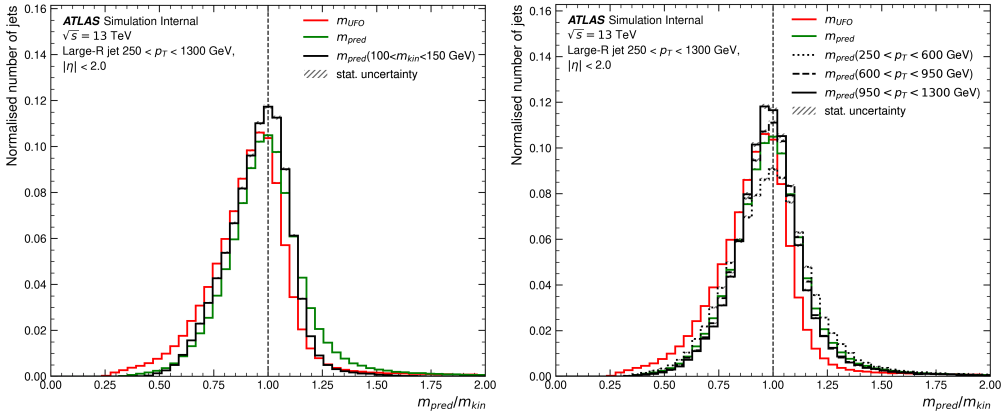


Figure 4.9: (a) The mass distribution of the kinematic mass, reco mass and predicted mass and (b,c,d) these distributions for different p_T cuts.

When plotting the mass distributions for different p_T cuts we observe that for higher p_T the predicted mass distribution better follows the target distribution, as expected from Figure 4.8a. This may be explained by the fact that the reco mass distribution also better matches the target distribution. No 3D resampling has been performed on the kinematic mass dataset, making the reco mass distribution p_T dependent. This

may explain why training on the kinematic mass causes the shape of the predicted mass distribution to better follow the shape of the target distribution, as compared to training on the true jet mass. Also, we observe in the ratio plots that for higher p_T the effect of clipping is reduced. The sudden ratio drop only starts closer to the edges of the distribution, while the over-abundance in the middle is reduced.



(a) **(b)**

Figure 4.10: The relative mass distributions, i.e. the reco and predicted mass divided by the kinematic mass, for (a) different m_{kin} cuts and (b) different p_T cuts.

When looking at the relative mass distribution in Figure 4.10 we find that the relative m_{pred} distribution is better centred around 1 than the relative m_{UFO} distribution. The peak gets slightly sharper after performing a mass cut to reduce the clipping effects or after a high p_T cut.

4.3 Predictions on SM sample

In this section, we present the results of the training on both the true jet mass and the kinematic mass using the SM sample. In the upcoming figures, the results from the true jet mass training are displayed on the left and those from the kinematic mass training on the right to allow for easy comparisons. From the different available processes, we will use the two-lepton channel $q\bar{q} \to ZH \to \ell\bar{\ell}b\bar{b}$ as that process has the most statistics. All the other channels, including the $H \to c\bar{c}$ channels, provide very similar results.[3] Only events having $|\eta| < 2$, $250 < p_T < 1300$ GeV and $50 < m_{\text{UFO}} < 200$ GeV are used. The latter upper cut should not have much effect as now all events have a reco mass near 125 GeV, but due to clipping it does, as will be explained in the next paragraph.

4.3.1 Improvement predictions

In Table 4.3 we show the RMSE of the reco and predicted masses w.r.t. the two different target variables for both models. In contrast to the results using the test samples we now see that the improvement is best for the training on the kinematic mass. Its improvement is even larger (2.0 instead of 1.6), which is partially because now only kinematic masses around 125 GeV are present, which is where the predictions are better. Meanwhile, the improvement is partially enhanced due to the effects of clipping. As stated earlier, the reco mass is sometimes able to deviate much from the kinematic mass as they are less

[3] This can be seen in Figure A.3 and A.4.

related. Since the error scales quadratically these deviations significantly contribute to the RMSE. Meanwhile, the predictions are confined to values between 50 and 200 GeV due to clipping. This automatically improves the predictions. Including events with $m_{\text{UFO}} > 200$ GeV increases the improvement factor up to 2.4 already. To somewhat reduce the effects of clipping and to make the comparison to m_{UFO} fairer, we only use $50 < m_{\text{UFO}} < 200$ GeV. Tighter cuts could be made to reduce the effect of clipping even further, but this would significantly alter the Higgs peaks and artificially increase the precision of the predictions even further.

Table 4.3: The RMSE of the reco mass and the predictions w.r.t. the kinematic mass and the true mass for the two trainings on the different target variables, alongside the improvement of the predicted mass w.r.t. the reco mass.

Target	**Error w.r.t.**	**RMSE [GeV]**		**Improvement**
		$\boldsymbol{m_{\text{UFO}}}$	$\boldsymbol{m_{\text{pred}}}$	
m_{kin}	m_{kin}	22	11	2.0
m_{kin}	m_{true}	18	18	1.0
m_{true}	m_{true}	18	13	1.4
m_{true}	m_{kin}	22	18	1.2

Meanwhile, the improvement on the reco mass using the training on the true jet mass is reduced (1.4 instead of 2.0). This can be explained by the fact that the p_T distribution in the SM sample is falling, whereas it is flatter (and contains a peak around 500 GeV) in the test sample. Hence relatively more low p_T events are present in the SM sample. For these p_T values we found that the resolution is best, which explains why the RMSE is now better (13 instead of 15 GeV). Meanwhile, the improvement factor is worse, as the reco mass is already able to predict the true jet mass well at low p_T.

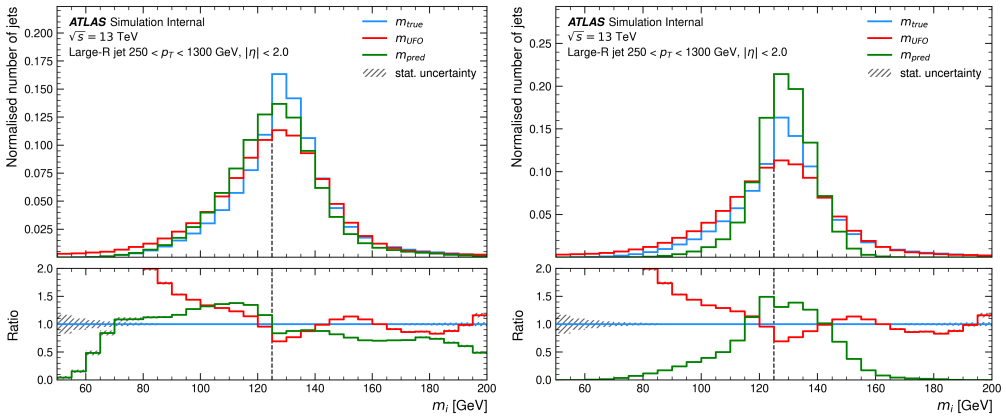
In Table 4.3 we can also see that the predictions from the training on the true jet mass automatically predict the the kinematic mass slightly better than the reco mass. Vice versa, the kinematic mass model is not able to better predict the true jet mass.

4.3.2 Signal peaks

In Figure 4.11 we plot the mass distributions of the true jet mass, kinematic mass and reco mass as well as the predicted mass from the two models. Both models are able to give stronger Higgs peaks than the reco mass does. Both models reduce the tails, but at the very ends of the distribution this is mainly due to clipping as can best be seen in the ratio plots in Figure 4.11a. This time the slightly reduced abundance of the true jet mass predictions around 200 GeV is due to the upper cut on m_{UFO} and not due to clipping, as the latter only arises near 300 GeV.

A bias in the mean can be observed in the predicted distribution as well as in Table 4.4, where the mean and standard deviations of the different masses are quantified. First of all, the true jet mass has a bias as it is centred around 129 GeV rather than 125 GeV. The mean of its predictions is 126 GeV and is therefore shifted to the left. This shift is partially a consequence of the peak in the training sample, since training on the flat mass sample showed a reduced bias.

Peculiarly, the standard deviation of the predicted true jet masses is smaller than that of the target values, the true jet mass. This is partially a consequence of clipping, as similarly in the test sample all predicted masses were pushed to within the target mass range. Also, it is partially because of the peak in the training sample, which causes the predictions to lean towards the mean of the peak. Training on the flat mass sample



(a) **(b)**

Figure 4.11: The true, reco and predicted mass distributions when trained on (a) the true jet mass and (b) the kinematic mass using a physical SM standard model sample. The kinematic mass distribution is represented by the dashed vertical line at 125 GeV. Only reco masses of $50 < m_{\text{UFO}} < 200$ GeV are used.

Table 4.4: The mean and standard deviation of the distributions of the kinematic mass[4] m_{kin}, true jet mass m_{true}, reco mass m_{UFO} and predicted mass using the training on the kinematic mass $m_{\text{pred,kin}}$ and that on the true jet mass $m_{\text{pred,true}}$.

Distribution	$\boldsymbol{\mu}$ **[GeV]**	$\boldsymbol{\sigma}$ **[GeV]**
m_{kin}	125.0	0.3
m_{true}	129	20
m_{UFO}	125	22
$m_{\text{pred,kin}}$	128	10
$m_{\text{pred,true}}$	126	18

created a broader SM Higgs peak, as compared to the same training with a peaked target and the same number of jets. But this could also be due to the reduced statistics around the Higgs mass. Meanwhile, in that training clipping still caused the predicted mass peak to be narrower than the target peak, as it was pushed to within 60 and 200 GeV.

Secondly, the kinematic mass distribution has its mean at 125 GeV, while its predicted distribution has its mean at 128 GeV. As previously mentioned, training on the kinematic mass using larger models caused the peak of its predictions to move to the right. By selecting the optimal model for the SM sample, this bias is already reduced, but still non-negligible. The exact reason as to why it occurs is unknown, but it may again have to do with the domain shift of the kinematic variables in the SM sample w.r.t. the test sample.

The standard deviation of $m_{\text{pred,kin}}$ is 10 GeV, which is only slightly smaller than its RMSE of 11 GeV. This indicates that the bias contribution is relatively small. Moreover, since the peak is relatively narrow, the effect of clipping on the distribution is only little as no sudden decline in the ratio plots is observed near the edges. On the other hand, the standard deviation in $m_{\text{pred,true}}$ of 18 GeV is fairly large as compared to its RMSE of

[4]The standard deviation of the m_{kin} distributions is not equal to the theoretical decay width of 4 MeV since it is adjusted by the Complex Pole Scheme (CPS) in the simulation to account for the effects of complex singularities near the threshold energies.

13 GeV. This indicates that the deviation in true jet mass strongly limits the maximum achievable resolution of the peak.

When comparing the peaks obtained from the two models, we observe that the kinematic mass model is able to produce stronger peaks than the true jet mass model. More importantly, the peak predicting the kinematic mass is stronger than the true jet mass. This means that any adaption to the training on the true jet mass will not be able to produce stronger peaks than the training on the kinematic mass, at least not for $H \to b\bar{b}$ events.

4.3.3 Transverse momentum and pseudorapidity

We would like to make sure that the physical properties of a sample are unchanged after making a selection in the data using the predicted mass. This is especially important in boosted decision trees, which try to filter out signal events using cuts on the data. The p_T and $|\eta|$ distributions are characteristic of a process, therefore it is important that they do not behave unphysically after cutting on the predicted mass. Checking these two variables is not representative of all other variables, but it gives us a quick indication.

In Figure 4.12 we plot the p_T and $|\eta|$ distribution of the SM sample, using cuts on different mass variables. The distributions are naturally untouched by cuts on the kinematic mass and we observe that they are nearly untouched by cuts on the reco mass. Meanwhile, a cut on the true jet mass does pose a clearly different p_T distribution in the ratio plots. We observe a slight excess at low p_T after cutting on the true jet mass, which explains why in all higher p_T bins a clear shortage arises.

Cutting on $m_{\text{pred,true}}$ causes the p_T distribution to mimic the p_T distribution after cutting on m_{true}. This is desired if you want the predictions to mimic the target values, however, in this case, it means that the physical properties of the sample are adjusted. Cutting on $m_{\text{pred,kin}}$ on the other hand causes the p_T distribution to better retain its shape. It only slightly deviates from the p_T distribution after cutting on m_{kin}. For $p_T > 600$ GeV the shortage is quite constant and clearly better than when cutting on $m_{\text{pred,true}}$.

The $|\eta|$ distributions are nearly untouched after cutting on any of the masses. Cutting on m_{true} poses the largest change, though it is still small. Cutting on $m_{\text{pred,true}}$ causes the p_T distribution to slightly follow the p_T distribution after cutting on m_{true} as expected. Meanwhile, cutting on $m_{\text{pred,kin}}$ leaves the p_T distribution virtually untouched, as desired.

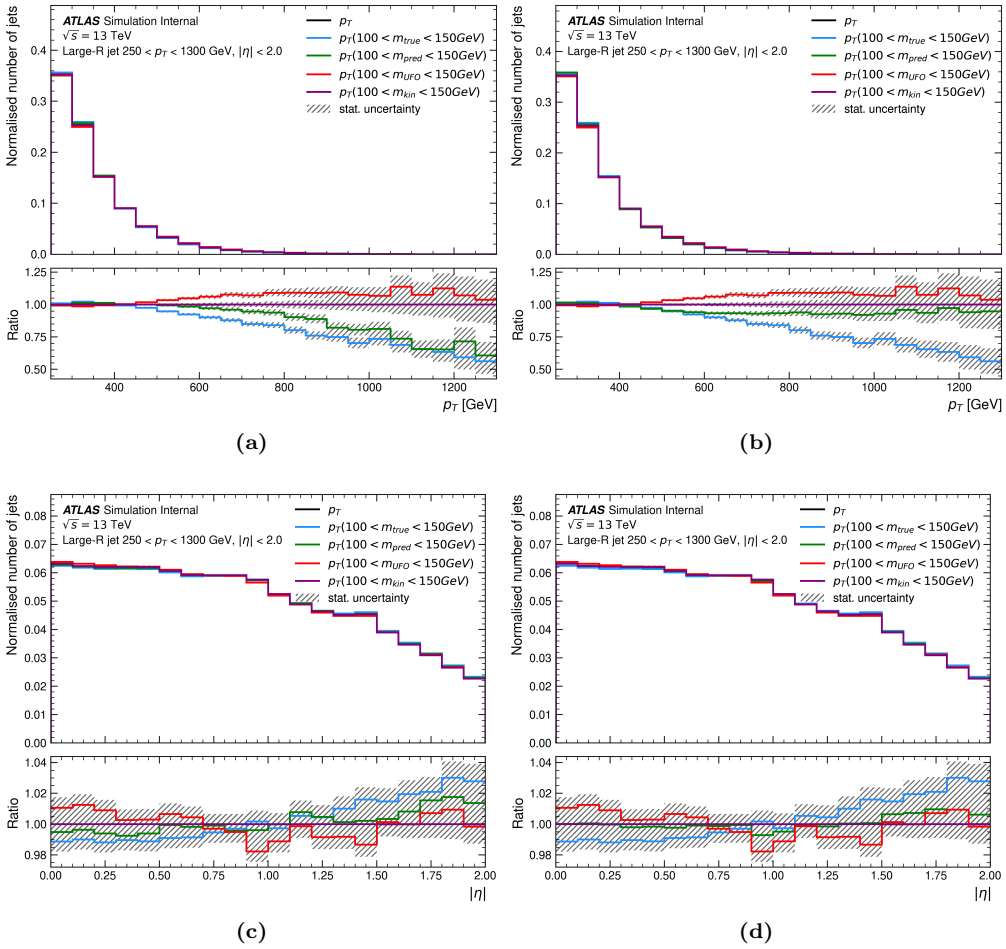


Figure 4.12: The (a,b) p_T and (c,d) $|\eta|$ distribution for different mass cuts, including the true jet, kinematic and reco mass as well as on the predicted mass trained on (a,c) the true jet mass and (b,d) the kinematic mass. Only reco masses of $50 < m_{\mathrm{UFO}} < 200$ GeV are used.

Chapter 5

Conclusion & Discussion

After investigating and optimising the ML models for the two target variables, the true jet mass and the kinematic mass, we found that the kinematic mass model has the most potential. Firstly, it is able to produce stronger SM Higgs peaks than both the true jet mass and the predicted true jet mass. Secondly, cutting on the predicted kinematic mass affects the p_T and $|\eta|$ distribution less than cutting on the predicted true jet mass. This implies that the physical properties of the Higgs decay are less affected using this model, which is desired in BDTs. Finally, at high p_T the RMSE is not increasing (much) as a function of p_T and the improvement w.r.t. the reco mass is largest. This is desired regarding the search for BSM physics using boosted $H \to b\bar{b}$ jets. Conversely, the error of the true jet mass model does increase as a function of p_T.

Both models showed the best improvement at high p_T and low η. At high p_T the b jets overlap most, hence this observation underlines the strength of ML models in such complex situations. The model predicting the true jet mass improves on the reco mass by a factor of 2.0 in the Higgs range in terms of the RMSE, using the test sample. In the SM sample, it improves by a factor of 1.4 and it provides a Higgs mass peak described by $\mu = 126$ GeV and $\sigma = 18$ GeV. The model predicting the kinematic mass improves on the reco mass by a factor of 1.6 in the Higgs range using the test sample, while it improves by a factor of 2.0 in the SM sample. It produces a Higgs peak described by $\mu = 128$ GeV and $\sigma = 10$ GeV. For comparison, the reco mass peak is described by $\mu = 125$ GeV and $\sigma = 22$ GeV and the true jet mass peak by $\mu = 129$ GeV and $\sigma = 20$ GeV. The spread of the predicted masses from the true jet mass model is smaller than that of its target values due to clipping, which pushes all predictions to within the interval of the target values. Similarly, the improvement w.r.t. the reco mass by the kinematic mass model is enhanced by clipping.

When training on the kinematic mass, the substructure variables worsened the results and the pixel information improved them. Vice versa, when training on the true jet mass the substructure variables improved the results, whereas the pixel information did not. This indicates that low-level information has more potential when it comes to predicting the kinematic mass. Higher level information would be more useful to predict the true jet mass, as the groomed UFO large-R jet (that is used to calculate these variables) is more closely related to the target variable. In both models using subjets was beneficial.

In both models clipping altered the mass predictions unphysically near the edges of the target mass distribution. However, the two models trained on m_{true} and m_{kin} have predictive power near the Higgs mass, i.e. within approximately $90 < m_{\text{true}} < 200$

GeV and $80 < m_{\text{kin}} < 140$ GeV respectively. Even in background processes, that are not used in the training, the predicted masses improve on the reco mass. Both models provide similar results for $H \to b\bar{b}$ and $H \to c\bar{c}$ events. Most importantly, the models do not simply predict the Higgs mass. Instead, they provide an increased resolution over a broad kinematic range, thanks to the broadened mass peak that was used in the training sample. Therefore, the recommended kinematic mass model, or slightly adapted versions thereof, could provide us with better results when applied in $H \to b\bar{b}$ research. This would help us to better understand the Higgs boson and related open questions about the universe.

5.1 Future studies

5.1.1 Input variables

The predictions can potentially be improved in numerous ways. Firstly, extra variables could be included that were not available in the provided data. Integer values were not compatible with the Salt code up to some point, hence they have not been considered yet. The variable `leptonID` could be included, which indicates whether a track has been used in the reconstruction of an electron, a muon, or neither of them. The GN2X tagger showed an improvement in the rejection rate of about 10-15% when using this variable. [62] Hence including the `leptonID` could potentially improve the mass resolution.

The number of muons ending up in the large-R jet could also improve the mass resolution. Usually, the weak bosons that are associated with the Higgs boson are produced back to back, meaning that its muons do not end up in the large-R jet. But in the few cases that it does, it does contribute to the reco mass, which you could compensate for using the number of muons. Currently, the leptons decaying from the weak boson are included when calculating the true jet mass, but this limits the maximum achievable resolution when training on the true jet mass. Hence we could remove these leptons before calculating the true jet mass at truth level. At both truth and detector levels, it is often possible in the 2-lepton channel to remove identified lepton pairs having an invariant mass consistent with the Z boson mass, before computing the reco mass and the true jet mass. In the 1-lepton channel, this cannot be done so straightforwardly. Hence it could still be useful to include the number of muons to compensate for the extra mass. Furthermore, information from the muon spectrometer could be added similarly to the way the subjets have been added.

In [27] the grooming techniques CS+SK+Soft Drop were recommended when using UFO jets, since this would result in the best performance for taggers. However, in Figure 16a of that paper the grooming techniques CS+SK+Trimming seem to give better results in terms of the mass resolution, especially for $p_T > 1500$ GeV. This was the case for $W \to q\bar{q}$ events after performing a mass cut of $70 < m_{\text{true}} < 90$ GeV. In this paper, $H \to b\bar{b}$ events were not investigated. Meanwhile, no significant improvement was observed in $t \to qqb$ events in Figure 16b after selecting only $150 < m_{\text{true}} < 200$. So it is uncertain whether Trimming could be beneficial.

Alternatively, Flow objects could be used as input, also known as Particle-Flow Objects (PFlow). Flow objects are objects generated from energy deposits in the calorimeter. Flow objects can be concatenated to the tracks that create these energy depositions to create a more informative track-like object. However, on one hand, some tracks are not associated with a flow object, and multiple tracks could end up in the same spot in the calorimeter. On the other hand, neutral particles could lead to the creation of flow objects without leaving a track behind. One could then set zeros for the missing

object when using a combined track+flow object, but many zeros can confuse an ML algorithm. Still, combining flow and track information could potentially improve the mass resolution.

5.1.2 Architecture

Secondly, apart from using different input variables, different architectures could be used. Currently, the optimal number of hidden layers is zero, as apparently relatively untouched input variables are more useful. However, this does not allow for the creation of sophisticated variables, which could potentially cause the MHA to better attend to the right objects of the sequence. Perhaps creating three separate dense layers for the query, key and value allows for better-suited embeddings. In this setup using more hidden layers could become beneficial again.

Moreover, one could choose to not put some (or all) substructure variables inside the MHA, since they are currently abundant as compared to the rest of the information. All jet variables are copied and concatenated to each track, which could cause such variables to dominate more than they should. Instead, only a few variables could be appended to each track, like η, p_T, E and the reco mass, since related variables are also present in the tracks and subjets. The rest of the variables, i.e. the substructure variables could then be appended to the output of the MHA via a dense layer. This would significantly reduce the computation time, but it could result in degraded performance when training on the true jet mass. When training on the kinematic mass, it could still be beneficial as in that case, the substructure variables would no longer be dominant in the MHA, while still providing potentially useful information.

5.1.3 Clipping

Thirdly, clipping effects could and should be reduced. Fortunately, we observed that only near the edges of the target distributions clipping significantly altered the predictions. Within approximately $70 < m_{\text{kin}} < 140$ GeV and $90 < m_{\text{true}} < 200$ GeV the bias was much reduced, increasing the predictive power. This still means that the model may not provide useful results for background processes having a true jet mass or kinematic mass far away from the Higgs mass. This could perhaps be solved in multiple ways, which could improve the predictions on $H \to b\bar{b}$ events at the same time.

This bias could be reduced by applying a loss function that is dependent on the target value, such that at the ends of the distribution the errors weigh stronger. This would automatically worsen the predictions in the middle of the distribution, i.e. around the Higgs mass. However, potentially the custom loss function could push the model out of a possible local minimum and push it towards a hopefully lower minimum that better generalises outside the target mass distribution.

This bias could also be reduced by including background events during the training and thus forcing to model to generalise. In Figure A.1, we find that the true jet mass model also better predicts $H \to c\bar{c}$, QCD and top processes than the reco mass over a relatively wide mass range. The kinematic mass model was also trained on $H \to c\bar{c}$ events and is therefore able to produce similar results on $H \to c\bar{c}$ events, as can be seen in Figure A.2. Background samples containing the kinematic mass were not available. A training on $H \to b\bar{b}$, $H \to c\bar{c}$, QCD and top all together provided better results on the background processes, however, at the cost of a worse resolution in the $H \to b\bar{b}$ events. Meanwhile, clipping was not reduced, because the m_{UFO} distribution was the same for all processes due to 3D resampling. Therefore, using a different resampling method could allow for a wider trainable mass range. This could reduce the effects of

clipping while simultaneously increasing the predictive power in all processes, including $H \to b\bar{b}$.

Simply increasing the target mass range, while only training on signal events would perhaps be the best solution. The flat true jet mass sample containing a smaller cut of $60 < m_{\text{true}} < 200$ GeV (instead of $50 < m_{\text{true}} < 300$ GeV) showed that the effects of clipping are increased when using smaller ranges. All predictions of this model ended up within $60 < m_{\text{pred}} < 200$ GeV, even those way beyond that range. Therefore, vice versa simply using an even larger mass range would likely boost the performance of the ML model while decreasing the effects of clipping near the Higgs mass. In that way, background processes could be described better, without having to use them in the training, which simplifies the task of the model.

To obtain a wider mass range, one could perform a reweighting scheme by altering the masses, momenta and energies, such that the mass distribution becomes flat over a wide mass range while not breaking the laws of physics. [63] This would also make the effects of a peaked distribution vanish. Alternatively, a much simpler solution could be tested, i.e. performing a wider cut on m_{true} and m_{kin}. The m_{true} distribution did not contain many events beyond the 200 GeV, due to the kinematic mass cut at generator level at that mass. But the remaining events already made sure that clipping did not significantly contribute to the RMSE at the end of the m_{true} distribution. Therefore removing the upper cut on both m_{kin} and m_{true} could improve the predictions for both models. When removing the cut on the kinematic mass at generator level, you would include way too many events at unnecessarily high masses as the decay width was chosen to be 400 GeV. So some cut has to be performed. However, any cut causes the mass distribution to decrease around that cut. Increasing the upper cut of the kinematic mass to 250 or 300 GeV could already reduce much of the clipping effects around the Higgs mass and improve the predictions for background processes.

Appendix A

Additional results and clarifications

Table A.1: Jet variables with their name as used in the code and their notation.

Name	Notation
pt	p_T
eta	η
abs_eta	$\|\eta\|$
energy	E
mass	m_{UFO}
Tau1	τ_1
Tau2	τ_2
Tau3	τ_3
Tau4	τ_4
Tau42	τ_{42}
Tau32	τ_{32}
Tau21	τ_{21}
ECF1	ECF_1
ECF2	ECF_2
ECF3	ECF_3
C1	C_1
C2	C_2
D2	D_2
Split12	$\sqrt{d_{12}}$
Split23	$\sqrt{d_{23}}$
Split34	$\sqrt{d_{34}}$
ZCut12	z_{12}
ZCut23	z_{23}
ZCut34	z_{34}
KtDR	$k_t \Delta R$
Angularity	τ_a
PlanarFlow	P

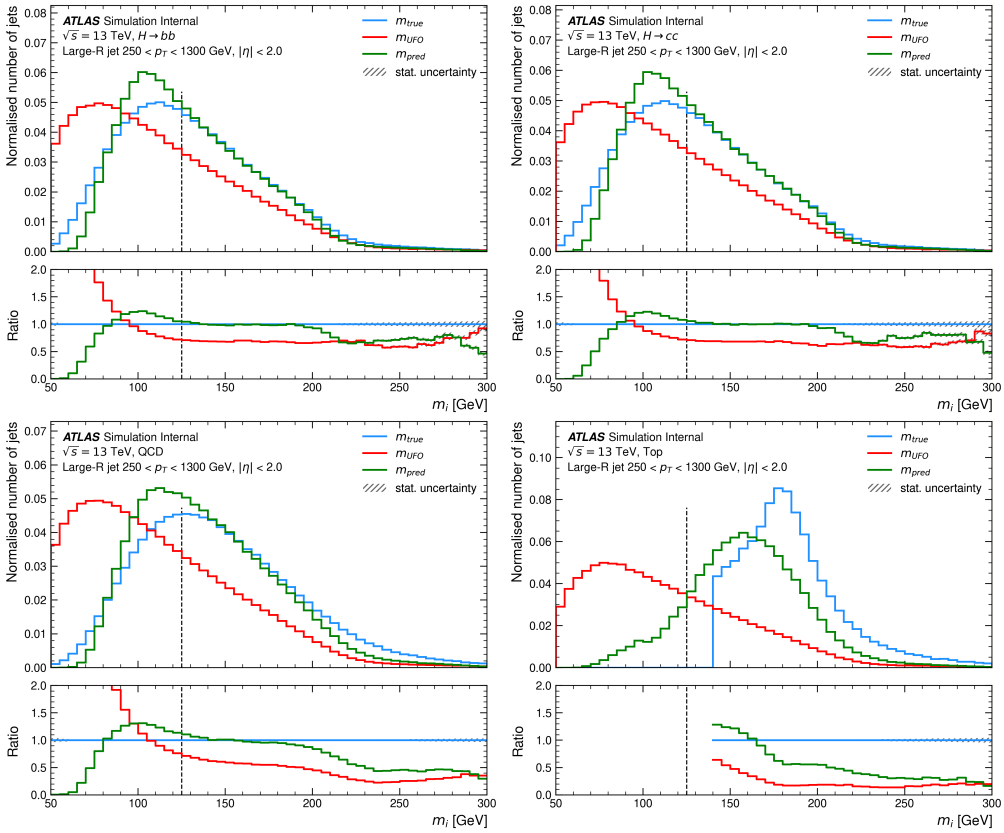


Figure A.1: The reco mass m_{UFO}, the true jet mass m_{true} and predicted mass m_{pred} distributions from the model trained on the true jet mass on $H \to b\bar{b}$, $H \to c\bar{c}$, QCD and top processes of the test samples.

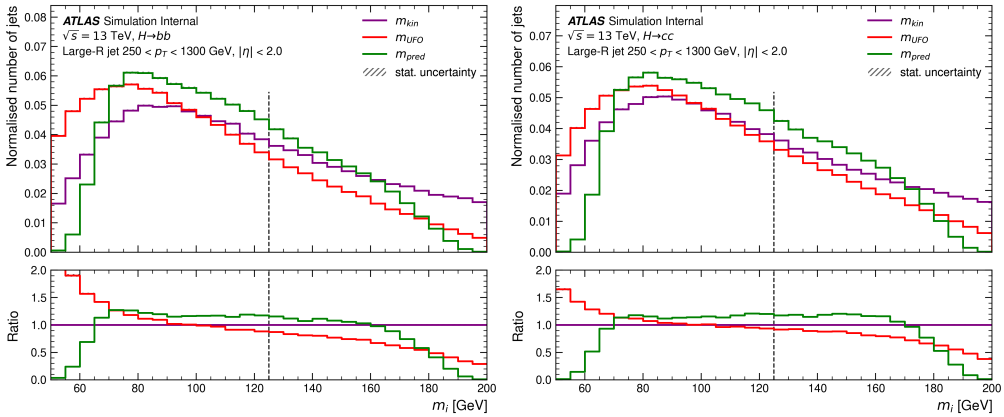


Figure A.2: The reco mass m_{UFO}, the true jet mass m_{true} and predicted mass m_{pred} distributions from the model trained on the true jet mass on $H \to b\bar{b}$ and $H \to c\bar{c}$ processes of the test samples.

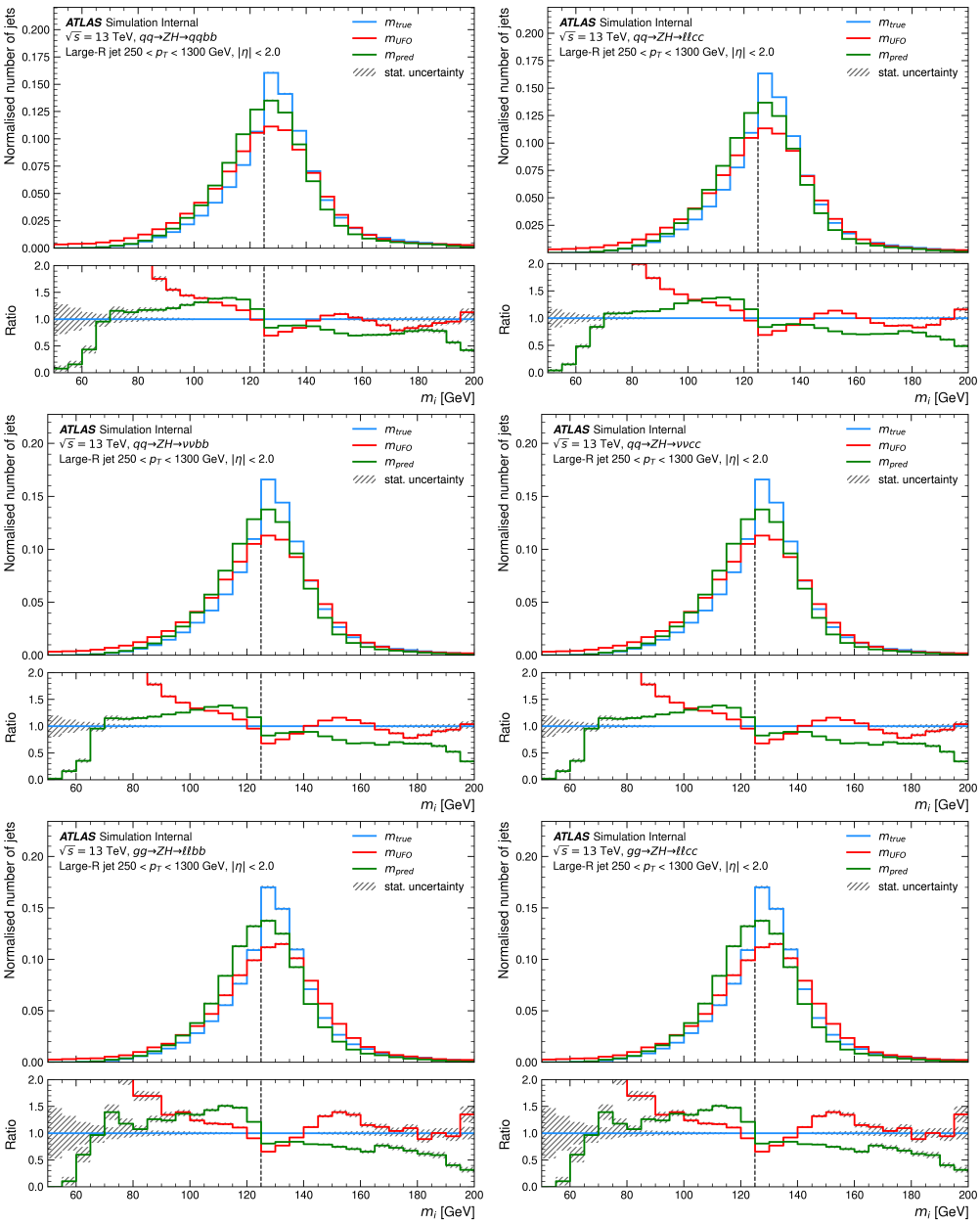


Figure A.3: The reco mass m_{UFO}, the true jet mass m_{true} and predicted mass m_{pred} distributions from the model trained on the true jet mass on different SM signal processes. The kinematic mass m_{kin} distribution is represented by the dashed line. Only reco masses of $50 < m_{\mathrm{UFO}} < 200$ GeV are used.

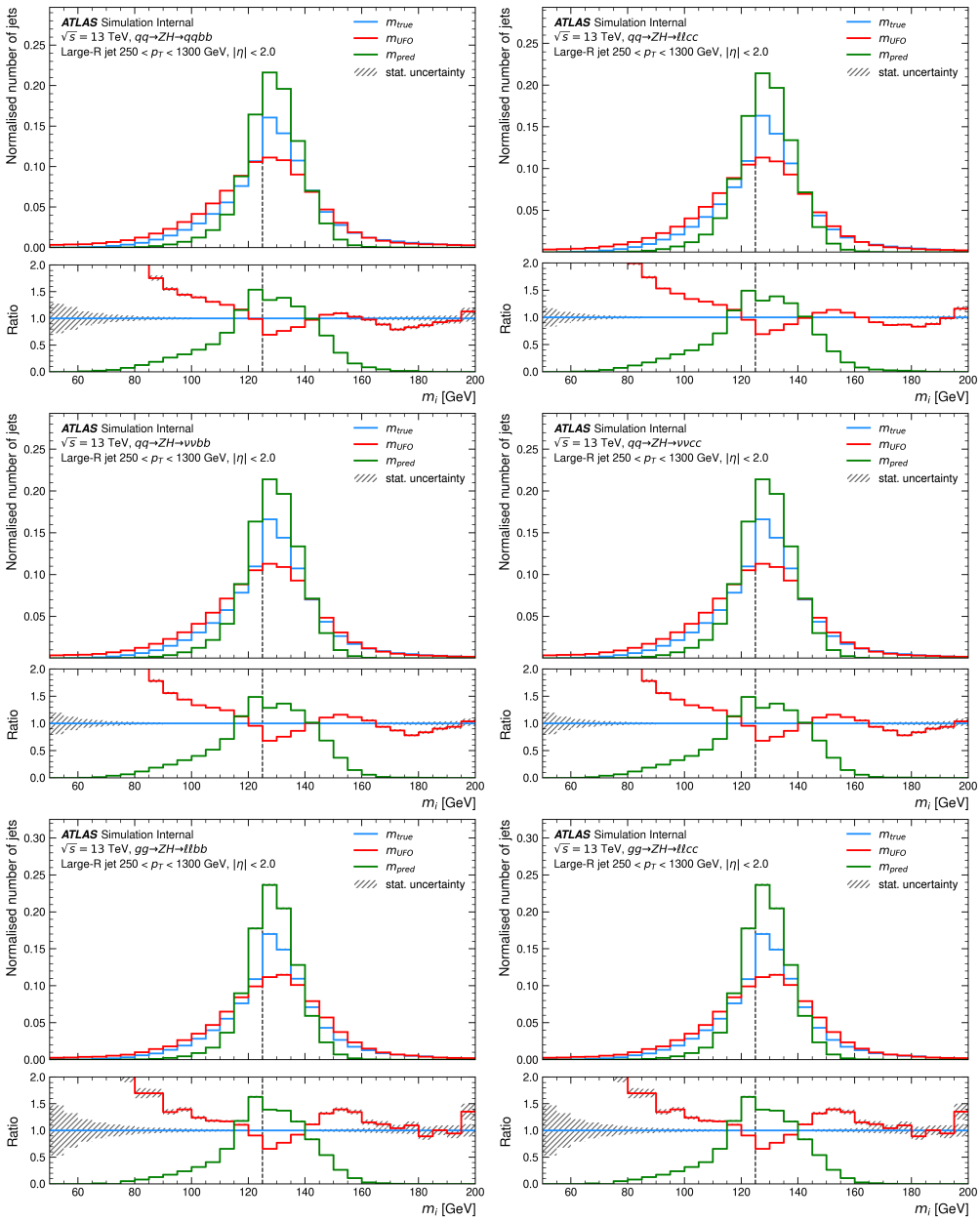
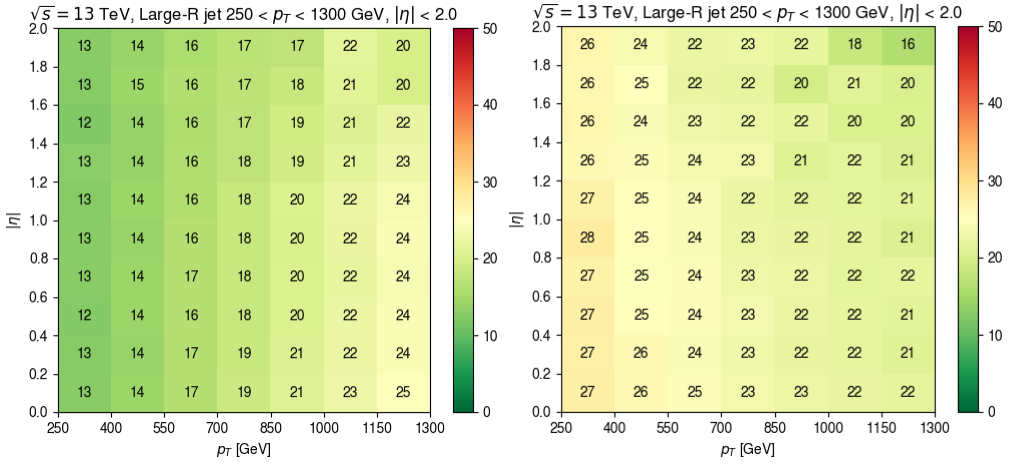
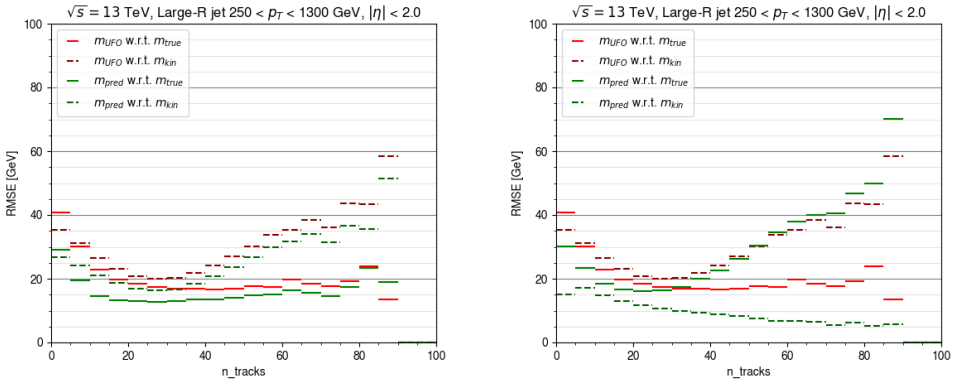


Figure A.4: The reco mass m_{UFO}, the true jet mass m_{true} and predicted mass m_{pred} distributions from the model trained on the kinematic mass on different SM processes. The kinematic mass m_{kin} distribution is represented by the dashed line. Only reco masses of $50 < m_{\text{UFO}} < 200$ GeV are used.



(a) **(b)**

Figure A.5: The RMSE of the predicted mass from the model trained on (a) the true jet mass and (b) the kinematic mass per p_T and $|\eta|$ bin, using the test sample.



(a) **(b)**

Figure A.6: The RMSE of the reco and predicted mass w.r.t. both the true jet mass and the kinematic mass, where the predictions are from the model trained on (a) the true jet mass and (b) the kinematic mass, vs. the number of associated tracks, using the SM sample. Only reco masses of $50 < m_{\mathrm{UFO}} < 200$ GeV are used.

References

[1] Peter W. Higgs. Broken symmetries, massless particles and gauge fields. *Phys. Lett.*, 12:132–133, 1964.

[2] The ATLAS Collaboartion. Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC. *Physics Letters B*, 716(1):1–29, 2012.

[3] The CMS Collaboartion. Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC. *Physics Letters B*, 716(1):30–61, 2012.

[4] The Muon $g-2$ Collaboration. Measurement of the Positive Muon Anomalous Magnetic Moment to 0.46 ppm. *Phys. Rev. Lett.*, 126:141801, Apr 2021.

[5] V. Barger, D. Marfatia, and K. Whisnant. *The Physics of Neutrinos*. Princeton University Press, 2012.

[6] NASA. Dark Energy, Dark Matter. `https://science.nasa.gov/astrophysics/focus-areas/what-is-dark-energy`. [Online accessed: 19-10-2023].

[7] Michelangelo L. Mangano, Giacomo Ortona, and Michele Selvaggi. Measuring the Higgs self-coupling via Higgs-pair production at a 100 TeV p–p collider. *The European Physical Journal C*, 80(11):1030, Nov 2020.

[8] The ATLAS Collaboartion. Combined measurement of the Higgs boson mass from the $H \to \gamma\gamma$ and $H \to ZZ^* \to 4\ell$ decay channels with the ATLAS detector using $\sqrt{s} = 7$, 8 and 13 TeV pp collision data. Technical report, CERN, Geneva, 2023.

[9] The ATLAS Collaboartion. Measurements of the higgs boson production and decay rates and coupling strengths using pp collision data at $\sqrt{s} = 7$ and 8 TeV in the ATLAS experiment. *The European Physical Journal C*, 76(1), jan 2016.

[10] Cush MissMJ. Standard model of elementary particles. `https://en.wikipedia.org/wiki/File:Standard_Model_of_Elementary_Particles.svg`, September 2019. [Online accessed: 07-12-2022].

[11] Writers of Wikipedia. B meson. `https://en.wikipedia.org/wiki/B_meson`, June 2023. [Online accessed: 25-09-2023].

[12] Luc Builtjes. Optimising Neural Networks for Classification of tttt-Events in the LHC. `https://www.ru.nl/publish/pages/913395/luc_builtjes_master_thesis_final.pdf`, August 2022. [Online accessed: 25-09-2023].

[13] The ATLAS Collaboration et al. The ATLAS Experiment at the CERN Large Hadron Collider. *Journal of Instrumentation*, 3(08):S08003, aug 2008.

[14] The ATLAS Collaboartion. Boosted Higgs ($\to b\bar{b}$) Boson Identification with the ATLAS Detector at $\sqrt{s} = 13$ TeV. Technical report, CERN, Geneva, 2016.

[15] The ATLAS Collaboartion. Transformer Neural Networks for Identifying Boosted Higgs Bosons decaying into $b\bar{b}$ and $c\bar{c}$ in ATLAS. Technical report, CERN, Geneva, 2023.

[16] Dao Valerio. Hbb/cc task force: discussion on available MC. `https://indico.cern.ch/event/1230970/contributions/5186455/attachments/2569598/4430655/FTAG_2022-11-19_TaskForceMC.pdf`, dec 2022. [Online accessed: 31-10-2023].

[17] Torbjörn Sjöstrand, Stephen Mrenna, and Peter Skands. A brief introduction to PYTHIA 8.1. *Computer Physics Communications*, 178(11):852–867, jun 2008.

[18] The ATLAS Collaboration. ATLAS Pythia 8 tunes to 7 TeV data. In *6th International Workshop on Multiple Partonic Interactions at the LHC*, page 29, 11 2014.

[19] Richard D. Ball, Valerio Bertone, Stefano Carrazza, Christopher S. Deans, Luigi Del Debbio, Stefano Forte, Alberto Guffanti, Nathan P. Hartland, José I. Latorre, Juan Rojo, and Maria Ubiali. Parton distributions with LHC data. *Nuclear Physics B*, 867(2):244–289, feb 2013.

[20] Stefano Frixione, Paolo Nason, and Carlo Oleari. Matching NLO QCD computations with parton shower simulations: the POWHEG method. *Journal of High Energy Physics*, 2007(11):070–070, nov 2007.

[21] The ATLAS Colaboration. Measurement of the Z/γ^* boson transverse momentum distribution in pp collisions at $\sqrt{s} = 7$ TeV with the ATLAS detector. *Journal of High Energy Physics*, 2014(9), sep 2014.

[22] David J. Lange. The EvtGen particle decay simulation package. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 462(1):152–155, 2001. BEAUTY2000, Proceedings of the 7th Int. Conf. on B-Physics at Hadron Machines.

[23] The ATLAS Colaboration. The ATLAS simulation infrastructure. *The European Physical Journal C*, 70(3):823–874, sep 2010.

[24] Matteo Cacciari, Gavin P. Salam, and Gregory Soyez. FastJet user manual. *The European Physical Journal C*, 72(3), mar 2012.

[25] Matteo Cacciari, Gavin P Salam, and Gregory Soyez. The anti-kt jet clustering algorithm. *Journal of High Energy Physics*, 2008(04):063–063, apr 2008.

[26] Galetsky Vladlen. Machine learning methods to improve boosted Higgs boson tagging at ATLAS, oct 2020. [Online accessed: 22-10-2023].

[27] ATLAS Collaboration. Optimisation of large-radius jet reconstruction for the ATLAS detector in 13 TeV proton–proton collisions. *Eur. Phys. J. C*, 81(4):334, 2021.

[28] Andrew J. Larkoski, Simone Marzani, Gregory Soyez, and Jesse Thaler. Soft drop. *Journal of High Energy Physics*, 2014(5), may 2014.

[29] Peter Berta, Martin Spousta, David W. Miller, and Rupert Leitner. Particle-level pileup subtraction for jets and jet shapes. *Journal of High Energy Physics*, 2014(6), jun 2014.

[30] Matteo Cacciari, Gavin P. Salam, and Gregory Soyez. SoftKiller, a particle-level pileup removal method. *The European Physical Journal C*, 75(2), feb 2015.

[31] The CMS Collaboration. Jet substructure. https://cms-opendata-workshop.github.io/workshop2022-lesson-physics-objects/06-substructure/index.html. [Online accessed: 26-07-2023].

[32] David Krohn, Jesse Thaler, and Lian-Tao Wang. Jets with variable R. *Journal of High Energy Physics*, 2009(06):059, jun 2009.

[33] The ATLAS Collaboration. ATLAS measurements of the properties of jets for boosted particle searches. *Physical Review D*, 86(7), oct 2012.

[34] CERN. CERN Yellow Reports: Monographs, Vol 2 (2017): Handbook of LHC Higgs cross sections: 4. Deciphering the nature of the Higgs sector, 2017.

[35] Jesse Thaler and Ken Van Tilburg. Identifying boosted objects with N-subjettiness. *Journal of High Energy Physics*, 2011(3), mar 2011.

[36] Thijs Miedema. Using multivariate analysis to improve boosted higgs to $b\bar{b}$ mass resolution. https://www.ru.nl/publish/pages/913454/170708_using_multivariate_analysis_to_improve_boosted_higgs_to_b-bar_b_mass_resolution_thijs_miedema.pdf, July 2017. [Online accessed: 27-09-2023].

[37] Hao Chen, Ming-Xing Luo, Ian Moult, Tong-Zhi Yang, Xiaoyuan Zhang, and Hua Xing Zhu. Three point energy correlators in the collinear limit: symmetries, dualities and analytic results. *Journal of High Energy Physics*, 2020(8), aug 2020.

[38] Andrew J. Larkoski, Gavin P. Salam, and Jesse Thaler. Energy correlation functions for jet substructure. *Journal of High Energy Physics*, 2013(6), jun 2013.

[39] Ian Moult, Lina Necib, and Jesse Thaler. New angles on energy correlation functions. *Journal of High Energy Physics*, 2016(12), dec 2016.

[40] Shan-Liang Zhang, Meng-Quan Yang, and Ben-Wei Zhang. Parton splitting scales of reclustered large-radius jets in high-energy nuclear collisions. *The European Physical Journal C*, 82(5), may 2022.

[41] S Catani, Yu L Dokshitzer, Michael H Seymour, and Bryan R Webber. Longitudinally-invariant $k_\perp$-clustering algorithms for hadron-hadron collisions. *Nucl. Phys. B*, 406:187–224, 1993.

[42] Leandro G. Almeida, Seung J. Lee, Gilad Perez, Ilmo Sung, and Joseph Virzi. Top quark jets at the LHC. *Phys. Rev. D*, 79:074012, Apr 2009.

[43] FTAG Dumpster documentation. PFlow Jets Variable. https://training-dataset-dumper.docs.cern.ch/vars_pflow/. [Online accessed: 11-07-2023].

[44] ATLAS Software Documentation. ATLAS Track Reconstruction – General Overview. https://atlassoftwaredocs.web.cern.ch/trackingTutorial/idoverview/, July 2023. [Online accessed: 25-07-2023].

[45] Writers of Wikipedia. Perceptron. https://en.wikipedia.org/wiki/Perceptron, July 2023. [Online accessed: 17-08-2023].

[46] Writers of Wikipedia. Rectifier (neural networks). https://en.wikipedia.org/wiki/Rectifier_(neural_networks)#Softplus, July 2023. [Online accessed: 17-08-2023].

[47] The FTAG group - ATLAS. GN2X. https://gitlab.cern.ch/atlas-flavor-tagging-tools/algorithms/salt/-/blob/main/salt/configs/GN2X.yaml?ref_type=heads, August 2023. [Online accessed: 17-08-2023].

[48] Vaclav Kosar. Cross-Attention in Transformer Architecture. https://vaclavkosar.com/ml/cross-attention-in-transformer-architecture, Dec 2021. [Online accessed: 14-01-2023].

[49] Stefan Rohrmanstorfer, Mikhail Komarov, and Felix Moedritscher. Image Classification for the Automatic Feature Extraction in Human Worn Fashion Data. *Mathematics*, 9:624, 03 2021.

[50] Writers of Wikipedia. Huber loss. https://en.wikipedia.org/w/index.php?title=Huber_loss&action=history, July 2023. [Online accessed: 03-10-2023].

[51] B. D. Hammel. What learning rate should I use? http://www.bdhammel.com/learning-rates/, March 2019. [Online accessed: 21-08-2023].

[52] Sylvain Gugger and Jeremy Howard. AdamW and Super-convergence is now the fastest way to train neural nets. https://www.fast.ai/posts/2018-07-02-adam-weight-decay.html, July 2018. [Online accessed: 21-08-2023].

[53] Writers of Wikipedia. Backpropagation. https://en.wikipedia.org/wiki/Backpropagation, September 2023. [Online accessed: 23-09-2023].

[54] Pytorch Documentation. Onecyclelr. https://pytorch.org/docs/stable/generated/torch.optim.lr_scheduler.OneCycleLR.html. [Online accessed: 22-08-2023].

[55] Geoffrey Gilles. Multi-head attention based classifier. https://indico.cern.ch/event/1182926/contributions/5029971/attachments/2500819/4295885/MultiHeadAttention.pdf, Sep 2022. [Online accessed: 14-01-2023].

[56] Phillip Lippe. Tutorial 6: Transformers and Multi-Head Attention. https://uvadlc-notebooks.readthedocs.io/en/latest/tutorial_notebooks/tutorial6/Transformers_and_MHAttention.html, 2022. [Online accessed: 09-01-2023].

[57] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *CoRR*, abs/1706.03762, 2017.

[58] Lilian Weng. Multi-Head Attention. https://paperswithcode.com/method/multi-head-attention#:~:text=Multi%2Dhead%20Attention%20is%20a,transformed%20into%20the%20expected%20dimension. [Online accessed: 09-01-2023].

[59] Pytorch Documentation. Layernorm. https://pytorch.org/docs/stable/generated/torch.nn.LayerNorm.html. [Online accessed: 22-08-2023].

[60] Sam Shleifer, Jason Weston, and Myle Ott. Normformer: Improved transformer pretraining with extra normalization. *CoRR*, abs/2110.09456, 2021.

[61] The FTAG group - ATLAS. Salt. https://gitlab.cern.ch/atlas-flavor-tagging-tools/algorithms/salt, August 2023. [Online accessed: 17-08-2023].

[62] Johannes Wagner. GN2X + Lepton ID and CMS comparisons. https://indico.cern.ch/event/1313256/contributions/5585840/attachments/2716474/4718306/jmwagner_230918_xbb.pdf, September 2023. [Online accessed: 20-09-2023].

[63] Josu Cantero. Generating flat mass W boson samples. https://indico.cern.ch/event/1220527/contributions/5135936/attachments/2545462/4383376/JetTagging_Nov10.pdf, nov 2022. [Online accessed: 31-10-2023].