Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Cybernetics

Optimization of the $t\bar{t}H$ Selection Including Systematics Using Machine Learning with ATLAS Data

Master's thesis

Bc. Vladyslav Yazykov

Study Programme: Cybernetics and Robotics
Supervisor: doc. Dr. André Sopczak

Prague, August 2023

Thesis Supervisor:
doc. Dr. André Sopczak
Institute of Experimental and Applied Physics
Czech Technical University in Prague
Husova 240/5
110 00 Prague 1
Czech Republic

Copyright © August 2023 Bc. Vladyslav Yazykov

MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Yazykov Vladyslav** Personal ID number: **452777**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Cybernetics and Robotics**

II. Master’s thesis details

Master’s thesis title in English:

Optimization of the ttH Selection Including Systematics Using Machine Learning with ATLAS Data

Master’s thesis title in Czech:

Optimalizace výběru ttH včetně systematiky pomocí strojového učení s z dat ATLASu

Guidelines:

The neutral Standard Model Higgs boson was discovered in 2012 at CERN. Currently, the focus of research is on the precision measurements of the Higgs boson properties. For this purpose, the Higgs boson sample shall be separated from unwanted background reactions. There are three analysis levels: generator level, full ATLAS detector simulation, and real recorded data. In this project, the simulated data from the full ATLAS detector simulation shall be used and the performance of the machine learning algorithms be optimized in the separation of ttH from unwanted background events.
Instructions:
1) Familiarize yourself with basic concepts of elementary particle searches in particle physics.
2) Study the provided ROOT framework, the ntuple data storage format, and existing analysis tools.
3) Get familiar with the implementation of a classifier to separate events of interest from the background, using high-level features and classical machine-learning techniques.
4) Design and implement a deep-learning based classifier to separate signal and background events based on simulated data.
5) Study the feature importance for the selection and implement systematic uncertainties of the features for the optimization of the signal sensitivity.
6) Evaluate its performance on simulated data and compare with existing approaches.
7) Apply the analysis on recorded data and determine the uncertainty of the ttH measurement.

Bibliography / sources:

[1] https://home.cern
[2] https://atlas.cern
[3] ATLAS Collaboration: Observation of Higgs boson production in association with a top quark pair at the LHC with the ATLAS detector. Physics Letters B, vol. 784, pp. 173-191, 2018
[4] Heather Gray, Bruno Mansoulié, The Higgs boson: the hunt, the discovery, the study and some future perspectives, https://atlas.cern/updates/feature/higgs-boson
[5] Guest et al., Deep Learning and Its Application to LHC Physics article in Annu. Rev. Nucl. Part. Sci. 2018. 68:1–22 https://arxiv.org/pdf/1806.11484.pdf
[6] M. Andrews et al., End-to-End Event Classification of High-Energy Physics Data http://www.sergeigleyzer.com/wp-content/uploads/2017/12/end-end-event.pdf

© ČVUT v Praze, Design: ČVUT v Praze, VIC

Name and workplace of master's thesis supervisor:

doc. Dr. André Sopczak High Energy Physics, IEAP CTU in Prague

Name and workplace of second master's thesis supervisor or consultant:

Date of master's thesis assignment: **17.10.2022** Deadline for master's thesis submission: **14.08.2023**

Assignment valid until: **22.09.2024**

doc. Dr. André Sopczak	prof. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor's signature	Head of department's signature	Dean's signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

. Date of assignment receipt

Student's signature

© ČVUT v Praze, Design: ČVUT v Praze, VIC

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, 14.08.2023 .....

Bc. Vladyslav Yazykov

Abstrakt

Tento výzkum se soustředí na cíl přesně vybrat události vzniklé interakcí dvou gluonů při srážce protonu s protonem na urychlovači LHC, jejímž výsledkem je pár top-antitop a Higgsův boson, což je proces známý jako $t\bar{t}H$.

Cílem studie je pomocí dat zaznamenaných detektorem odlišit tyto události $t\bar{t}H$ od událostí generovaných jinými procesy. Za tímto účelem využíváme pro výběr událostí přístup hlubokého učení, konkrétně architekturu FT-Transformer.

Použití takové metody strojového učení zvyšuje naši schopnost přesně identifikovat události $t\bar{t}H$, což vede ke zlepšení poměru signálu k šumu a statistické významnosti, a tím přispívá k našemu pochopení vlastností Higgsova bosonu. Kromě zkoumání pokročilejších architektur NN vylepšujeme předchozí práci tím, že zkoumáme použití rozšířené trénovací množiny, což nám umožňuje výrazně zvýšit trénovací statistiku a dosáhnout tak mnohem lepšího výkonu.

Nedílnou součástí tohoto výzkumu je vyhodnocení statistických i systematických nejistot spojených s tímto procesem výběru událostí. Zjištění a metodiky prezentované v této práci nabízejí slibný pokrok v oblasti výběru událostí částicové fyziky a přispívají k pokračujícímu úsilí kolaborace o zkoumání základních vlastností vesmíru.

Keywords: CERN, ATLAS, Higgsův boson, Strojové učení, Neuronové Sítě

Abstract

This research is centered on the goal of accurately selecting events produced from the interaction of two gluons in a proton-proton collision at the LHC, resulting in a top-antitop pair and a Higgs boson, a process known as $t\bar{t}H$.

With data recorded by the detector, the study aims to distinguish these $t\bar{t}H$ events from those generated by other processes. To this end, we employ a deep learning approach, specifically a FT-Transformer architecture, for the event selection. The use of such a machine learning method enhances our ability to identify $t\bar{t}H$ events accurately, leading to an improved signal-to-noise ratio and statistical significance, thereby contributing to our understanding of the Higgs boson's properties. Aside of exploring more advanced NN architectures, we improve previous work, by exploring the use of an extended training set, which allows us to dramatically increase the training statistic and thus achieve a much better performance.

An integral part of this research is the evaluation of both statistical and systematic uncertainties associated with this event selection process. The findings and methodologies presented in this thesis offer promising advancements in particle physics event selection, contributing to the Collaboration's ongoing endeavors to probe the fundamental properties of the universe.

Keywords: CERN, ATLAS, Higgs boson, Machine Learning, Neural Networks

Acknowledgements

I wish to express my profound gratitude to the following individuals, whose invaluable contributions were indispensable in the realization of this thesis:

Firstly, I would like to offer my deepest thanks to my supervisor, doc. Dr. André Sopczak. His consistent guidance, valuable insights, and steadfast support were integral throughout the course of this research.

My sincere appreciation extends to the remarkable CERN community. Their relentless support, coupled with their readiness to offer assistance, helped me overcome many challenges. My experience as a part of this esteemed organization has been life-changing, and I am incredibly grateful for the opportunity.

I owe a debt of gratitude to Luca Martinelli, Martin Vatrt, and Nihal Brahimi. Their mentorship and patient elucidation of complex physical and technical principles have proven invaluable. Their generosity in sharing their expertise is deeply appreciated, as are the numerous hours they dedicated to answering my relentless inquiries.

My heartfelt thanks go to my dear friend, Andrii, whose expertise in machine learning was instrumental in resolving numerous complex issues. His patience, assistance and eagerness to aid me were indispensable in implementing advanced techniques for this research.

To my beloved partner, Anna, I express my eternal gratitude. Her unwavering support, understanding, and encouragement have been my constant source of strength and inspiration during this journey.

Lastly, my deepest appreciation goes to all my friends and family. Their faith in my capabilities and their constant encouragement have fueled my pursuit of this academic endeavor.

Each person named above has played an instrumental role in shaping this thesis, and I am profoundly grateful for their influence in my life. Their collective support has been a beacon throughout this challenging yet rewarding academic journey. My sincerest thanks to you all.

Contents

List of Figures

List of Tables

Glossary

AdamW Adaptive Moment Estimation with Weight Decay.

ATLAS A Toroidal LHC ApparatuS.

AUC Area Under the Curve.

BDT Boosted Decision Tree.

CERN European Organization for Nuclear Research.

CL Confidence Level.

CLT Central Limit Theorem.

CMS Compact Muon Solenoid.

CR Control Region.

DL Deep Learning.

ERM Empirical Risk Minimization.

FT-Transformer **F**eature **T**okenizer + **Transformer**.

GPU Graphics Processing Unit.

HEP high energy physics.

i.i.d. independent and identically distributed.

IG Integrated Gradients.

LHC Large Hadron Collider.

MC Monte Carlo.

MLE Maximum Likelihood Estimator.

MLP Multi-Layer Perceptron.

NLP Natural Language Processing.

NN Neural Network.

NP nuissance parameter.

ResNet Residual Neural Network.

RNN Recurrent Neural Network.

ROC Receiver Operating Characteristic.

SGD Stochastic Gradient Descent.

SM Standard Model.

SOTA State-of-the-Art.

SR Signal Region.

VR Validation Region.

Structure of the thesis

We begin with a short introduction to the fundamentals of particle physics and the Standard Model (SM) in the **Chapter** 1, explaining key terms such as Feynman diagrams, branching fractions, production cross-sections, particle interactions, and decays.

Thereafter, we provide a brief overview of the Large Hadron Collider (LHC) and the detector, enabling the readers to comprehend their operational mechanics, data collection techniques, and the subsequent analysis involved.

We then proceed to explain the process of interest - the production of a top-antitop pair and a Higgs boson from two gluons in a proton-proton collision ($t\bar{t}H$), which constitutes the central focus of our research - the separation of $t\bar{t}H$ events from other background processes.

This is followed by an in-depth discussion on the concept of significance in particle physics, covering its theoretical underpinnings and practical implications in event selection.

Chapter 2 outlines the machine learning techniques and strategies utilized, including problem formulation, details on the Monte Carlo (MC) simulation, performance evaluation metrics.

In **Chapter** 3, a comprehensive analysis of the statistical and systematic uncertainties is presented.

Finally, in **Conclusions** we summarize our findings, briefly discuss the implications of our research, and provide suggestions for future studies.

For further insights, the **Appendix** provides details about preselection, used features, and training large Neural Networks (NNs) efficiently.

Chapter 1

Introduction

1.1 Overview of Particle Physics

Particle physics, also known as high energy physics (HEP), is a field of study that seeks to understand the fundamental constituents of matter and their interactions. This branch of physics has led to the discovery and characterization of a multitude of subatomic particles, providing deep insights into the structure of the universe.

1.1.1 The Standard Model

The framework that describes elementary particles and their interactions is known as the Standard Model (SM) of particle physics[1]. The SM consists of two types of elementary particles: fermions and bosons. Fermions are particles with half-integer spin and include quarks and leptons, while bosons are particles with integer spin and act as force carriers in the SM.

Quarks come in six different flavors: up, down, charm, strange, top, and bottom, and interact through the strong nuclear force. Quarks combine in various ways to form hadrons, such as protons and neutrons.

Leptons, on the other hand, are particles that do not interact via the strong nuclear force. They include electrons, muons, taus, and their corresponding neutrinos. Each lepton flavor is associated with a specific neutrino.

Bosons are integral to the fundamental forces of the universe. The photon mediates the electromagnetic force, the W and Z bosons mediate the weak nuclear force, and the gluon mediates the strong nuclear force.

[1] `https://home.cern/science/physics/standard-model`

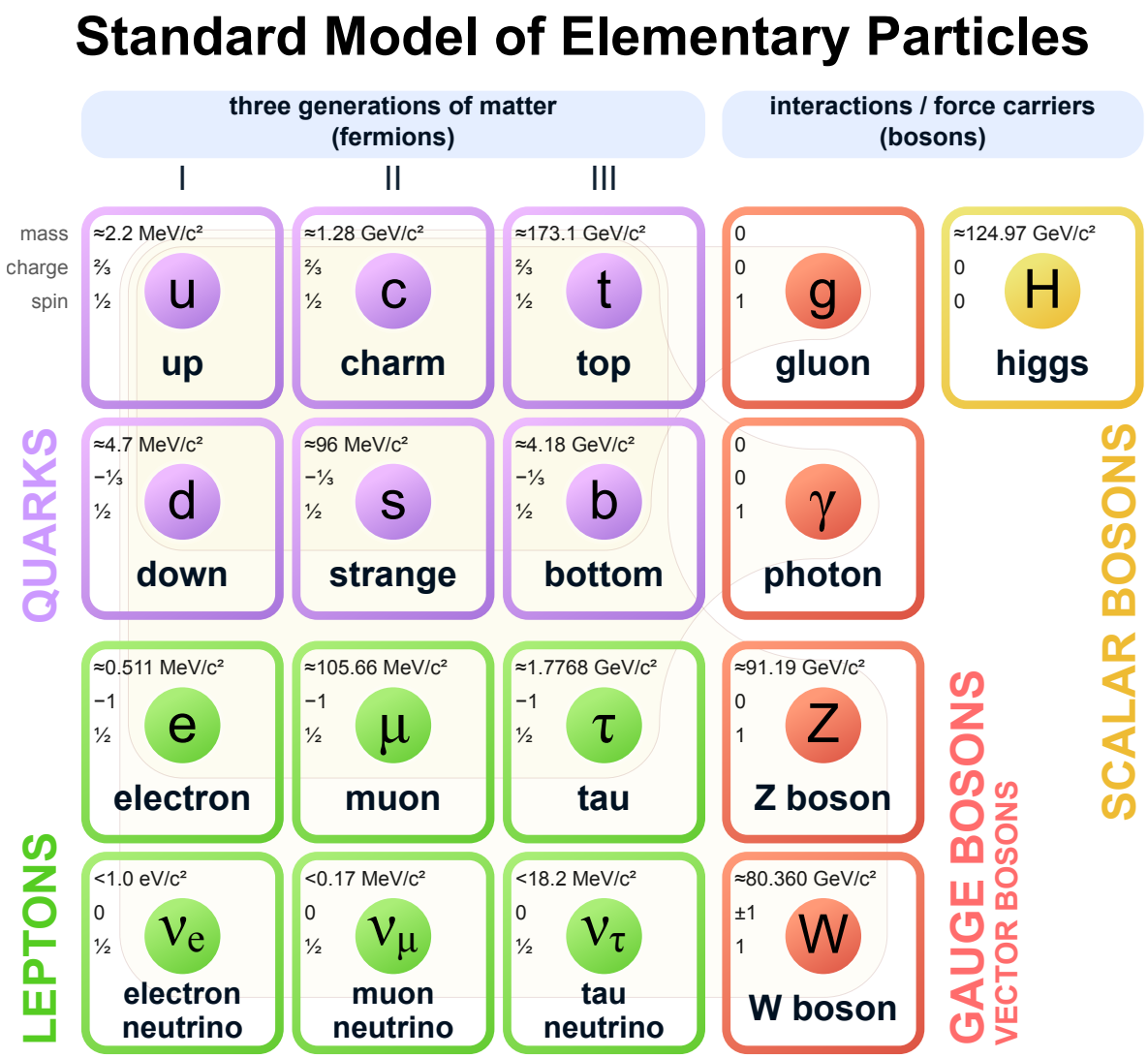


Figure 1.1: Particles and forces of the Standard Model (SM).

1.1.2 The Higgs Boson

One additional particle that plays a significant role in the SM is the Higgs boson. Unlike the other bosons, which mediate the fundamental forces, the Higgs boson is associated with the Higgs field, a scalar field that permeates all space. The Higgs boson was proposed in 1964 by physicist Peter Higgs and others as a result of their work on the so-called "Higgs mechanism" [1].

The Higgs mechanism is responsible for the mass of elementary particles. According to this theory, particles acquire mass by interacting with the Higgs field. The more strongly a particle interacts with this field, the greater its mass. Particles that do not interact with the Higgs field, such as photons, are massless.

The existence of the Higgs boson and the Higgs field was confirmed experimentally in 2012 by the ATLAS and Compact Muon Solenoid (CMS) collaborations at European Organization for Nuclear Research (CERN)[2], a discovery that led to the awarding of the 2013 Nobel Prize in Physics to François Englert and Peter Higgs.

The Higgs boson itself is unstable and quickly decays into other particles. The decay channels and their corresponding probabilities, or branching ratios, are predicted by the SM. One particular decay of interest in this work is the Higgs decay into a tau lepton pair, denoted as $H \to \tau\bar{\tau}$, which has a branching ratio of approximately 6% (Table 1.1).

Decay Channel	Branching Ratio	Relative Uncertainty
$H \to \gamma\gamma$	2.28×10^{-3}	$^{+5.0\%}_{-4.9\%}$
$H \to ZZ$	2.64×10^{-2}	$^{+4.3\%}_{-4.1\%}$
$H \to W^+W^-$	2.15×10^{-1}	$^{+4.3\%}_{-4.2\%}$
$H \to \tau^+\tau^-$	6.32×10^{-2}	$^{+5.7\%}_{-5.7\%}$
$H \to b\bar{b}$	5.77×10^{-1}	$^{+3.2\%}_{-3.3\%}$
$H \to Z\gamma$	1.54×10^{-3}	$^{+9.0\%}_{-8.9\%}$
$H \to \mu^+\mu^-$	2.19×10^{-4}	$^{+6.0\%}_{-5.9\%}$

Table 1.1: Branching ratios for various Higgs boson decay channels. Taken from `http://opendata.atlas.cern/books/current/get-started/_book/the-higgs-boson.html`

1.1.3 Particle Interactions and Decays

In particle physics, interactions between particles result from the exchange of force-carrying particles, or gauge bosons. Each of the four fundamental forces (gravitational, electromagnetic, strong nuclear, and weak nuclear) has its associated bosons, which mediate these interactions.

Decays are processes by which a particle transforms into two or more other particles. These are inherently probabilistic processes, with specific probabilities associated with each possible decay path, given by the branching ratios.

1.1.4 Branching Ratios

A branching ratio (or branching fraction) in particle physics is a measure of the fraction of particles that decay via a particular decay mode with respect to the total decay modes.

[2] `https://atlas.cern/updates/feature/higgs-boson`

It is a critical parameter in understanding particle interactions, as it gives insight into the relative probabilities of various outcomes of a decay process.

For example, let us consider the production of a top-antitop pair along with a Higgs boson in a proton-proton collision, also referred to as the $t\bar{t}H$ process. The Higgs boson can decay into different particles, each with a certain branching ratio. In the case of the Higgs boson decaying into a bottom-antibottom quark pair, the branching ratio is approximately 58%, which is the highest branching ratio among all the Higgs decay channels.

However, the probability of producing a Higgs boson, along with a top-antitop pair, in a proton-proton collision is very small, estimated to be about 1% of the total cross-section (subsection 1.3.1) [3] [4]. This small percentage, coupled with the fact that the Large Hadron Collider (LHC) produces millions of collisions per second, means that a substantial amount of data must be analyzed to isolate and identify the relatively few $t\bar{t}H$ events.

The ability to isolate these rare events depends not only on the branching ratios of the Higgs boson and the top quarks, but also on our ability to accurately detect and identify the decay products. For instance, in the case of the Higgs boson decaying into a bottom-antibottom pair, we need to be able to identify these 'b-jets' among a plethora of other particles. This is a major challenge in particle physics and is a central focus of this thesis.

1.1.5 Feynman Diagrams

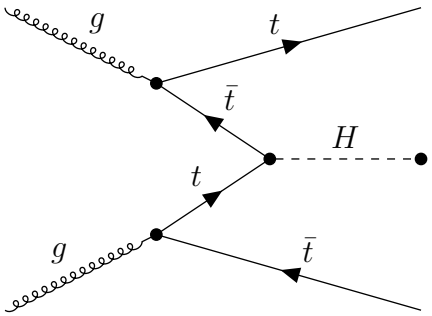


Figure 1.2: Feynman diagram of the $t\bar{t}H$ process.

A powerful tool in the field of particle physics is the Feynman diagram, named after its creator, the renowned physicist Richard P. Feynman. Feynman diagrams offer a graphical representation of the mathematical expressions describing the behavior of subatomic particles. These diagrams have become integral to predicting and understanding the results of experiments in quantum mechanics, particularly those involving subatomic particles.

[3] https://home.cern/news/news/physics/higgs-boson-comes-out-top

[4] https://home.cern/news/press-release/cern/higgs-boson-reveals-its-affinity-top-quark

In a Feynman diagram, each line represents a particle propagating through space (horizontal axis) over time (vertical axis). Straight lines depict fermions (matter particles) such as quarks and leptons, while wavy lines represent bosons (force-carrying particles), including gluons, photons, and W and Z bosons. Interaction vertices, where lines meet, indicate interactions between particles where energy and momentum are conserved.

Consider, for instance, the $t\bar{t}H$ process mentioned earlier. In a simple Feynman diagram representing this process, two incoming gluons (represented as spiral lines) interact at a vertex to produce a top-antitop pair (straight lines) and a Higgs boson (another straight line). The top and antitop quarks then decay into other particles, typically W bosons and b-quarks.

These diagrams do not just describe which particles are involved in an interaction; they can also provide insight into the probability of the interaction occurring. By calculating the areas of the regions enclosed by the lines and vertices of the diagram, physicists can predict the likelihood of an interaction. However, while Feynman diagrams can be extremely informative, they also can be complex, with multiple possible diagrams for a single interaction - another aspect that adds to the challenges faced in particle physics.

1.2 LHC and the Detector

The LHC is the world's largest and highest-energy particle accelerator. It was built by CERN between 1998 and 2008 in collaboration with over 10,000 scientists and hundreds of universities and laboratories from around the world. The LHC is located near Geneva, beneath the border of Switzerland and France. The accelerator is housed in a tunnel with a circumference of 27 kilometers, around 100 meters underground [5] [6].

The LHC's primary purpose is to accelerate protons to near-light speeds and collide them together, thereby producing extreme conditions that allow us to probe the nature of the subatomic world. The protons are grouped into "bunches" and circulated in opposite directions around the LHC ring. When these bunches cross paths, protons collide, producing energetic particle showers. The LHC is capable of generating up to a billion proton-proton collisions per second.

The ATLAS[7] detector is one of the two general-purpose detectors at the LHC. It is a large and complex system designed to measure the properties of particles produced in the proton-proton collisions. Covering an area equivalent to a five-story building, ATLAS is a

[5] https://home.cern/science/accelerators/large-hadron-collider
[6] https://home.cern/resources/faqs/facts-and-figures-about-lhc
[7] https://home.cern/science/experiments/atlas

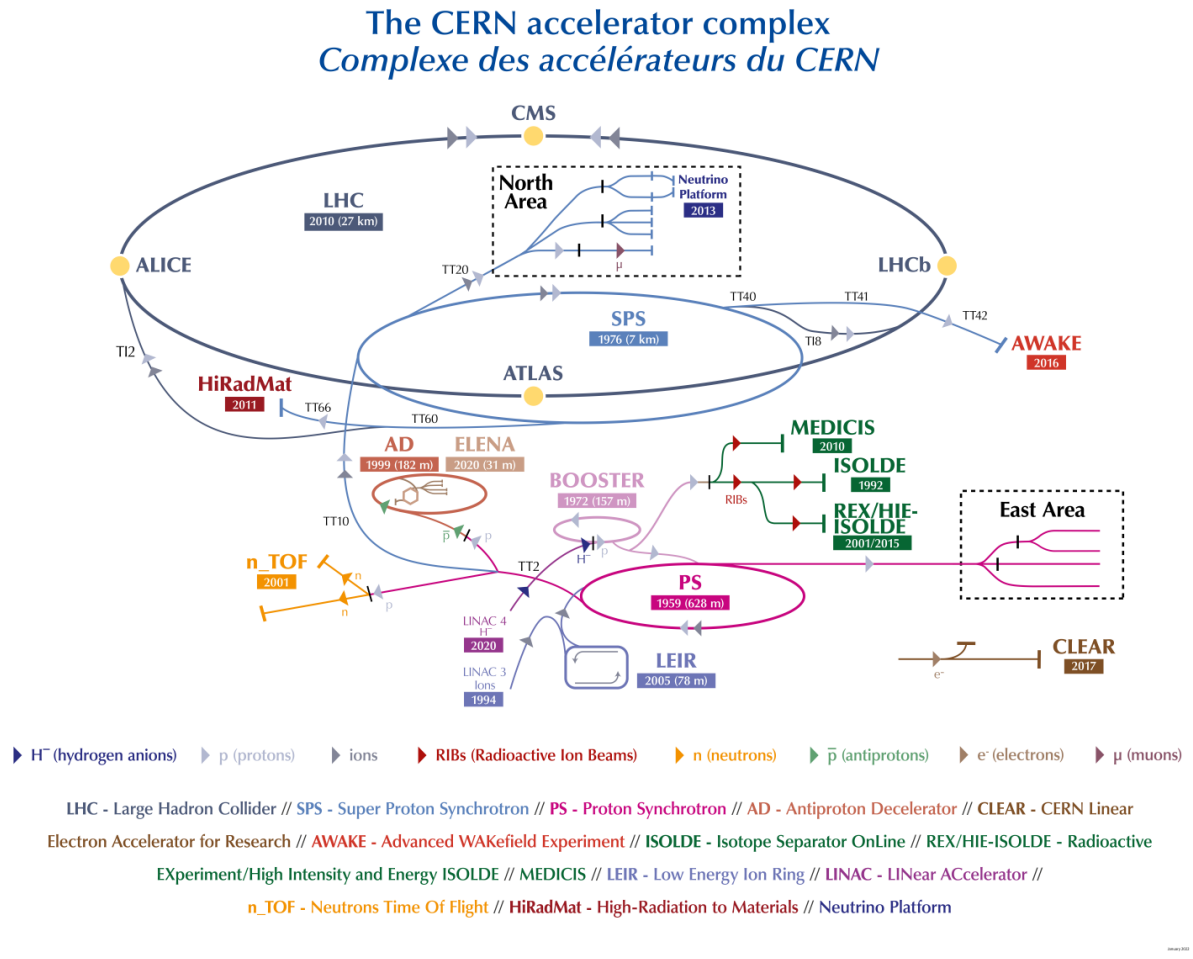


Figure 1.3: CERN with the LHC and ATLAS detector.

multi-layered device consisting of several subsystems, each optimized to measure different particle properties.

The innermost layer, or the tracking detector, records the trajectory of charged particles, allowing the reconstruction of their paths. The subsequent layers, calorimeters, measure the energy of particles. Finally, the outermost layer, the muon spectrometer, specifically designed to detect muons, one of the few particles that can penetrate all inner layers.

The data collected by ATLAS is stored and subsequently analyzed by physicists around the world. The sheer volume of this data, combined with the complexity of the collision events, presents a significant challenge, requiring sophisticated statistical methods and computational tools to make sense of the data. Machine learning techniques have been increasingly employed in recent years to classify and analyze this data, which has led to several significant breakthroughs in our understanding of particle physics, including the discovery of the Higgs boson in 2012.

1.3 $t\bar{t}H$ Process

The $t\bar{t}H$ process is a specific type of event that occurs within the LHC's high-energy particle collisions. In this process, two gluons collide, resulting in a pair of top quarks and a Higgs boson.

The $t\bar{t}H$ process is of particular interest in particle physics due to its potential to shed light on the nature of the Higgs boson. The Higgs boson is pivotal in the SM of particle physics, as it is associated with the Higgs field, which gives other particles their mass. Despite its significance, the Higgs boson remains one of the least understood particles due to its elusive nature and the difficulty involved in its detection.

In a $t\bar{t}H$ event, the Higgs boson quickly decays into other particles, making it impossible to observe it directly. Instead, physicists must reconstruct its presence from the particles into which it decays. The top quarks, on the other hand, have a higher lifetime, making them easier to detect. The $t\bar{t}H$ process is unique in that it allows the observation of the Higgs boson's interactions with top quarks - the heaviest known fundamental particle - thereby providing a direct probe into the Higgs mechanism.

The primary goal of this research is to separate $t\bar{t}H$ process events from other events, which is a challenging task given the similarities between $t\bar{t}H$ events and certain background processes. By leveraging Deep Learning (DL) techniques, this task can be achieved more effectively, thereby facilitating a more detailed study of the Higgs boson and its properties.

1.3.1 $2l_{SS} + 1\tau_{\mathbf{had}}$ channel

The work at CERN is often divided into different channels, with each channel focusing on a specific final state of interest. These channels are orthogonal, meaning that each event can only belong to one channel, preventing any overlap in the analysis.

One such channel, and the main focus of this thesis, is the two-lepton same-sign plus one tau ($2l_{SS} + 1\tau_{\text{had}}$) ($e$, μ) channel. This specific final state arises from the decay of a top-antitop-Higgs ($t\bar{t}H$) system, where one top quark decays to a W boson and a b quark, with the W boson further decaying to a lepton and a neutrino. The other top quark also decays to a W boson and a b quark, but in this case, the W boson decays into a pair of quarks. Lastly, the Higgs boson decays to a pair of tau leptons, where one tau lepton decays into another lepton and two neutrinos, while the other tau lepton decays to a pair of quarks and a neutrino. This complex series of decays results in a final state consisting of two same-sign leptons and a hadronically decaying tau lepton, hence the

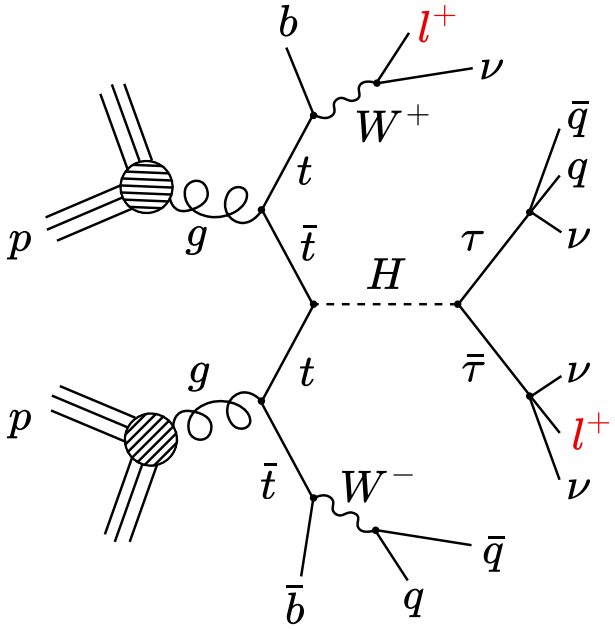


Figure 1.4: Feynman diagram of the $2l_{SS} + 1\tau_{\text{had}}$ process.

name $2l_{SS} + 1\tau_{\text{had}}$ channel (Figure 1.4).

The branching ratios for these decays are approximately

$$\begin{aligned}
\text{Higgs boson to a pair of tau leptons: } & \mathcal{B}(H \to \tau\tau) \approx 6.0\% \\
\text{Tau lepton to a } W \text{ boson and a neutrino: } & \mathcal{B}(\tau \to W\nu) \approx 100\% \\
\text{Top quark to a } W \text{ boson and a b quark: } & \mathcal{B}(t \to Wb) \approx 100\% \\
W \text{ boson to an electron and a neutrino: } & \mathcal{B}(W \to e\nu) \approx 11\% \\
W \text{ boson to a muon and a neutrino: } & \mathcal{B}(W \to \mu\nu) \approx 11\% \\
W \text{ boson to a pair of quarks: } & \mathcal{B}(W \to qq) \approx 67\% \,.
\end{aligned}$$

Given that the theoretical cross-section for the $t\bar{t}H$ process is about $\sigma_{t\bar{t}H} \approx 500$ fb. The luminosity for the analyzed period (LHC Run-2) is $\mathcal{L} = 140$ fb^{-1}, the total number of expected $t\bar{t}H$ events is approximately $N_{t\bar{t}H} = \sigma_{t\bar{t}H} \cdot \mathcal{L} = 70\ 000$. When accounting for the aforementioned branching ratios, the expected number of $t\bar{t}H$ events in the $2l_{SS} + 1\tau_{\text{had}}$ final state is approximately reduced to

$$\begin{aligned}
N_{t\bar{t}H,\, 2l_{SS}+1\tau_{\text{had}}} &= 2 \cdot N_{t\bar{t}H} \cdot \mathcal{B}(H \to \tau\tau) \cdot (\mathcal{B}(W \to e\nu) + \mathcal{B}(W \to \mu\nu))^2 \cdot \mathcal{B}(W \to qq)^2 \\
&= 2 \cdot 70\ 000 \cdot 0.06 \cdot (0.11 + 0.11)^2 \cdot 0.67^2 \approx 183 \,.
\end{aligned}$$

Note that factor 2 takes into account the two configurations where leptons have either both positive sign, or both negative sign. The first (positive) case is presented in Figure 1.4, while the second one would have the top quark decay into $t \to b\bar{q}q$, antitop quark decay into $\bar{t} \to \bar{b}l^{-}\nu$, and then from the Higgs, $\bar{\tau} \to \bar{q}q\nu$, and $\tau \to l^{-}\nu\nu$.

It is a simplified calculation, but it provides a good estimate of the scale of the challenge we face in detecting the $t\bar{t}H$ in the $2l_{SS}+1\tau_{\text{had}}$ channel. As we apply the selection criteria to the simulated data (section 1.6), we drop from about 8.9M to just 32K raw events. For the $t\bar{t}H$, this corresponds to 0.8M $\to$ 15K raw events. Note that additional cuts (subsection 1.3.2) are applied, numbers here correspond to the SR. In terms of weighted events the transition is from 46K $\to$ 32.72 weighted events across all the processes, and 523.42 $\to$ 12.22 weighted events for the $t\bar{t}H$. The details for all processes are given in Table 1.2.

1.3.2 Regions in Particle Physics

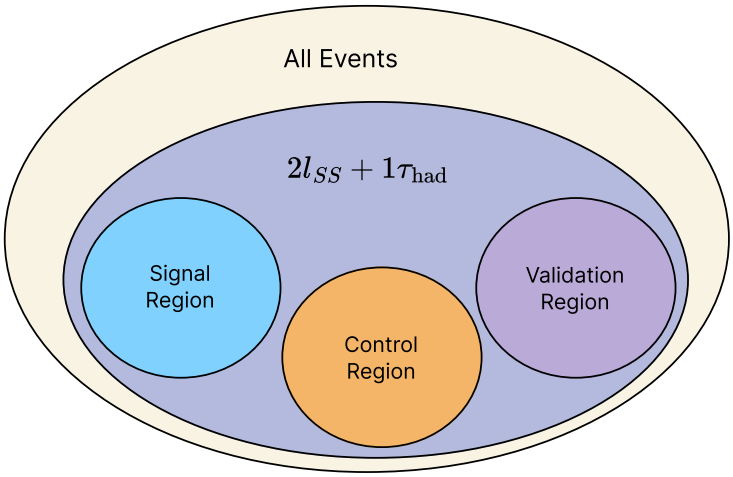


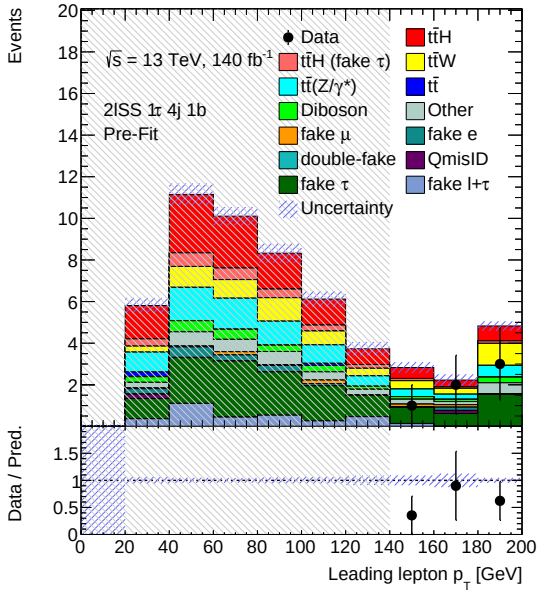
Figure 1.5: Relationship between different regions.

We briefly describe the notion of regions in particle physics. There are three types of regions:

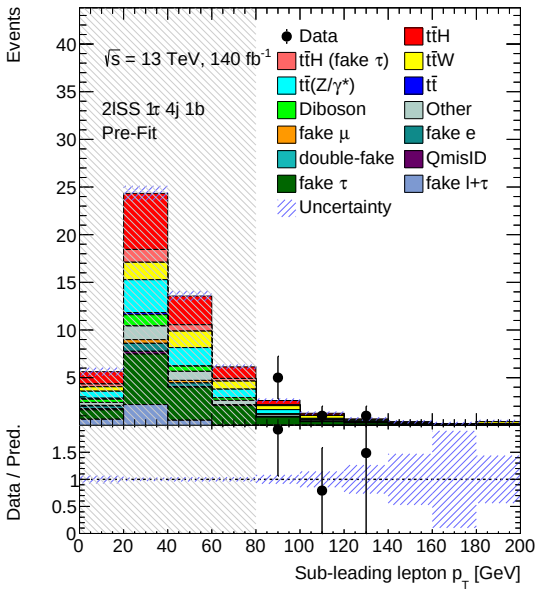
- **Signal Region (SR)** is where we expect the events of interest, the signal, to be most prevalent. It is defined by certain selection criteria that maximize the signal's prominence against the background. Appendix A.2 shows the precise definition of the SR.
- **Control Regions (CRs)** is where we estimate the amount of background contamination present in the SR. The CR is characterized by negligible signal but a significant amount of background events, similar to what we expect in the SR. By studying the CR, we can understand and model the background in the SR.

- **Validation Regions (VRs)** are used to test the reliability of our model predictions and the Monte Carlo (MC) simulations. VRs are typically chosen where neither the signal nor the background is expected to be particularly high or low. Any significant deviation of the observed data from our model predictions in the VR may indicate the presence of a new physics process or systematic errors in our model or simulation.

These regions are not arbitrarily defined but are carefully chosen based on detailed knowledge of the physics processes involved and the detector's characteristics.



(a) Distribution of the transverse momentum of the leading lepton.



(b) Distribution of the transverse momentum of the subleading lepton.

Figure 1.6: Distributions of the transverse momentum of the leading and subleading leptons inside the SR. Table 1.2 shows the precise numbers of events of the backgrounds, and Table 1.3 shows the numbers for the fake lepton events. The areas are crossed out because the events in these regions are blinded. Blinding refers to hiding the data points in the bin if signal to background ratio is larger than a certain threshold. A threshold of 0.3 is used as blinding condition for all bins.

Figure 1.6a and Figure 1.6b show the distributions of the leading and subleading leptons' transverse momenta inside the SR. This type of plots (yields plots) is most common generated by the `TRExFitter`[8] software. Such plots are often used to check the modelling of the variables, catch some trivial issues early on, and compare the differences between the different versions of the MC production (section 1.6). Further plots are presented in the Appendix A.4.

Note that the fake electron, muon and tau contributions are based on a Template Fit

[8] `TRExFitter` is a framework for binned template profile likelihood fits heavily used at CERN. The documentation can be found at `https://trexfitter-docs.web.cern.ch/trexfitter-docs/`. The code can be found at `https://gitlab.cern.ch/TRExStats/TRExFitter`

Process	SR (Raw)	SR (Weighted)	ALL (Raw)	ALL (Weighted)
$t\bar{t}H$	15293	12.22	834970	523.42
$t\bar{t}W$	1479	5.49	581089	1680.34
$t\bar{t}W_{EW}$	74	0.59	15314	122.36
$t\bar{t}Z$	9612	7.78	1750978	1550.66
$t\bar{t}$	3	0.33	299808	33487.35
Diboson (VV)	1433	2.55	3829589	7046.18
tZ	32	0.29	40946	378.09
WtZ	209	1.20	45931	248.63
tW	0	0.00	4493	413.89
$t\bar{t}t$	449	0.16	22980	8.08
$t\bar{t}t\bar{t}$	482	0.97	22746	45.27
$ggVV$	38	0.01	792331	354.08
VVV	20	0.02	151324	60.93
VH	0	0.00	255	102.55
$WttW$	94	0.81	5944	50.56
$tHjb$	3058	0.14	445666	21.41
tWH	492	0.18	30587	15.95
Total	32768	32.72	8874951	46109.77

Table 1.2: Comparison of the number of events generated by the MC simulation (section 1.6) before and after applying the SR selection criteria. For each process, the number of raw events and the number of weighted events are given.

method[9] using simulated shapes and overall normalization with recorded data in dedicated selections (CRs). Fake lepton events are not used in the Neural Network (NN) training.

Fake lepton events are subtracted from the simulated MC sample, and the fake lepton events from the Template Fit method are added. In total, there are 14.96 $t\bar{t}H$ and 40.21 background events after the preselection in the SR.

As a general rule in the ATLAS Collaboration, the recorded data must be blinded for those bins where the signal over background ratio is above a certain threshold. This blinding avoids a bias in the analysis, and the blinding condition is only removed after the analysis is completed and finalized, and the ATLAS Collaboration gives approval for unblinding.

1.4 Significance

Significance represents the confidence level at which we can assert that a detected event is indeed a $t\bar{t}H$ event rather than a product of background noise or other processes. A robust measurement of significance ensures that the observations are not mere statistical fluctuations.

[9]Private Communication, August 2023, Nello Bruscino.

Process	SR (Weighted)
fake μ	0.71
fake e	1.62
fake $(e + \mu)$	0.02
fake $(\mu + \mu)$	$9 \cdot 10^{-06}$
fake $(e + e)$	0.02
Q_{mis-ID}	0.39
fake τ	3.90
$t\bar{t}H$ (fake τ)	2.74
fake $\tau(t\bar{t}W)$	6.17
fake $\tau(t\bar{t}Z)$	3.50
fake $\tau + \mu$	1.33
fake $\tau + e$	1.48
fake $\tau + \mu + e$	$1.4 \cdot 10^{-3}$
fake $\tau + \mu + \mu$	10^{-5}
fake $\tau + e + e$	10^{-3}
fake $\tau + Q_{mis-ID}$	0.53
Total	22.43

Table 1.3: Number of fake lepton events inside the SR.

1.4.1 Poisson Distribution

In particle physics, data collected from collision events are assumed to occur randomly and independently, thus following the principles of a Poisson distribution. A Poisson distribution is a discrete probability distribution that expresses the probability of a given number of events occurring in a fixed interval of time or space, given a fixed average rate of these events.

An illustrative example might be the number of phone calls received by a call center within an hour, assuming a constant average rate. If the call center typically receives an average of 10 calls per hour, and the calls are independent of each other, the number of calls received in any given hour can be modeled by a Poisson distribution with a mean of $\lambda = 10$. However, in real-world applications, the assumption of a constant rate might not always hold. For example, the probability of receiving a call at 1pm may differ substantially from that at 1am, reflecting variations in call patterns throughout the day. Such complexities necessitate a more nuanced approach in actual implementations, where time-dependent variations in the rate of occurrence may need to be considered.

For a Poisson process, the average number of events in an interval is designated by the parameter λ, which is the rate parameter. The probability of observing k events in an interval is given by the equation:

$$P(k \text{ events in interval}) = \frac{\lambda^k e^{-\lambda}}{k!}, \tag{1.1}$$

where e is the base of the natural logarithm, and $k!$ is the factorial of k.

Figure 1.7 shows the Poisson distribution for various values of λ.

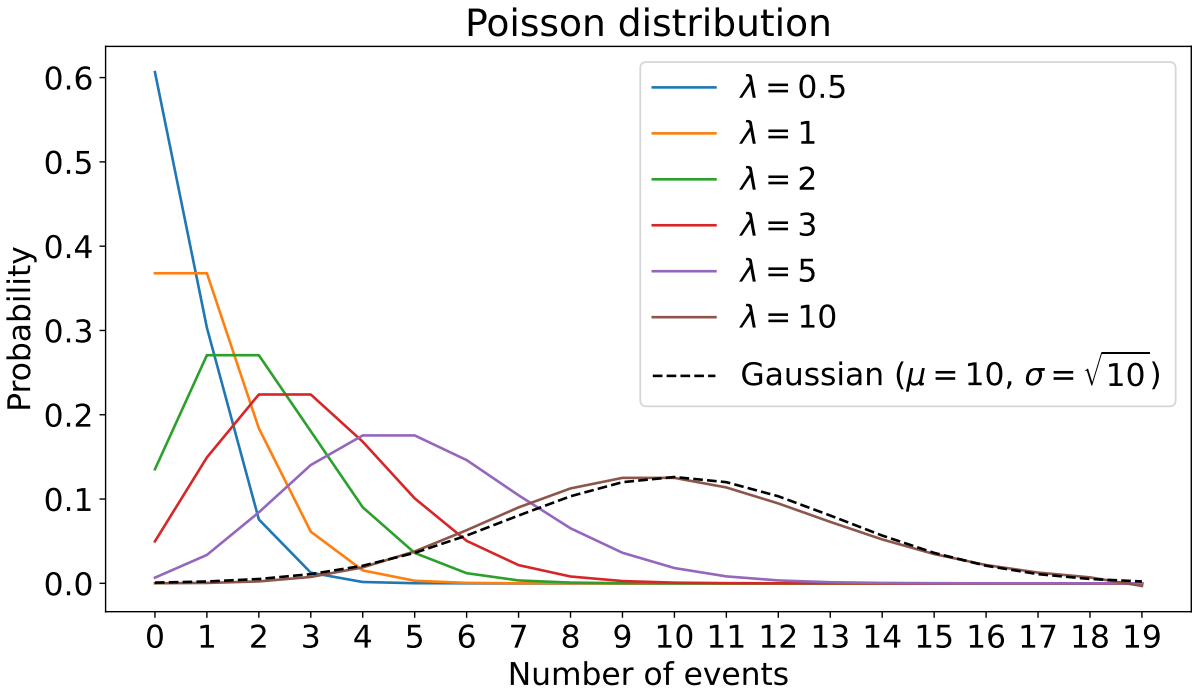


Figure 1.7: Poisson distribution for various values of λ. For large λ, the Poisson distribution approaches a Gaussian distribution.

1.4.2 Central Limit Theorem

The Poisson distribution and the Gaussian (or Normal) distribution are linked by the Central Limit Theorem (CLT), one of the fundamental theorems in probability theory and statistics.

The CLT states that the sum of a large number of independent and identically distributed random variables, each of which may be arbitrarily distributed, will approximately follow a Gaussian distribution, regardless of the shape of the original distribution. This is true provided the mean and variance of the original distributions are finite.

For a Poisson process, consider that each event occurs independently. So, we can imagine that our λ (average number of events in a given time period) is the sum of several smaller, independent rates. For instance, λ could be the result of n independent processes each with a rate of $\frac{\lambda}{n}$. Thus, our Poisson process with rate λ can be viewed as the sum of n Poisson processes each with a rate $\frac{\lambda}{n}$.

So, what happens when n is large and $\frac{\lambda}{n}$ is small? Each individual Poisson process will rarely contribute an event to the sum, but there are a lot of them (n is large). This is precisely the sort of condition under which the CLT operates. As such, we expect for large λ that the Poisson distribution will start to look more and more like a Gaussian distribution, as the Poisson distribution is the sum of many independent and identically distributed random variables.

Mathematically, a random variable X that is Poisson-distributed with mean λ can be approximated by a Gaussian distribution with mean λ and variance λ, if λ is large. This is usually taken to mean $\lambda > 20$ or 30 (Figure 1.7). Thus, for a Poisson distribution with large λ, it can be approximated as

$$X \sim N(\lambda, \sqrt{\lambda})\,. \tag{1.2}$$

This allows us to use Gaussian statistics for large λ, which are mathematically more tractable. In the context of particle physics, the number of events is often large, so the Poisson distribution can be approximated by a Gaussian distribution, allowing us to use Gaussian statistics to analyze the data.

In our context, each collision event in the $t\bar{t}H$ could either be a signal (in our case, a $t\bar{t}H$ event) or part of the background (any other process). The events are counted, and the count follows a Poisson distribution.

1.4.3 Significance

In the context of particle physics, and scientific experiments in general, significance plays a crucial role in hypothesis testing. Hypothesis testing is a statistical method used to make inferences or draw conclusions about a population based on a sample of data. The methodology of hypothesis testing involves the formulation of two competing hypotheses, the null hypothesis (H0) and the alternative hypothesis (H1). The null hypothesis is a statement about the population that will be accepted if the sample data do not provide sufficient evidence that it is false. On the other hand, the alternative hypothesis is a claim about the population that will be accepted if the sample data provide sufficient evidence that it is true. In our case specifically, the null hypothesis is that $t\bar{t}H$ process does not exist, which we either accept or reject based on the amount of evidence provided by the data.

The process of hypothesis testing involves collecting data and calculating a test statistic which is then compared to a critical value to decide whether to accept or reject the null

hypothesis. This is where the p-value comes into play. The p-value is a probability that provides a measure of the evidence against the null hypothesis provided by the data. A smaller p-value provides stronger evidence against the null hypothesis. If the p-value is below a predetermined significance level, typically 0.05 or 0.01, the null hypothesis is rejected in favor of the alternative hypothesis. Figure 1.8 shows the relationship between the p-value and the significance level.

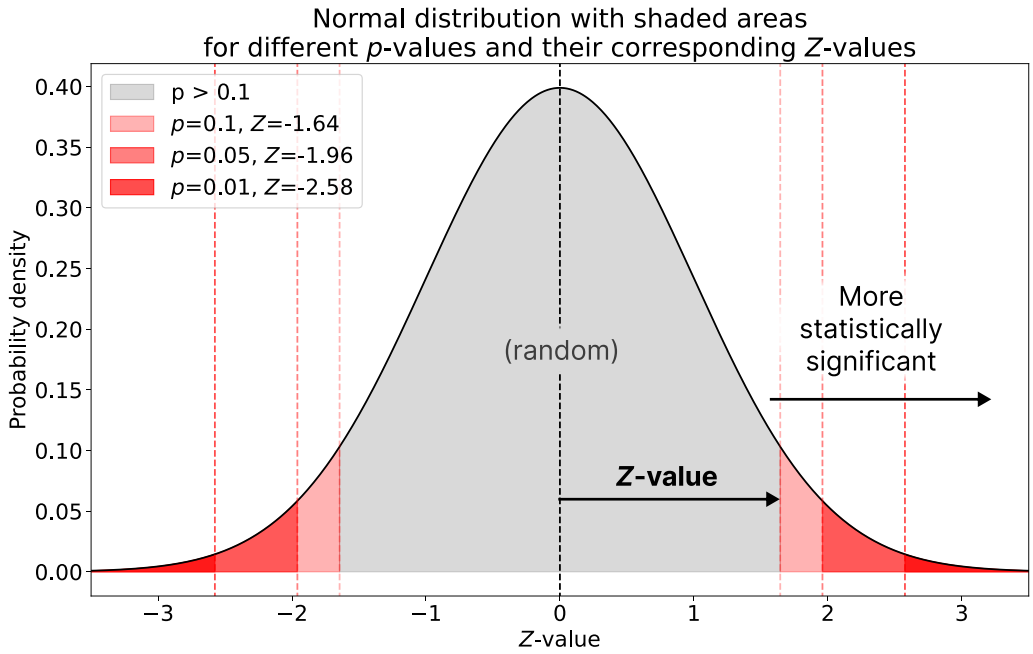


Figure 1.8: The relationship between the p-value and the significance level. Note that p-value is defined for the two-tailed test, while in our case we concerned with the case where we have more events than expected. However, this is not a problem, as the the significance level computed would underestimate the true significance level, meaning that in reality we would expect even better results.

In particle physics, the null hypothesis often refers to the background-only hypothesis, i.e., the hypothesis that only known SM processes are occurring. The alternative hypothesis, on the other hand, includes both the background and a potential new signal.

This is where we connect hypothesis testing to Poisson statistics. We model the number of observed events as a random variable that follows a Poisson distribution, with an expectation value equal to the sum of the expected number of background events (b) and signal events (s). If the actual observed number of events is significantly larger than the expected number of background events, then we have evidence against the null hypothesis.

The "significance" in particle physics refers to how many standard deviations an observed result is away from the expectation under the null hypothesis. If our data gives a result that is very unlikely under the null hypothesis (say, less than a 0.01 chance), we have

strong evidence against the null hypothesis. The significance Z is, in essence, the number of standard deviations that the observed data is away from the expectation under the null hypothesis, with $Z = 1$ corresponding to a p-value of about 0.16 (or 16%), and $Z = 2$ corresponding to a p-value of about 0.023 (or 2.3%). Note that we are interested in a one-sided conference level (as the number of signal events s is assumed to be positive), thus 2σ corresponds approximately to the $1 - 0.05 = 0.95$ CL (Table 1.4).

$$Z(p) = \mathbf{\Phi^{-1}}(1-p) \qquad p(Z) = 1 - \mathbf{\Phi}(Z) = \int_Z^\infty \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx \tag{1.3}$$

Significance	1	2	3	4	5
p-value (%)	15.87	2.28	0.13	$3 \cdot 10^{-5}$	$3 \cdot 10^{-7}$
p-value (one-sided) (%)	31.73	4.55	0.27	$6 \cdot 10^{-3}$	$6 \cdot 10^{-5}$

Table 1.4: Significances and corresponding p-values.

We illustrate this with an example. Suppose we only expect background processes to occur. In other words, during a measurement period, we expect to observe b background events. However, we observe $n = b+s$ events. This could be due to statistical fluctuations, or it could be due to the presence of a new signal. Because events follow the Poisson distribution, which can be approximated by a Gaussian for large b and s, we can use the Normal distribution $N(b, \sqrt{b})$ to model the number of observed events. The Z-score is then given by:

$$Z = \frac{n-b}{\sqrt{b}} = \frac{s}{\sqrt{b}} \tag{1.4}$$

We should note, however that due to the approximation of the Poisson distribution by the Gaussian distribution, the number of background events b must be large enough for the Z-score to be a good approximation of the significance.

In practice, such approximation is often used in the early stages of a search due to its simplicity. For more accurate results, a more complex statistical model is used that also accounts for different nuissance parameters (NPs) - systematic uncertainties (**Chapter** 3, [2], [3] for more details):

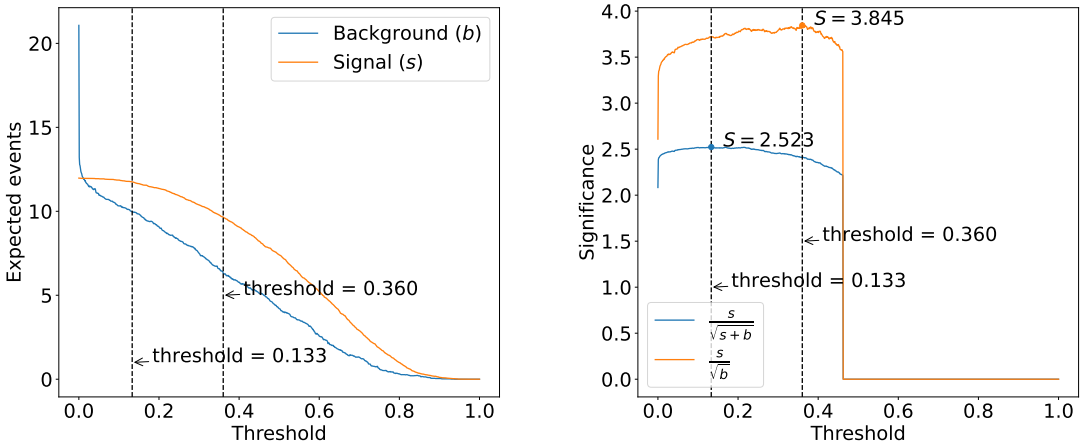
$$q_0 = -2\frac{L(\mu = 0, \hat{\hat{\boldsymbol{\theta}}}(\mu = 0))}{L(\hat{\mu}, \hat{\boldsymbol{\theta}})} \text{ where } \hat{\mu} > 0 \tag{1.5}$$

$$Z = \mathbf{\Phi^{-1}}(1 - p) = \sqrt{q_0} \tag{1.6}$$

$$p(Z) = 1 - \mathbf{\Phi}(\sqrt{q_0})\,, \tag{1.7}$$

where L is the profile likelihood function[10], (Equation 3.2), μ is the parameter of interest[11], $\boldsymbol{\theta}$ are the nuissance parameters (NPs), $\hat{\mu}$ and $\hat{\boldsymbol{\theta}}$ are the Maximum Likelihood Estimators (MLEs) of μ and $\boldsymbol{\theta}$ respectively, and $\hat{\hat{\boldsymbol{\theta}}}$ is the MLE of $\boldsymbol{\theta}$ for a fixed value of μ.

This provides a powerful tool for identifying new phenomena in particle physics. If the significance of a signal exceeds a certain threshold (often 5σ in particle physics, corresponding to a p-value of about $3 \cdot 10^{-7}$), the signal is considered a discovery.



(a) Selected events. (b) Significance

Figure 1.9: Influence of the threshold on the significance. Two approximations are used. As the threshold changes, the significance changes accordingly. The threshold that maximizes the significance is chosen.

For our experiments, we conduct the more precise calculation of significance once the best network is selected. However, to get an estimate of what we can expect, we use the approximation described above to calculate the significance during the model selection. It is important to note that when optimizing for significance, the arg max strategy is usually not optimal, so the threshold scanning (or working point optimization) is performed (Figure 1.9b):

[10] `https://statisticalmethods.web.cern.ch/StatisticalMethods/statisticaltests`

[11] For our case specifically, μ refers to the signal strength, and so it is set to 0 as the null hypothesis assumes that there is no signal.

When we start with a threshold of 0, all events are classified as $t\bar{t}H$, producing a high number of true positives, but also a high number of false positives. As the threshold increases, the model gets more conservative and starts classifying fewer events as $t\bar{t}H$, reducing the number of true positives and false positives. For each threshold, we compute both values for the significance and choose the threshold that maximizes the significance. As noted, this is only an approximation, and the significance is calculated again using the better statistical model of significance.

1.5 Motivation for Deep Learning in Particle Physics

In recent years, the application of DL techniques in various scientific domains has gained significant attention due to their ability to extract complex patterns and relationships directly from data. In the field of particle physics, DL has shown promising potential for improving the analysis of high-dimensional datasets and enhancing the discrimination power between signal and background events [4].

Within the context of the $t\bar{t}H$ process in the $2l_{SS} + 1\tau_{\text{had}}$ channel, previous work by [5] and [6] has already explored the involvement of NNs to tackle the signal and background separation problem. Their work has provided valuable insights into the feasibility and initial performance of DL approaches for this specific process. Building upon their research, we aim to further advance the application of DL techniques and address some challenges encountered in their work.

While it is true that DL methods are usually outperformed [7] by "standard" machine learning techniques like XGBoost [8], Boosted Decision Trees (BDTs), random forests [9] for tabular data, in many other applications (i.e. computer vision, Natural Language Processing (NLP)) they are commonly a State-of-the-Art (SOTA) method [10]–[12]. In the context of particle physics, DL methods have also shown potential for surpassing these traditional methods [13]. The remarkable capacity of deep neural networks to learn complex representations and capture intricate dependencies within the data makes them well-suited for tasks involving high-dimensional and non-linear patterns.

An important factor that affects the performance of DL algorithms is the availability of sufficient training data. In our work, we address this limitation by experimenting with an extended dataset obtained by dropping all the selection cuts (see section 2.4). This extended dataset provides a larger sample size and allows the neural networks to capture more diverse patterns and improve their generalization capabilities. By leveraging this extended dataset, we aim to demonstrate that DL models can achieve comparable or superior performance to other machine learning techniques, such as XGBoost, BDT, or

random forests, in the context of the $t\bar{t}H$ signal and background separation.

Moreover, we explore the application of the famous transformer [14] architecture, originally developed for Natural Language Processing (NLP) tasks, in the context of particle physics analysis. Transformers have since became extremely popular and were applied to almost every domain of machine learning. We research the performance of the transformer architecture in the context of particle physics and compare it to the previous results, as well as other tree-based methods.

By leveraging the capabilities of DL and exploring novel architectures like transformers, we seek to address the challenges encountered in previous work and enhance the performance of signal and background separation in the $2l_{SS} + 1\tau_{\text{had}}$ channel. Through our research, we aim to contribute to the growing body of knowledge on DL in particle physics and demonstrate its potential for advancing the field.

1.6 Monte Carlo (MC) Simulation

To be able to train classifiers for the precise identification of $t\bar{t}H$ events, we must adopt a method that transcends mere observation of final state particles. Our particle accelerator, engineered to collide particles and equipped with advanced detectors, can capture various attributes of these final states, including their energy, momentum, and type.

The obstacle we face is the intrinsic limitation of detecting only the final state particles and not the elusive intermediate processes. Since the intermediate particles remain undetectable, due to their extremely short lifetimes, limitations in current detection technology, the inherent complexity of the interactions, and the energy levels at which they exist, we must reconstruct the entire collision event from the observable final state particles.

This problem can be elegantly formulated as a supervised learning task (see section 2.1). Although clustering presents an alternative, supervised learning often proves to be more efficient and is favored when the conditions allow for its use (see Appendix A.5 for more details).

By employing a theoretical model with established branching rules, we construct a MC simulation. We use software like Pythia[12] to generate a labeled dataset where the final state particles and their properties are the features, and the intermediate processes, such as $t\bar{t}H$ or $t\bar{t}W$, are known and are thus assigned as the corresponding labels.

A vital aspect of our simulation is its adaptability to various detector and accelerator configurations. This ensures an accurate representation, capturing not only the fundamental

[12] `https://pythia.org/`

physical processes but also the specific characteristics of how these processes are observed in a real-world particle accelerator environment.

This simulation relies heavily on having an accurate theoretical model and precise modeling of various technical aspects of the detectors. Recognizing that the simulation may have imperfections, we engage in an iterative process that involves comparing the simulated results with real data, finding and accounting for any discrepancies, and producing a new more refined and accurate version of the simulated dataset[13].

1.6.1 Event Weighting

The $t\bar{t}H$ process is very rare and accounts for only about 1% of all Higgs production. This results in a very low number of signal events, and, as noted before, training on such a small dataset is practically not tractable. To alleviate this difficulty, more signal events are explicitly generated.

However, this leads to a very different distribution than the one observed in the real data. To align the MC simulation with the real, the weight w_i is applied to each event i. This weight is essentially the estimate of the probability of the event i occurring in the real data, also given specific detector configuration.

Various factors contribute to the weight of an event. The most important ones are the luminosity, that is different for different runs of the LHC, and the cross-section. We provide the complete formula in Appendix A.3.

[13]This work was done on the version 8 (v0801) of the n-tuples. Appendix A.1 shows list of files for each process.

Chapter 2

Methodology

2.1 Problem Formulation

2.1.1 Empirical Risk Minimization (ERM)

As stated before, our primary objective in this thesis is to distinguish the $t\bar{t}H$ events from other events of the other detected by the ATLAS detector. Given an observation $\boldsymbol{x} \in \mathcal{X}$, we want to predict its corresponding class label $y \in \mathcal{Y}$. Here $\mathcal{X}$ denotes the space of all possible observations (in our case this corresponds to the different measurements of the event), and $\mathcal{Y}$ denotes the space of all the class labels we are differentiating between. We can further split the problem into either a binary classification (seeking to differentiate between $t\bar{t}H$ (signal) and not $t\bar{t}H$ (background)) or a multi-class classification (seeking to correctly discriminate between each of the processes - $t\bar{t}H$, $t\bar{t}W$, $t\bar{t}Z$, $t\bar{t}$, etc.).

As we approach this task as a supervised learning problem (section 1.6), we assume that a set of labeled observations $\mathcal{T}^{\text{trn}} = (\boldsymbol{x}_i, y_i)_{i=1}^{N}$ is provided. Here $\boldsymbol{x}_i \in \mathbf{X}$ is a feature vector representing different properties (features) of an event and $y_i \in \mathbf{Y}$ is its corresponding true class label. We assume there exists a joint probability distribution $P(\boldsymbol{x}, y)$ over the observations $\boldsymbol{x}$ and their corresponding class labels y. Then, we require the examples in the training set $(\boldsymbol{x}_i, y_i) \in \mathcal{T}^{\text{trn}}$ to be drawn i.i.d. from the joint distribution $P(\boldsymbol{x}, y)$.

We also assume that there is a non-negative real-valued loss function $L(y, \hat{y})$ that quantifies the discrepancy between the true label y and the predicted label $\hat{y}$. The common example of such a loss function would be a zero-one loss function, which is defined as

$$L_{0/1}(y, \hat{y}) = \begin{cases} 0 & \text{if } y = \hat{y} \\ 1 & \text{otherwise} \end{cases}. \tag{2.1}$$

The goal is to find the best hypothesis $h^* \in \mathcal{H} : \mathbf{X} \to \mathbf{Y}$ that would minimize the expected loss over the joint distribution $P(\boldsymbol{x}, y)$:

$$h^* = \arg\min_{h \in \mathcal{H}} \mathbb{E}_{(\boldsymbol{x},y)\sim P(\boldsymbol{x},y)}[L(y, h(\boldsymbol{x}))] \,. \tag{2.2}$$

In practice, we do not have access to the joint distribution $P(\boldsymbol{x}, y)$, but only to the training set $\mathcal{T}^{\text{trn}}$. To tackle this problem, we use the ERM principle [15], which states that the best hypothesis h^* is the one that minimizes the empirical risk over the training set $\mathcal{T}^{\text{trn}}$:

$$\hat{h} = \arg\min_{h \in \mathcal{H}} R^{\text{trn}}_{\mathcal{T}}(h) = \arg\min_{h \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^{N} L(y_i, h(\boldsymbol{x}_i)) \,. \tag{2.3}$$

2.1.2 Validation and Test Sets

Consider, for example the following "cheating" classifier:

$$h(\boldsymbol{x}) = \begin{cases} y_i & \text{if } \exists i : \boldsymbol{x} = \boldsymbol{x}_i \\ y_0 & \text{otherwise} \end{cases} . \tag{2.4}$$

This classifier would have zero empirical risk, but would perform poorly on unseen data (would not generalize well). This is referred to as overfitting (Figure 2.1). Generally, in case of an unconstrained hypothesis space $\mathcal{H}$, we have no guarantee that the empirical risk $R^{\text{trn}}_{\mathcal{T}}(h)$ is a good approximation of the true risk $R(h)$.

Specifically, the problem in this case is that the prediction $\hat{y}_i$ depends not only on the observation $\boldsymbol{x}_i$, but also on the labels $y_1, \ldots, y_N$. Consider, for example, a NN classifier $h_{\boldsymbol{\theta}}$ with trainable parameters $\boldsymbol{\theta}$. When training $h_{\boldsymbol{\theta}}$ on the training set by back-propagation, $\boldsymbol{\theta}$ becomes implicitly conditioned on the true labels $y^1, \ldots, y^s$ that the network has encountered before (s denotes the training step). This violates the i.i.d. assumption and thus the empirical risk $R^{\text{trn}}_{\mathcal{T}}(h_{\boldsymbol{\theta}})$ is not a good approximation of the true risk $R(h_{\boldsymbol{\theta}})$ anymore.

To address this issue and more accurately assess the generalization ability of the classifier h, we need a separate set $\mathcal{T}^{\text{val}} \sim P(\boldsymbol{x}, y)$ that provides an unbiased estimate of the true risk $R(h)$. This set is called the validation set. The validation set is used to compare the performance of different classifiers. Consider two classifiers h_1 and h_2, where the risk on the training set is $R^{\text{trn}}_{\mathcal{T}}(h_1) < R^{\text{trn}}_{\mathcal{T}}(h_2)$, but the risk on the validation set is $R^{\text{val}}_{\mathcal{T}}(h_1) > R^{\text{val}}_{\mathcal{T}}(h_2)$. In this case, we prefer the classifier h_2 over h_1 as it generalizes better to unseen data. The specific case is often seen with the NN classifiers, where the classifier h is parametrized by $\boldsymbol{\theta}$. Then essentially we compare $h_1 = h_{\boldsymbol{\theta}_1}$ and $h_2 = h_{\boldsymbol{\theta}_2}$, where $\boldsymbol{\theta}_1$

and $\boldsymbol{\theta}_2$ are two different sets of parameters. The process of selecting the best classifier h^* from a set of classifiers $\mathcal{H}$ is called model selection.

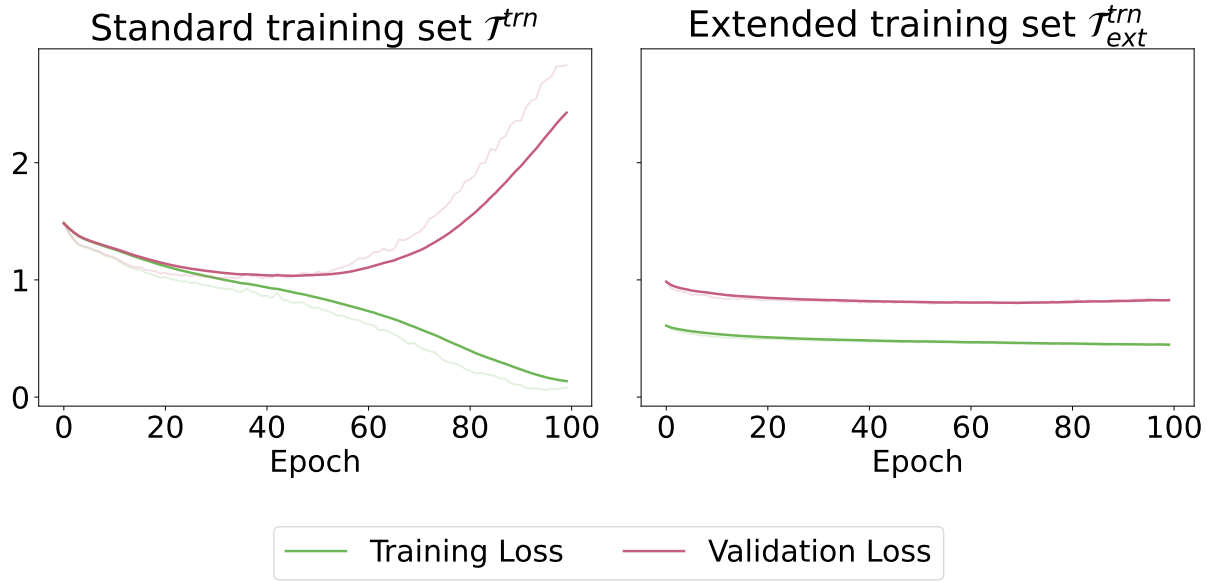


Figure 2.1: Training and validation losses during the training process on standard versus extended training sets. The semi-transparent lines show the actual loss values, while the solid lines show the exponential moving average. The left plot shows the training of the FT-Transformer with 2 blocks (subsection 2.3.5) on the standard training set. Because of the lack of training samples, and high capacity of the model, we observe overfitting. The training loss continues to decrease while the validation loss starts to increase. The checkpoint with the validation loss is the lowest is then used as the final model. This is referred to as early stopping. The best way to prevent overfitting is to get more training data (section 2.4). If that is not possible, one might also consider augmentation techniques, or regularization - dropout (subsection 2.3.6), weight decay etc. The right plot shows the training of the FT-Transformer with 5 blocks (subsection 2.3.5) on the extended training set. Also, the 20% dropout is introduced. We observe only a slight overfitting, which means that the model has generalized a lot better.

The caveat of using the validation for model selection is that in doing so we are implicitly fitting to the validation set, as now our best classifier h^* is also conditioned on the evaluations of the other classifiers on the validation set. To address the similar issue, a third set is normally introduced, called the test set $\mathcal{T}^{\text{tst}} \sim P(\boldsymbol{x}, y)$. The test set should be used only once to assess the performance of the fully-trained classifier h^*.

2.1.3 Training

The process of finding the best hypothesis h^* is called training (or learning). In the context of ERM, this further reduces to minimization of the empirical risk $R^{\text{trn}}_{\mathcal{T}}(h)$. As described before, the empirical risk is an expectation of the loss function $L(y, h(\boldsymbol{x}))$ over the training set $\mathcal{T}^{\text{trn}}$. Thus, the choice of the loss function will determine the available

training algorithms.

This work focuses on training Neural Network (NN) classifiers. A NN can be formally described as a parametric model parametrized by a vector of parameters $\boldsymbol{\theta}$. NNs are generally composed of multiple layers $f^1_{\boldsymbol{\theta}_i}, \dots f^D_{\boldsymbol{\theta}_L}$ (D denotes the total number of layers - depth of the network), where each layer $f^i_{\boldsymbol{\theta}_i}$ is a parametric function parametrized by $\boldsymbol{\theta}_i$. NNs can take different architectures, a common one[1] is a feed-forward NN where the output of the layer $f^i_{\boldsymbol{\theta}_i}$ is fed as an input to the next layer $f^{i+1}_{\boldsymbol{\theta}_{i+1}}$. The output of the last layer $f^D_{\boldsymbol{\theta}_L}$ is the output of the network $h_{\boldsymbol{\theta}}$. Alternative to feed-forward NNs would be the network that contain cycles (e.g. Recurrent Neural Networks (RNNs)[2] [16], [17]).

The parameter vector of the whole neural network can be seen as a concatenation of the parameters of the individual layers:

$$\boldsymbol{\theta} = \begin{bmatrix} \boldsymbol{\theta}_1 \\ \vdots \\ \boldsymbol{\theta}_L \end{bmatrix} . \tag{2.5}$$

The goal is then to find the optimal set of parameters $\boldsymbol{\theta}^*$ that minimizes the empirical risk $R^{\text{trn}}_{\mathcal{T}}(h_{\boldsymbol{\theta}})$.

Training NNs efficiently involves the use of gradient-based optimization algorithms where the gradient of the empirical risk $R^{\text{trn}}_{\mathcal{T}}(h_{\boldsymbol{\theta}})$ with respect to the parameters $\boldsymbol{\theta}$ is computed and is used to update the parameters $\boldsymbol{\theta}$ in the direction of the steepest descent. The gradient of the empirical risk $R^{\text{trn}}_{\mathcal{T}}(h_{\boldsymbol{\theta}})$ with respect to the parameters $\boldsymbol{\theta}$ can be computed using the chain rule:

$$\nabla_{\boldsymbol{\theta}} R^{\text{trn}}_{\mathcal{T}}(h_{\boldsymbol{\theta}}) = \nabla_{\boldsymbol{\theta}} \frac{1}{N} \sum_{i=1}^{N} L(y_i, h_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) = \frac{1}{N} \sum_{i=1}^{N} \nabla_{\boldsymbol{\theta}} L(y_i, h_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) \,, \tag{2.6}$$

and is essentially an average of the gradients of the loss function $L(y_i, h_{\boldsymbol{\theta}}(\boldsymbol{x}_i))$ with respect to the parameters $\boldsymbol{\theta}$ over the training set $\mathcal{T}^{\text{trn}}$. During the update step, the parameters $\boldsymbol{\theta}$ are updated in the direction of the steepest descent:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} R^{\text{trn}}_{\mathcal{T}}(h_{\boldsymbol{\theta}}) \,, \tag{2.7}$$

[1] In this thesis we use an adaptation of Residual Neural Networks (ResNets) to tabular data and **F**eature **T**okenizer + **Transformer**s (FT-Transformers), which are both special cases of feed-forward NNs.

[2] Overview of the different types of RNNs `https://paperswithcode.com/methods/category/recurrent-neural-networks`.

where α is the learning rate, a hyperparameter controlling the size of the update step. In practice, instead of computing the gradient over the whole training set $\mathcal{T}^{\text{trn}}$, the gradient is computed on the so-called mini-batches of the training data. This gradient, computed on the mini-batch acts as an unbiased estimate of the true gradient. This approach is called Stochastic Gradient Descent (SGD) and is the most common optimization algorithm used for training NNs[3]. Throughout our experiments we use an improvement of SGD called Adaptive Moment Estimation with Weight Decay (AdamW) [18], [19] which is an adaptive learning rate optimization algorithm that uses the first and second moments of the gradient to adapt the learning rate dynamically.

Computation of the gradient of the loss function $L(y_i, h_{\boldsymbol{\theta}}(\boldsymbol{x}_i))$ with respect to the parameters $\boldsymbol{\theta}$ is done using the chain rule. The chain rule is a formula for computing the derivative of the composition of two or more functions:

$$\frac{\mathrm{d}(\boldsymbol{f} \circ \boldsymbol{g})}{\mathrm{d}\boldsymbol{x}} = \frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}\boldsymbol{g}} \frac{\mathrm{d}\boldsymbol{g}}{\mathrm{d}\boldsymbol{x}}\,, \tag{2.8}$$

where $\boldsymbol{f}$ and $\boldsymbol{g}$ are functions of $\boldsymbol{x}$.

In the context of NNs, the chain rule is used to compute the gradient of the loss function with respect to the parameters $\boldsymbol{\theta}$ by an iterative approach. First, let the outputs of the individual layers are recorded during the forward pass (or forward propagation):

$$\boldsymbol{z}^i = \boldsymbol{f}^i(\boldsymbol{z}^{i-1}) \quad i = 1, \ldots, L \tag{2.9}$$

$$\boldsymbol{z}^0 = \boldsymbol{x}\,, \tag{2.10}$$

where $\boldsymbol{z}^i$ is the output of the i-th layer and $\boldsymbol{z}^0 = \boldsymbol{x}$ is the input to the first layer. Then, we can compute the gradients with respect to the outputs of the layers:

[3]Aside from having low computational and memory requirements, being able to learn online, SGD has some other advantages, such as being able to escape local minima, generalize better and provide the regularization effect - all consequences of an inherent noise in the gradient estimate.

$$\boldsymbol{\delta}^D = \frac{\mathrm{d}L}{\mathrm{d}\boldsymbol{f}^D} \tag{2.11}$$

$$\boldsymbol{\delta}^{D-1} = \frac{\mathrm{d}L}{\mathrm{d}\boldsymbol{f}^D}\frac{\mathrm{d}\boldsymbol{f}^D}{\mathrm{d}\boldsymbol{z}^{D-1}} = \boldsymbol{\delta}^D\frac{\mathrm{d}\boldsymbol{f}^D}{\mathrm{d}\boldsymbol{f}^{D-1}} \tag{2.12}$$

$$\boldsymbol{\delta}^{D-2} = \boldsymbol{\delta}^{D-1}\frac{\mathrm{d}\boldsymbol{f}^{D-1}}{\mathrm{d}\boldsymbol{f}^{D-2}} \tag{2.13}$$

$$\vdots$$

$$\boldsymbol{\delta}^1 = \boldsymbol{\delta}^2\frac{\mathrm{d}\boldsymbol{f}^2}{\mathrm{d}\boldsymbol{f}^1}\,. \tag{2.14}$$

Next, the gradient of the loss function with respect to the parameters of each layer $\boldsymbol{\theta}_i$ is computed as

$$\frac{\mathrm{d}L}{\mathrm{d}\boldsymbol{\theta}_D} = \frac{\mathrm{d}L}{\mathrm{d}\boldsymbol{f}^D}\frac{\mathrm{d}\boldsymbol{f}^D}{\mathrm{d}\boldsymbol{\theta}_D} = \boldsymbol{\delta}^D\frac{\mathrm{d}\boldsymbol{f}^D}{\mathrm{d}\boldsymbol{\theta}_D} \tag{2.15}$$

$$\frac{\mathrm{d}L}{\mathrm{d}\boldsymbol{\theta}_{L-1}} = \frac{\mathrm{d}L}{\mathrm{d}\boldsymbol{f}^D}\frac{\mathrm{d}\boldsymbol{f}^D}{\mathrm{d}\boldsymbol{f}^{D-1}}\frac{\mathrm{d}\boldsymbol{f}^{D-1}}{\mathrm{d}\boldsymbol{\theta}_{L-1}} = \boldsymbol{\delta}^{D-1}\frac{\mathrm{d}\boldsymbol{f}^{D-1}}{\mathrm{d}\boldsymbol{\theta}_{D-1}} \tag{2.16}$$

$$\frac{\mathrm{d}L}{\mathrm{d}\boldsymbol{\theta}_{D-2}} = \boldsymbol{\delta}^{D-2}\frac{\mathrm{d}\boldsymbol{f}^{D-2}}{\mathrm{d}\boldsymbol{\theta}_{D-2}} \tag{2.17}$$

$$\vdots$$

$$\frac{\mathrm{d}L}{\mathrm{d}\boldsymbol{\theta}_1} = \boldsymbol{\delta}^1\frac{\mathrm{d}\boldsymbol{f}^1}{\mathrm{d}\boldsymbol{\theta}_1}\,. \tag{2.18}$$

The chain rule is the reason why NNs are so successful in practice. It allows for the efficient computation of the gradient even for very deep NNs.

2.1.4 Cross-Entropy Loss

In order for the back-propagation to work, all the functions must be differentiable. The zero-one loss function that we used in the previous section does not conform to this requirement. In practice, when training NNs on the classification tasks, the cross-entropy loss function is used. Cross-entropy loss operates on the probabilities, rather than on the predicted label, thus making it differentiable and suitable to be used in the back-propagation algorithm. The cross-entropy loss function is defined as:

$$L(y_i, \boldsymbol{h}(\boldsymbol{x}_i)) = -\sum_{j=1}^{|\mathbf{Y}|}[\![y_i = y_j]\!]\log(h_j(\boldsymbol{x}))\,. \tag{2.19}$$

In certain scenarios, such as imbalanced datasets, it may be beneficial to apply different weights to different classes. Class weights are used to give more importance to under-represented classes, effectively balancing the contribution of each class to the overall loss. This helps the learning algorithm focus more on the minority class, which may be of particular interest or significance.

For example, consider a medical diagnosis application where 95% of the samples are negative ($y = 0$) and only 5% are positive ($y = 1$) for a specific condition. Training a model on this dataset without any adjustments may lead to a classifier that almost always predicts the negative class, since it is encountering it much more often. Such a skewed prediction can be problematic in critical applications, as missing the rare positive cases could have serious consequences. To alleviate this issue, class weights can be introduced to the loss function to give equal importance to both classes. The modified loss function is:

$$L(y_i, \boldsymbol{h}(\boldsymbol{x}_i)) = -\sum_{j=1}^{|\mathbf{Y}|} [\![y_i = y_j]\!] w_j \log(h_j(\boldsymbol{x}))\,. \tag{2.20}$$

Here weights $w_1, \ldots, w_{|\mathbf{Y}|}$ are assigned to each class, where w_i is calculated as:

$$w_i = \frac{|\{y \sim \mathbf{Y} \mid y = i\}|}{|\mathbf{Y}| \sum_{i=1}^{|\mathbf{Y}|} w_i} \quad i = 1, \ldots, |\mathbf{Y}|\,. \tag{2.21}$$

Essentially, we count the number of examples of class i and divide it by the total number of examples in the dataset. Then we normalize[4] the weights so that they sum up to 1.

Similarly, the way the cross-entropy is defined, we can also introduce a more refined sample-wise weighting. In this case, the loss function is defined as:

$$L(y_i, \boldsymbol{h}(\boldsymbol{x}_i)) = -\sum_{j=1}^{|\mathbf{Y}|} [\![y_i = y_j]\!] w_i \log(h_j(\boldsymbol{x}))\,. \tag{2.22}$$

Here we should note that w_i is not the same as the class weight w_i from the previous example. In this case, w_i is a weight assigned to each sample, rather than to each class, which we note by using the same index i as in the y_i and $\boldsymbol{x}_i$. This formulation is not commonly used, but was explored by the previous analysis [5], [6], thus we include it here for completeness. More details are given in section 2.6.

[4]This is optional, but helpful - inverse frequencies can have a large range, especially if there is extreme imbalance between the classes. Keeping the weights in the [0, 1] range also help interpretability.

2.2 Evaluating the Classifier Performance

During evaluation, it is essential to go beyond the loss function and consider different metrics that shed light on various aspects of the model's performance. These metrics offer a more comprehensive understanding of how well the classifier is doing. For example, in a binary classification problem such as diagnosing a specific medical condition, merely looking at the loss might not reveal how well the model is identifying positive cases among the minority class.

Classification metrics include measures like Accuracy, which gives an overall picture of correct classifications, and Precision and Recall, which focus on the model's performance with respect to a specific class. Other metrics like the F1-Score provide a balance between Precision and Recall, and AUC-ROC measures the ability of the model to discriminate between positive and negative classes. Choosing the right combination of these metrics is vital, as it guides the optimization during training and influences the model's generalization to unseen data.

Understanding and selecting the appropriate classification metrics ensures alignment with the problem's unique requirements and goals, enhancing the model's utility and effectiveness in real-world applications.

Confusion Matrix

The confusion matrix provides a comprehensive view of the classifier's performance. For a binary classification task, it is a 2×2 matrix where the rows correspond to the true classes and the columns correspond to the predicted classes:

$$\mathbf{C^{bin}} = \begin{pmatrix} \mathrm{TP} & \mathrm{FP} \\ \mathrm{FN} & \mathrm{TN} \end{pmatrix} \tag{2.23}$$

$$\mathrm{TP} = \sum_{i=1}^{N} [\![y_i = 1 \wedge \hat{y}_i = 1]\!] \tag{2.24}$$

$$\mathrm{FP} = \sum_{i=1}^{N} [\![y_i = 0 \wedge \hat{y}_i = 1]\!] \tag{2.25}$$

$$\mathrm{FN} = \sum_{i=1}^{N} [\![y_i = 1 \wedge \hat{y}_i = 0]\!] \tag{2.26}$$

$$\mathrm{TN} = \sum_{i=1}^{N} [\![y_i = 0 \wedge \hat{y}_i = 0]\!] \,, \tag{2.27}$$

where TP (true positive) is the number of positive instances correctly identified as positive, TN (true negative) is the number of negative instances correctly identified as negative, FP (false positive) is the number of negative instances incorrectly identified as positive (Type I error), and FN (false negative) is the number of positive instances incorrectly identified as negative (Type II error).

As explained in section 1.6, the event weights should always be used when evaluating the classifier's performance. Otherwise, the results we obtain are not representative of the real-world performance. All metrics we use thus stem from the *weighted* confusion matrix, defined as:

$$\mathbf{C}_w^{\mathbf{bin}} = \begin{pmatrix} \mathrm{TP}_w & \mathrm{FP}_w \\ \mathrm{FN}_w & \mathrm{TN}_w \end{pmatrix} \tag{2.28}$$

$$\mathrm{TP}_w = \sum_{i=1}^{N} w_i [\![y_i = 1 \wedge \hat{y}_i = 1]\!] \tag{2.29}$$

$$\mathrm{FP}_w = \sum_{i=1}^{N} w_i [\![y_i = 0 \wedge \hat{y}_i = 1]\!] \tag{2.30}$$

$$\mathrm{FN}_w = \sum_{i=1}^{N} w_i [\![y_i = 1 \wedge \hat{y}_i = 0]\!] \tag{2.31}$$

$$\mathrm{TN}_w = \sum_{i=1}^{N} w_i [\![y_i = 0 \wedge \hat{y}_i = 0]\!] , \tag{2.32}$$

where $\mathbf{C}_w^{\mathbf{bin}}$ is the confusion matrix, w_i is the MC weight of the i-th event, calculated as described in the Appendix A.3, and N is the total number of events in the evaluation set. Further on, when referring to the confusion matrix, true positives, false positives, false negatives, and true negatives, we will always be referring to their weighted counterparts, dropping the subscript w for brevity, unless otherwise specified.

Figure 2.2 shows how such confusion matrix looks in our case for the binary classification task - when we only care about differentiating between $t\bar{t}H$ and non-$t\bar{t}H$ events.

Binary confusion matrix naturally extends to a multi-class formulation (Figure 2.3), leading to a $|\mathbf{Y}| \times |\mathbf{Y}|$ matrix for a $|\mathbf{Y}|$-class classification task:

$$\mathbf{C}_{ij}^{\mathbf{multiclass}} = \sum_{k=1}^{|\mathbf{Y}|} w_k [\![y_k = i \wedge \hat{y}_k = j]\!] , \tag{2.33}$$

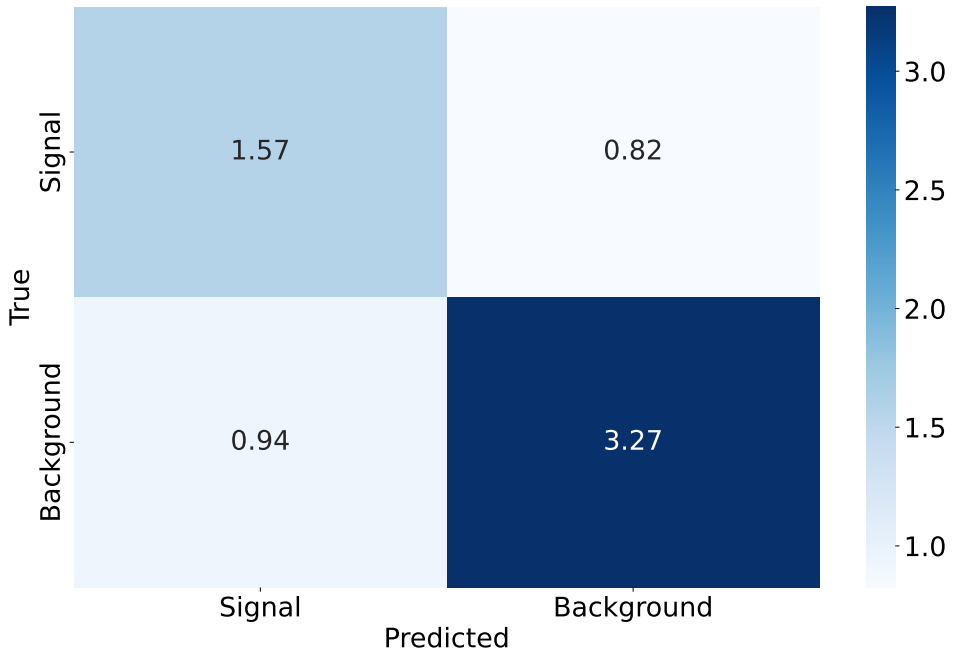


Figure 2.2: Confusion matrix for a binary classification task. This confusion matrix is produced using 5-blocks FT-Transformer (subsection 2.3.5) trained on the extended training set (section 2.4) evaluated on the validation set, that contains 20% of the SR events. Signal refers to $t\bar{t}H$, and background refers to all the other classes. The arg max classification strategy was used. Note that the classifier was *trained* to differentiate between all the classes, but during evaluation, all the non-$t\bar{t}H$ events are grouped together.

where i and j are the true and predicted classes, respectively. The diagonal elements of the matrix correspond to the correctly classified events, while the off-diagonal elements correspond to the misclassified events. The multi-class confusion matrix is not symmetric, and the sum of the elements in each row is equal to the number of events in the corresponding true class.

The confusion matrix serves as the basis for several other performance metrics, including accuracy, F1 score, and area under the Receiver Operating Characteristic (ROC) curve (AUC-ROC).

Accuracy: The Proportion of Correct Predictions

Accuracy is the most intuitive performance metric. It is the ratio of the number of correctly classified examples to the total number of examples (called events in particle physics). Accuracy is calculated trivially from the confusion matrix:

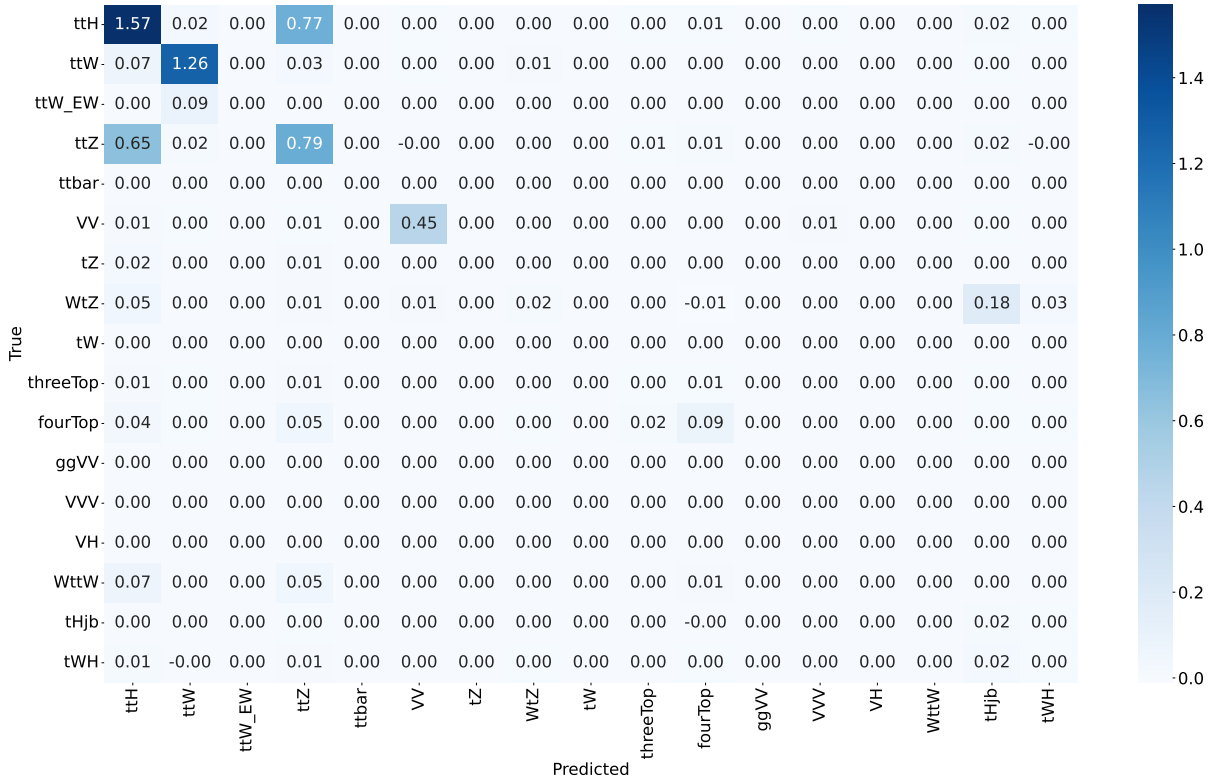


Figure 2.3: Confusion matrix for a multi-class classification task. This confusion matrix is produced using 5-blocks FT-Transformer (subsection 2.3.5) trained on the extended training set (section 2.4). The arg max classification strategy was used.

$$\text{Accuracy} = \frac{\operatorname{tr} \mathbf{C}_{ii}}{\sum_{i=1}^{\mathbf{Y}} \sum_{j=1}^{\mathbf{Y}} \mathbf{C}_{ij}}, \tag{2.34}$$

While accuracy is straightforward and commonly used, it may not always be the most representative metric, especially in cases where the classes are imbalanced. Consider the example of our particle physics problem where we are searching for $t\bar{t}H$ events and suppose that 90% of the events are background and only 10% are signal. A naïve classifier that always predicts the background class would achieve an accuracy of 90%. However, such a classifier would be entirely unhelpful for the task at hand since it fails to identify any $t\bar{t}H$ events.

Certainly, the accuracy is not entirely without value, and there are contexts where it might still be useful. Even in imbalanced scenarios, accuracy can provide a general sense of how often the classifier is correct across both the majority and minority classes. While it may not provide a nuanced view of performance on the minority class (such as $t\bar{t}H$ events in our case), it still provides information on the overall hit rate of correct predictions.

Additionally, in scenarios where the cost of false positives and false negatives are roughly equivalent, or when the class distribution in the model's operational environment matches the training data, accuracy might still be a relevant metric. It offers a quick and easily interpretable measure of performance.

However, in the specific context of searching for rare or significant events, such as $t\bar{t}H$ in particle physics, relying solely on accuracy can be misleading. It would typically be considered alongside other metrics that give more insight into the performance on the class of interest. Thus, while accuracy may not be the most representative metric in such cases, it might still hold some value as part of a broader evaluation framework.

Precision and Recall

Two other important metrics which are derived from the confusion matrix are precision and recall, which are particularly useful when dealing with imbalanced classes.

Precision is the proportion of TP to all *predicted* positives. Specifically, in our case, it is the ratio of the correctly classified $t\bar{t}H$ events, to all events classified (or misclassified) as $t\bar{t}H$. From here on, we assume $y_1 = t\bar{t}H$, and so the true $t\bar{t}H$ events correspond to the first row of the confusion matrix, while predicted $t\bar{t}H$ events correspond to the first column. Precision is then given by

$$\text{Precision} = \frac{\mathbf{C}_{11}}{\sum_{j=1}^{\mathbf{Y}} \mathbf{C}_{1j}} \,. \tag{2.35}$$

Recall (or sensitivity) is the proportion of TP to all *actual* positives. In our case, it is the ratio of the correctly classified $t\bar{t}H$ events to all actual $t\bar{t}H$ events (which the model might have missed by classifying them as background). Recall is given by

$$\text{Recall} = \frac{\mathbf{C}_{11}}{\sum_{i=1}^{\mathbf{Y}} \mathbf{C}_{i1}} \,. \tag{2.36}$$

Precision tells us how reliable our positive predictions are, while recall informs us how many of the actual $t\bar{t}H$ events we were able to detect. Both these metrics provide complementary insights, and understanding the trade-off between them is essential in many real-world classification tasks. Next, we will introduce the F1 score, a metric that combines both precision and recall to provide a balanced view of the model's performance on

both fronts.

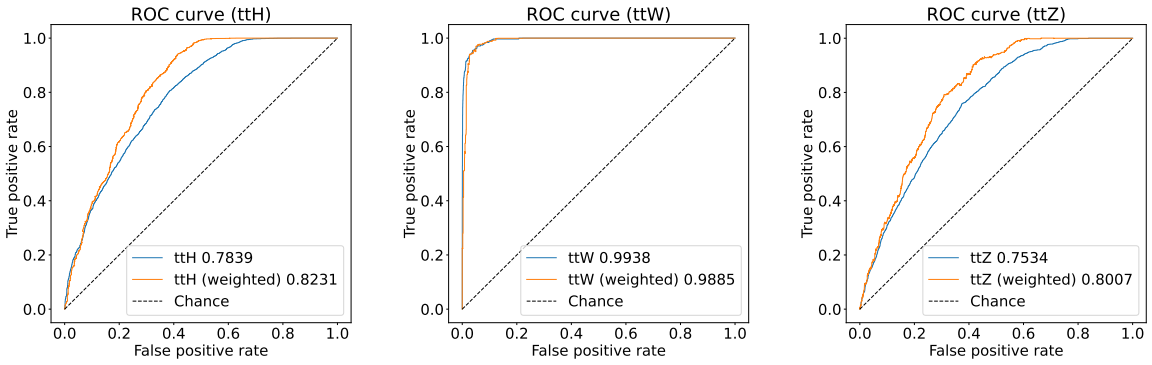
F1 Score: The Balance Between Precision and Recall

The F1 score is the harmonic mean of precision and recall, providing a balance between the two. It is calculated as:

$$\text{F1 score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.37}$$

ROC Curve and AUC: The Trade-off Between Sensitivity and Specificity

The ROC curve is a plot of the true positive rate (recall or sensitivity) against the false positive rate (1 - specificity) for different classification thresholds. The area under the ROC curve (AUC-ROC) measures the classifier's ability to distinguish between classes. A perfect classifier has an AUC-ROC of 1, while a random classifier has an AUC-ROC of 0.5. The ROC curves for the $t\bar{t}H$ and two most dominant background processes $t\bar{t}W$ and $t\bar{t}Z$ are shown in Figure 2.4.



(a) $t\bar{t}H$ (b) $t\bar{t}W$ (c) $t\bar{t}Z$

Figure 2.4: ROC curves for the $t\bar{t}H$, $t\bar{t}W$, and $t\bar{t}Z$ computed using one process versus all other processes. The curves are produced using 5-blocks FT-Transformer (subsection 2.3.5) trained on the extended training set (section 2.4). We provide both the ROCs computed with the weighted and unweighted confusion matrices for completeness and comparison with the previous analysis, however, we emphasize that the weighted ROCs are the ones that should be used always.

These metrics, combined with the loss function, provide a comprehensive view of the classifier's performance and guide the optimization process during training. They also provide a robust measure for comparing different classifiers or the same classifier with different hyperparameters. Generally, one should examine all of these metrics to get a complete picture of the classifier's performance.

2.3 Architectures and Optimization Techniques

This section covers the classifier architectures and optimization techniques we employed in our research. As a baseline model, we used an adaptation of the ResNets to the tabular data, inspired by [7]. The primary model that has shown the best results was the FT-Transformer, also adopted from [7]. The process of training these models and the optimization techniques used to improve their performance is described in the subsequent sections.

2.3.1 Previous Work

In an attempt to extend our understanding of the $t\bar{t}H$ process, this work builds upon the previous efforts made by [5] and [6]. Their research also involved training feed-forward neural networks to distinguish between the $t\bar{t}H$ and the background processes.

The primary architecture [5] utilized was a Multi-Layer Perceptron (MLP). The work examines the effect of various hyperparameters such as the number of layers, embedding size, learning rate, and batch size on the significance. The work also includes the estimation of statistical uncertainties, associated with the median signal strength. The authors experimented with binary and multi-class classification, as well as proposed a staged network approach. The staged network is composed of 5 binary classifiers (for $t\bar{t}Z$, $t\bar{t}W$, $t\bar{t}$, VV, and all the other backgrounds grouped together in one category "others"). Each of the classifiers is a MLP itself, and during the training only receives events of $t\bar{t}H$ and the corresponding background.

Although the staged network is its essence is equivalent to a single, larger MLP, it allows for the training of each subnetwork on a *different set of features*. This can potentially reduce the systematic uncertainties, associated with the final prediction.

The highest mean significance obtained with a multi-class classifier was reported to be $Z = 3.064$ while the highest mean significance obtained with a binary classifier was reported to be $Z = 3.114$. The expectation was that having the NN focus on differentiating between two classes only would improve the performance. However, the results show that there was no significant difference. In our work, however, we have observed that when the classifier is trained on multiple classes, the performance is increased (section 2.7). The highest mean significance obtained with a staged network was reported to be $Z = 2.964$, which is very similar to the other results[5].

In [6], the experiments were also extended beyond simple MLPs, experimenting with

[5]Note that different production of the input n-tuples are used. They have different calibrations, and thus a direct comparison is only approximate.

TabNet [20] and XGBoost [8]. The authors have also experimented with different hyper-parameters, as well as different fractions of the training set, and different feature sets. The best results were obtained with XGBoost trained on all the features and the whole training set, which is unsurprising. The highest mean significance obtained with XGBoost is reported as $Z = 2.90$. The authors have also evaluated the uncertainty, associated with the prediction, which is reported as $\mu = 1 + 0.42/-0.37$, where μ is the ratio of expected signal to the expectation in the SM (Equation 3.1).

Our research seeks to improve upon these previous efforts by introducing more complex architectures and advanced optimization techniques, which will be discussed in the following sections.

2.3.2 Multi-Layer Perceptron (MLP)

In previous work, [5] utilized Multi-Layer Perceptrons (MLPs) as the primary model architecture. While he experimented with combining multiple MLPs, this approach is essentially equivalent to using a single, larger MLP. This can be formalized as in [21]:

$$\texttt{MLP}(\boldsymbol{x}) = \texttt{Linear}(\texttt{MLPBlock}(\dots(\texttt{MLPBlock}(\boldsymbol{x})))) \tag{2.38}$$

$$\texttt{MLPBlock}(\boldsymbol{x}) = \texttt{Dropout}(\sigma(\texttt{Linear}(\boldsymbol{x}))) \tag{2.39}$$

Where σ is the activation function[6], `Linear` is a linear transformation:

$$\texttt{Linear}(\boldsymbol{x}) = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}\,, \tag{2.40}$$

and `Dropout` is a layer that randomly (with a fixed probability p) sets a fraction of the input features to zero [25]:

$$\texttt{Dropout}_i(\boldsymbol{x}) = \begin{cases} 0, & \text{with probability } p \\ x_i, & \text{otherwise}\,. \end{cases} \tag{2.41}$$

Here, $\boldsymbol{x} \in \mathbb{R}^{d_f}$ is an input vector of features size of size d_f. The first `Linear` layer transforms the input vector into a vector of size d_h, where d_h is the embedding size (number of hidden units), thus $\boldsymbol{W} \in \mathbb{R}^{d_h \times d_f}$ and $\boldsymbol{b} \in \mathbb{R}^{d_h}$. Each `Linear` layer in principle

[6]Common activation functions include ReLU, Leaky ReLU (LReLU) [22], [23], sigmoid, tanh, and others. Throughout our experiments we mostly use GELU activation [24].

may have a different embedding size, but in practice, we use the same embedding size for all the layers. The last `Linear` layer transforms maps the output vector to the desired output size d_o, where d_o is the number of classes to distinguish between.

2.3.3 Residual Neural Network (ResNet)

We compare the staged network to a slightly improved version of the MLP that introduces residual/skip connections between the layers (Figure 2.5) proposed by [26].

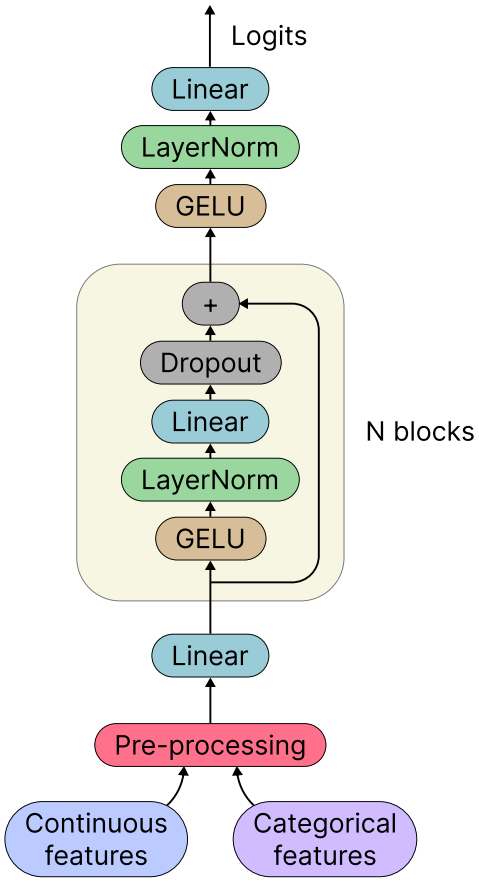


Figure 2.5: ResNet architecture.

These connections improve the training of deep neural networks, as the gradient can flow unimpeded back to the first layers. This helps to address the vanishing gradient problem. We can formalize this as in [21]:

$$\texttt{ResNet(x)} = \texttt{Prediction}(\texttt{ResNetBlock}(\ldots(\texttt{ResNetBlock}(\texttt{Linear}(\texttt{Embed}(x)))))) \quad (2.42)$$

$$\texttt{ResNetBlock(x)} = x + \texttt{Dropout}(\texttt{Linear}(\texttt{LayerNorm}(\sigma(x)))) \quad (2.43)$$

$$\texttt{Prediction}(x) = \texttt{Linear}(\texttt{LayerNorm}(\sigma(x))) \quad (2.44)$$

Because of residual connections, ResNets are very fast to train, and are more sample efficient than MLPs. While keeping the number of trainable parameters the same, we have observed that deeper networks perform better than wider ones. Although wide NNs are fast to train, they are extremely prone to overfitting, as the wide layers close to the inputs essentially memorize the training data.

We introduce a few other changes to the training procedure:

1. We use an Adaptive Moment Estimation with Weight Decay (AdamW) optimizer [19].
2. We use a `GELU` activation [24].
3. We introduce `LayerNorm` [27] layers before each `Linear` layer.

2.3.4 Pre-Processing and Embedding

In previous work, all the features were treated as continuous variables. Before feeding them to the network, they were normalized to have zero mean and unit variance. This is essential for the training of deep neural networks, as it prevents the gradients from exploding or vanishing.

However, this approach is far from optimal when working with the categorical features. The standard way of dealing with categorical features is to either use one-hot encoding, or use learnable embeddings.

Suppose we have categorical features a, b, c, d. Now let us focus a categorical feature a with n_a unique values. One-hot encoding would turn it into a vector of size n_a, with all the values being zero, except for the one corresponding to the value of the feature. Suppose, a takes it's j-th unique value a^j. Then one-hot encoding would turn it into an n_a-dimensional vector with 1 at the j-th position and 0 everywhere else:

$$a = a^j \xrightarrow[\text{one-hot}]{} \begin{pmatrix} 0 \\ \vdots \\ 1 \text{ at } j\text{-th position} \\ \vdots \\ 0 \end{pmatrix}. \tag{2.45}$$

Often, such one-hot encoded vector is then further multiplied with a learnable matrix $\boldsymbol{W}^a \in \mathbb{R}^{d_h \times n_a}$, where d_h is the dimensionality of the hidden layer. This operation can be implemented more efficiently as a lookup table:

$$a = a^j \underset{\text{lookup}}{\Longrightarrow} ((\boldsymbol{W}^a)^T)_j = \boldsymbol{e}^{a^j} . \tag{2.46}$$

Here we denote $\boldsymbol{e}^{a^j}$ as the embedded vector, corresponding to the feature a and its j-th unique value a^j.

Essentially, we construct a matrix with the number of rows corresponding to the number of unique values of the feature n_a, where each row is a vector of learnable weights of size d_h. As noted before, this reduces the space and time complexity, and so this is what is commonly used in practice.

Often we would also like to learn affine embeddings instead of just linear ones:

$$a = a^j \underset{\text{affine}}{\Longrightarrow} ((\boldsymbol{W}^a)^T)_j + \boldsymbol{b}^a = \boldsymbol{e}^{a^j} . \tag{2.47}$$

Here $\boldsymbol{b}^a$ is a learnable bias vector of size d_h for the feature a.

Note that as embedding each categorical feature turns it to a 2-dimensional vector, we would need to combine it with the continuous features, which are 1D. We have explored two options: first would be to simply "flatten" the embeddings:

$$(\underbrace{x_1, \dots, x_n}_{\text{continuous}}, \underbrace{a, \dots, d}_{\text{categorical}}) \underset{\text{embed}}{\Longrightarrow} (\underbrace{x_1, \dots, x_n}_{\text{continuous}}, \underbrace{\begin{pmatrix} e_1^{a^j} \\ \vdots \\ e_{d_h}^{a^j} \end{pmatrix}, \dots, \begin{pmatrix} e_1^{d^j} \\ \vdots \\ e_{d_h}^{d^j} \end{pmatrix}}_{\text{categorical}}) \tag{2.48}$$

$$\underset{\text{flatten}}{\Longrightarrow} (\underbrace{x_1, \dots, x_n}_{\text{continuous}}, \underbrace{e_1^{a^j}, \dots, e_{d_h}^{a^j}, \dots, \ e_1^{d^j}, \dots, e_{d_h}^{d^j}}_{\text{categorical}}), \tag{2.49}$$

while the second option would be to map each continuous feature x^i to a d_h-dimensional space as well:

$$x_i \underset{embed}{\Longrightarrow} x_i \, \boldsymbol{w}^i + \boldsymbol{b}^i = \boldsymbol{e}^{x_i} . \tag{2.50}$$

$$(\underbrace{x_1, \dots, x_n}_{\text{continuous}}, \underbrace{a, \dots, d}_{\text{categorical}}) \underset{\text{embed}}{\Longrightarrow} (\underbrace{\begin{pmatrix} e_1^{x^1} \\ \vdots \\ e_{d_h}^{x^1} \end{pmatrix}, \dots, \begin{pmatrix} e_1^{x^n} \\ \vdots \\ e_{d_h}^{x^n} \end{pmatrix}}_{\text{continuous}}, \underbrace{\begin{pmatrix} e_1^{a^j} \\ \vdots \\ e_{d_h}^{a^j} \end{pmatrix}, \dots, \begin{pmatrix} e_1^{d^j} \\ \vdots \\ e_{d_h}^{d^j} \end{pmatrix}}_{\text{categorical}}) . \tag{2.51}$$

For the FT-Transformer, the Equation 2.51 is necessary (subsection 2.3.5), as each attention layer operates on an array of 2D tokens, as opposed to an array of 1D features. For the ResNets, however, we have found that such embeddings were completely unnecessary, suggesting that categorical features were not particularly useful in prediction.

Furthermore, the dataset contains missing or invalid values for some samples. Some[7] features use -1 to indicate a missing value, some[8] use -99, and some[9] use -999. To properly handle these, we introduce a separate category for them when the feature is categorical (as an invalid value is essentially an extra unique value, so the number of unique values for the feature would be automatically increased by 1). For continuous features, we replace the missing values with a learnable parameter[10] w_i^{NaN} for each feature x_i:

$$x_i \xrightarrow[\text{handle invalid values}]{} \begin{cases} w_i^{\text{NaN}} & \text{if } x_i \text{ is missing or invalid} \\ x_i & \text{otherwise} \end{cases} . \tag{2.52}$$

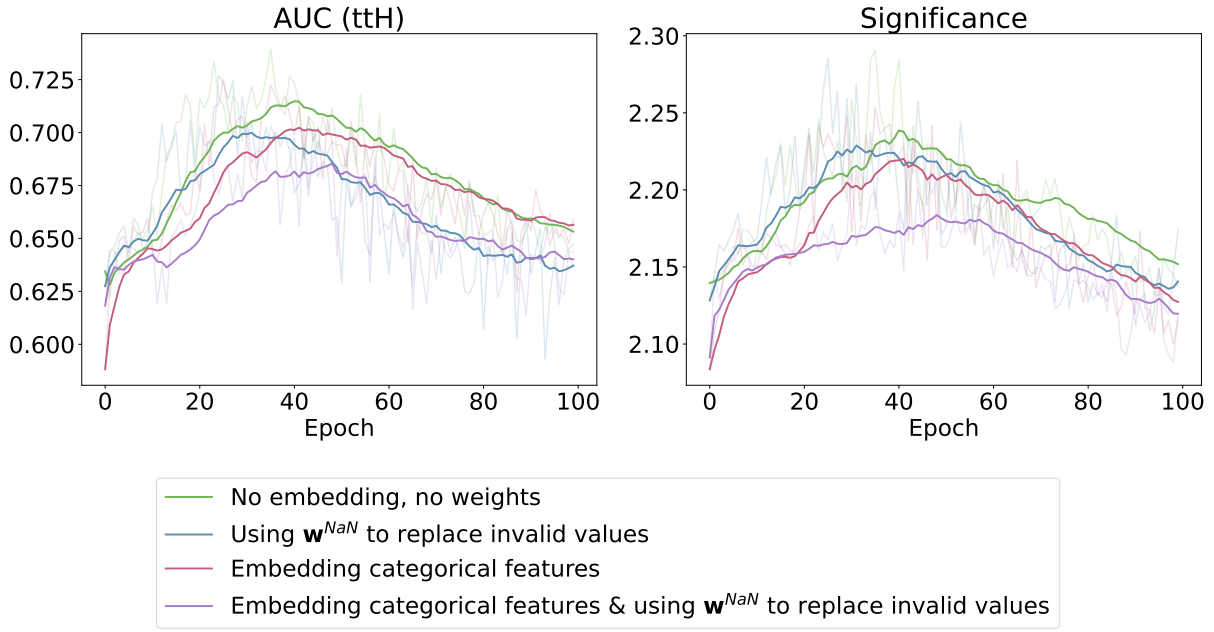


Figure 2.6: Impact of using learnable parameters for missing/invalid values and/or embedding categorical features for ResNets. The semi-transparent lines show the real values, while the solid lines show the exponential moving average.

Overall, with the 8 blocks ResNet with an embedding size of 64 we were able to achieve a

[7] lep_nTrackParticles_0, lep_nTrackParticles_1

[8] taus_passJVT_0

[9] lep_nInnerPix_0, lep_nInnerPix_1, lep_Mtrktrk_atConvV_CO_0, lep_Mtrktrk_atPV_CO_1, lep_Mtrktrk_atPV_CO_0, lep_Mtrktrk_atConvV_CO_1

[10] We have experimented with also setting such values to zero. We didn't observe any notable difference with ResNets, but for FT-Transformer it seemed to have been an important optimization to make (Figure 2.6, Figure 2.7).

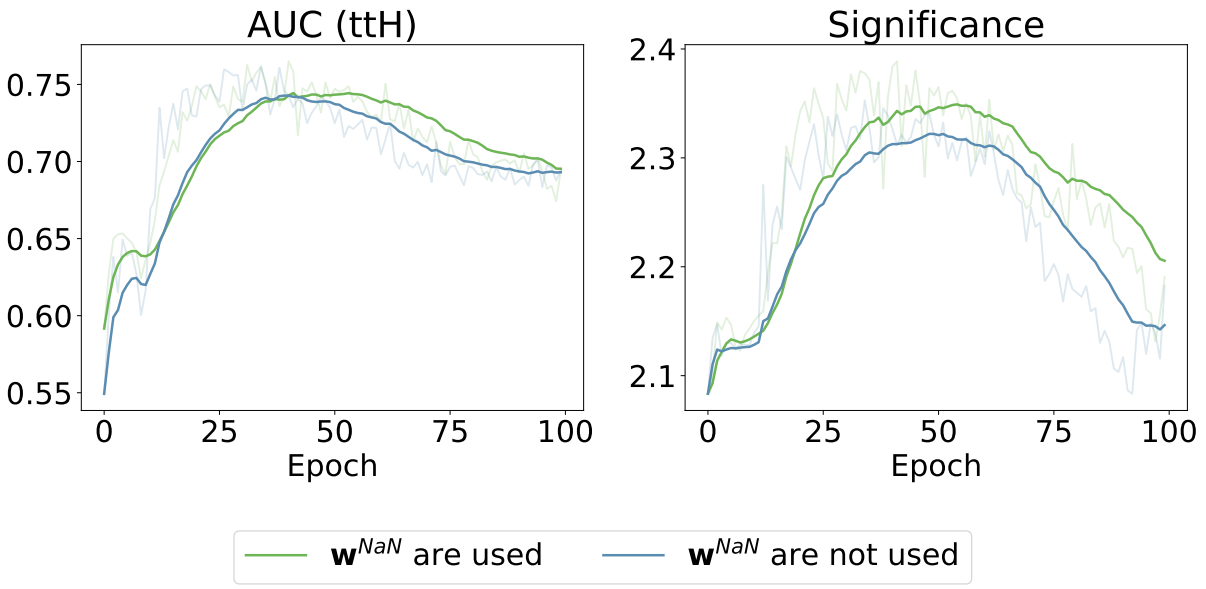


Figure 2.7: Impact of using learnable parameters for missing/invalid values on FT-Transformers. FT-Transformerwith 2 blocks trained on the standard training set $\mathcal{T}^{\text{trn}}$ was used. The semi-transparent lines show the real values, while the solid lines show the exponential moving average.

significance $Z = 2.29$, $\text{AUC}_{t\bar{t}H} = 0.74$, $\text{AUC}_{\text{mean}} = 0.83$. The results are summarized on the Figure 2.13.

2.3.5 Feature Tokenizer + Transformer

We adopt the **F**eature **T**okenizer + **Transformer** (FT-Transformer) as our primary architecture. The architecture was proposed in [21] and is an adaptation of the famous transformer architecture [14] to the tabular data.

Transformer Architecture

The transformer architecture was originally proposed for the NLP processing tasks, but has since been applied to almost every domain of machine learning. The transformer architecture is a fully-attentional architecture, which means that it does not use any convolutional [28] or recurrent [29] layers. Instead, it uses the attention[11] [12] mechanism [30] to learn the dependencies between the input features.

The crux of the transformer's power lies in its scaled dot-product attention mechanism, which allows the model to weigh the importance of different input features relative to

[11]Here we use self-attention, which means that the keys and values are produced from the same source as queries.

[12]We use the *scaled* attention, which refers to a division of the scores by $\sqrt{d_k}$ before applying the softmax.

each other. It can be thought of as a method to compute a weighted sum of values based on their relevance to a given query. Attention requires all the features to be mapped into the $\boldsymbol{X} \in \mathbb{R}^{d_h \times n_{\text{features}}}$ subspace. Then, for all the features, the so-called query, keys, and values are computed:

$$\boldsymbol{Q} = \boldsymbol{W}_Q \boldsymbol{X} \tag{2.53}$$

$$\boldsymbol{K} = \boldsymbol{W}_K \boldsymbol{X} \tag{2.54}$$

$$\boldsymbol{V} = \boldsymbol{W}_V \boldsymbol{X}\,, \tag{2.55}$$

where $\boldsymbol{W}_Q, \boldsymbol{W}_K \in \mathbb{R}^{d_k \times d_h}, \boldsymbol{W}_V \in \mathbb{R}^{d_v \times d_h}$ are the trainable projection matrices. The dimensionality of the query, key, and value vectors is d_k, d_k, d_v, respectively[13].

The output of the whole layer is computed as:

$$\texttt{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_h}})\boldsymbol{V}\,. \tag{2.56}$$

First, the dot-product of the queries and keys is computed and scaled by $\frac{1}{\sqrt{d_k}}$. For large values of d_k this helps to avoid the softmax function from saturating. Then the softmax function is applied to obtain the attention weights[14]. Finally, the dot-product is computed with the values $\boldsymbol{V}$ to obtain the output of the layer.

This mechanism enables the transformer to focus on different parts of the input data depending on the context provided by the query. In practice, applying multiple such heads in parallel was found to be more effective[15]:

$$\texttt{MultiHeadAttention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \texttt{Concat}(\text{head}_1, \ldots, \text{head}_h)\boldsymbol{W}_O\,, \tag{2.57}$$

$$\text{where head}_i = \texttt{Attention}(\boldsymbol{Q}\boldsymbol{W}_i^Q, \boldsymbol{K}\boldsymbol{W}_i^K, \boldsymbol{V}\boldsymbol{W}_i^V)\,. \tag{2.58}$$

Here, queries, keys, and values, are projected by different matrices $\boldsymbol{W}_i^Q, \boldsymbol{W}_i^K, \boldsymbol{W}_i^V$ to each head i, and then attention is applied in parallel. This allows the model to jointly attend to information from different representation subspaces at different positions. Then, the

[13] We used $d_k = d_v = d_h$, which is a fairly standard practice.

[14] Optionally, masking can be applied before softmax. For example in the NLP, when predicting the next token, the model is only allowed to look at the previous tokens, so all the tokens after the current position are set to $-\infty$.

[15] We used 4 heads throughout all our experiments.

outputs of the heads are concatenated and projected to the desired dimensionality by the matrix $\boldsymbol{W}_O$.

The full FT-Transformer transformer architecture is formalized as follows:

$$\texttt{FT-Transformer}(x) = \texttt{Linear}(\texttt{LayerNorm}(\texttt{FTTBlock}(\dots(\texttt{FTTBlock}(\texttt{Embed}(x)))))_1) \tag{2.59}$$

$$\texttt{FTTBlock}(x) = \texttt{FeedForward}(x + \texttt{MultiHeadAttention}(\texttt{LayerNorm}(x))) \tag{2.60}$$

$$\texttt{FeedForward}(x) = x + \texttt{Dropout}(\texttt{Linear}(\texttt{GELU}(\texttt{Linear}(\texttt{LayerNorm}(x))))) \,. \tag{2.61}$$

The architecture is composed of multiple blocks which are applied sequentially after mapping continuous and categorical features into the embedding space. Each block consists of a `LayerNorm`, `MultiHeadAttention`, and two `Linear` layers. The residual connections are applied after the `MultiHeadAttention`, and one more time after at the end of each block. `LayerNorm` layers are applied before each `MultiHeadAttention`, and additionally before the first `Linear` layer. After the first `Linear` layer, a `GELU` activation is applied. Additionally, `Dropout` is applied in the end of the block before the residual connection. The output of the last block is passed to the `LayerNorm`, and then the first token as taken and passed to the final `Linear` layer to obtain the log probabilities (logits) The whole structure is formalized in Equation 2.59 as well as presented on the Figure 2.8.

Since the original paper [14], not many things have changed with the transformer design. The notable change is that `LayerNorm` layer has been moved from after the `MultiHeadAttention` and `FeedForward` layers (post-norm formulation) to before them (pre-norm formulation). The pre-norm formulation has been shown to be more stable and easier to train [31]. We also adopt this change, following [21].

To conclude, transformer is a powerful architecture. Attention mechanism essentially represent a form of the computation and have shown to be able to approximate a wide range of functions really well. The possibility of parallelization that comes with using modern GPUs makes it possible to train such models very efficiently. The usage of residual connections and `LayerNorm` layers makes it possible to train very deep models.

In our experiments, we have observed that transformers have shown better regularization and better results. On the other hand, training requires much more time and memory compared to the simple ResNets, and produces a larger CO_2 footprint.

2.3.6 Regularization, Ensembling, and Dropout

Regularization is a crucial component in training neural networks to prevent overfitting and improve generalization. From the ERM perspective, regularization can be viewed as

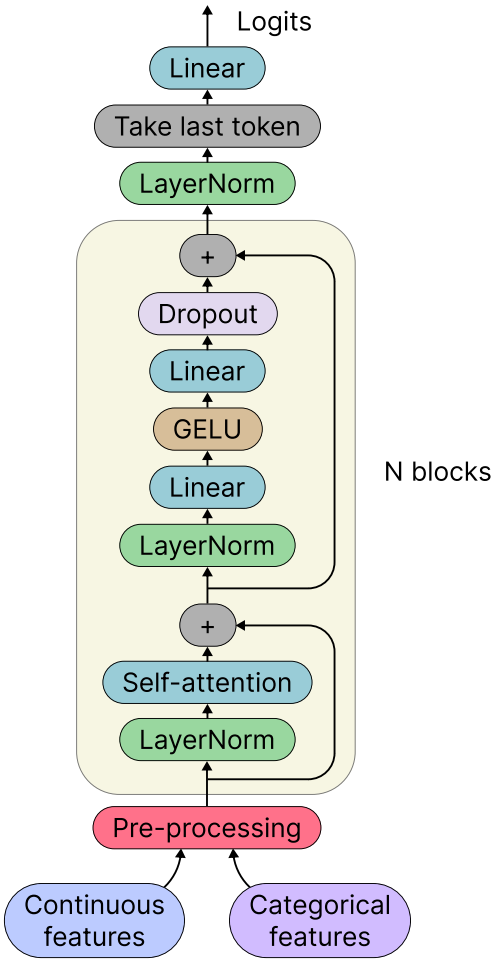


Figure 2.8: FT-Transformer architecture.

constraining the hypothesis space $\mathcal{H}$. Given a hypothesis space $\mathcal{H}$ with VC dimension d, training set $\mathcal{T}^{\text{trn}}$ of size N, and a classifier $h^{\mathcal{T}^{\text{trn}}}$ training on that training set, for any $\delta > 0$ the following bound:

$$\underbrace{R(h^{\mathcal{T}^{\text{trn}}}) - R_{\mathcal{T}}^{\text{trn}}(h^{\mathcal{T}^{\text{trn}}})}_{\text{generalization error}} \leq 2\sqrt{\frac{2d \log \frac{2N}{d} + 2 \log \frac{2}{\delta}}{N}} \tag{2.62}$$

holds with probability at least $1 - \delta$ [32].

From this inequality, it follows that the generalization error can be reduced by increasing the size of the training set[16], or by reducing the VC dimension of the hypothesis space[17].

[16] Because $\lim_{N \to \text{inf}} \frac{\log N}{N} = 0$.

[17] $\frac{\partial}{\partial d} d \log \frac{2N}{d} = \frac{2N}{d} - 1$, which is larger than 0 when $d < \frac{2N}{e}$. In practice, it is almost always true, it is uncommon to have VC dimension as disproportionally large, compared to the training set size.

The latter can be achieved by constraining the hypothesis space, which is precisely what regularization does.

An example of a regularization technique is *weight decay*, which operates by adding a penalty term to the loss function that penalizes large weights. This penalty term is typically proportional to the L_2 norm of the weights $\boldsymbol{w}$, and the regularization strength is controlled by a hyperparameter λ. We use a standard value of 0.01 throughout our experiments.

Another extremely popular regularization method is *dropout*, which operates by randomly dropping out, or "turning off", a proportion of the neurons during training. Introduced by [25], this technique is simple and computationally efficient, and it has been widely adopted in the deep learning community.

The dropout rate, the proportion of neurons to drop, is a hyperparameter that requires tuning. A moderate dropout rate (e.g., 0.5) introduces noise into the training process, which helps prevent the model from memorizing the training data and potentially improves generalization. However, a high dropout rate may hinder the learning process by adding too much noise, while a low dropout rate may not provide sufficient regularization. Generally, for input neurons a dropout rate of 0.2 is recommended, while for hidden neurons a dropout rate of 0.5 is recommended [25]. With FT-Transformers, we have found that applying a dropout rate of 0.2 for all the layers worked best.

From the perspective of model ensembling, dropout can be viewed as a way to implicitly create an ensemble of different "thinned" networks, which share parameters. The output of the network with dropout can be seen as an averaged prediction of these thinned networks. Ensembling typically provides a boost in model performance by aggregating predictions of diverse models, thus reducing the risk of overfitting to specific patterns in the training data.

In our experiments, applying dropout to the neural network architectures led to an improvement in generalization, reflected by a decrease in the gap between training and validation performance metrics, decreased validation loss and increase in the Area Under the Curves (AUCs) of the ROCs curves. The detailed results are presented in the Figure 2.13.

Despite its advantages, dropout does introduce an additional layer of randomness into the training process, making the convergence slower and sometimes harder. It is crucial to first ensure that the model can fit the training data closely, even if it overfits, before applying dropout. Also, while dropout can improve generalization, it does not replace the need for sufficient training data (see the next section), careful feature selection, and

other components of a successful machine learning project.

2.4 Increasing Statistics by Dropping the Cuts

As seen from the Equation 2.62, one of the best ways to improve the generalization of the classifier is to increase the number of samples. This is the most reliable way to improve the performance of the classifier.

In [7], authors explore why deep neural networks despite having shown a great performance on a variety of tasks such as computer vision, natural language processing, and speech recognition, have not been widely adopted in the tabular data domain. The main reason is that the tabular data is very sparse, and the number of samples is very small. This makes it difficult to train a deep neural network which would generalize well. Random forests and gradient boosting methods perform much better in this domain. However, as the number of samples increases, the performance of deep neural networks improves and becomes comparable to the other methods.

As noted before (Table 1.2), the number of samples in the SR is very small to allow for a reliable training of deep NNs. Moreover, some classes such as $t\bar{t}$ are extremely underrepresented, where the SR contains only 3 samples of them. Furthermore, the relative event weight $\frac{n_{\text{raw}}}{n_{\text{weighted}}}$ associated with these events is usually very high, hence misclassifications impact the metrics severely in a negative way.

Ultimately, our goal is to train a classifier, that would be able to perform well on the events inside our SR. However, we can exploit the fact that events outside the SR likely share the same underlying physics and are not completely unrelated to the SR. In other words:

1. We suppose there is a joint distribution, $P_{\text{SR}}(\boldsymbol{x}, y)$, from which we have sampled a training dataset $\mathcal{T}^{\text{trn}} \sim P_{\text{SR}}(\boldsymbol{x}, y)$. All the previous works have used such training set.

2. We further suppose that there is a joint distribution $P(\boldsymbol{x}, y)$, such that $P_{\text{SR}} \subset P(\boldsymbol{x}, y)$. This is a more general distribution, which represents more general physics, related to all the events.

3. Thus, we suppose that by learning from the more general distribution P would allow us to generalize better, while having more samples (statistics) to train on. We thus propose to include these samples, which lie outside the signal region into the training set, and to train the classifier on the extended training set $\mathcal{T}^{\text{trn}}_{\text{ext}}$, while keeping the validation set the same.

A somewhat similar approach was proposed by the BDT group[18], where some cuts[19] were dropped to obtain higher number $t\bar{t}$ samples. Similarly to our work, the group has reported an increased classification performance.

An important detail is, however, accounting for the event weights. When we change the cut expression, the distribution of events changes as well. We thus need to apply the correct scaling to each class to keep the total number of weighted events the same as in the SR. This is precisely what is done in the BDT working group. In our experiments, however, we have not observed any particular difference with training with or without weights, however, the subject requires more investigation (section 2.6).

We should be careful to keep the validation set the same as it was (composed solely of events inside SR), as we would like the evaluations to be unaffected by the change of the cut expression. Otherwise, we would be potentially reporting results on very different region than SR or even $2l_{SS} + 1\tau_{\text{had}}$. Figure 2.9 illustrates the relationship between the standard, extended training sets, as well as the validation set.

We have observed experimentally that extending the training set has a significant impact on the performance. There is an increase along all the metrics (Figure 2.13). The most significant improvement is observed for the underrepresented classes, such as $t\bar{t}$. Even though we are most interested in the discriminative performance for the $t\bar{t}H$, we have observed that correctly recognizing each background class has a positive impact on the accuracy of the prediction for the $t\bar{t}H$ as well. Furthermore, a model that has a good performance on differentiating between all the classes has more potential in a subsequent fine-tuning to the binary classification (section 2.7).

2.5 Effect of the Reduced Training Set Size

Our extended training set contains roughly 8.7 million events. Combined with the fact that our NN is quite large as well (about 4 million learnable parameters), the training takes a long time. We have thus decided to investigate the effect of the reduced training set size on the performance of the classifier, hoping that we can reduce the training time without sacrificing too much performance. The results are summarized on the Figure 2.10.

We have performed 4 experiments with 100%, 50%, 10%, and 5% of the extended training set $\mathcal{T}^{\text{trn}}$. The results are shown in Figure 2.10. During each experiment, we kept the same random seed to keep the initialization of the NN the same. We also kept the same batch size and learning rate. The results show that the performance of the classifier does

[18]Private Communication, August 2023, Nello Bruscino.
[19]PLIV cuts, specifically - Appendix A.2

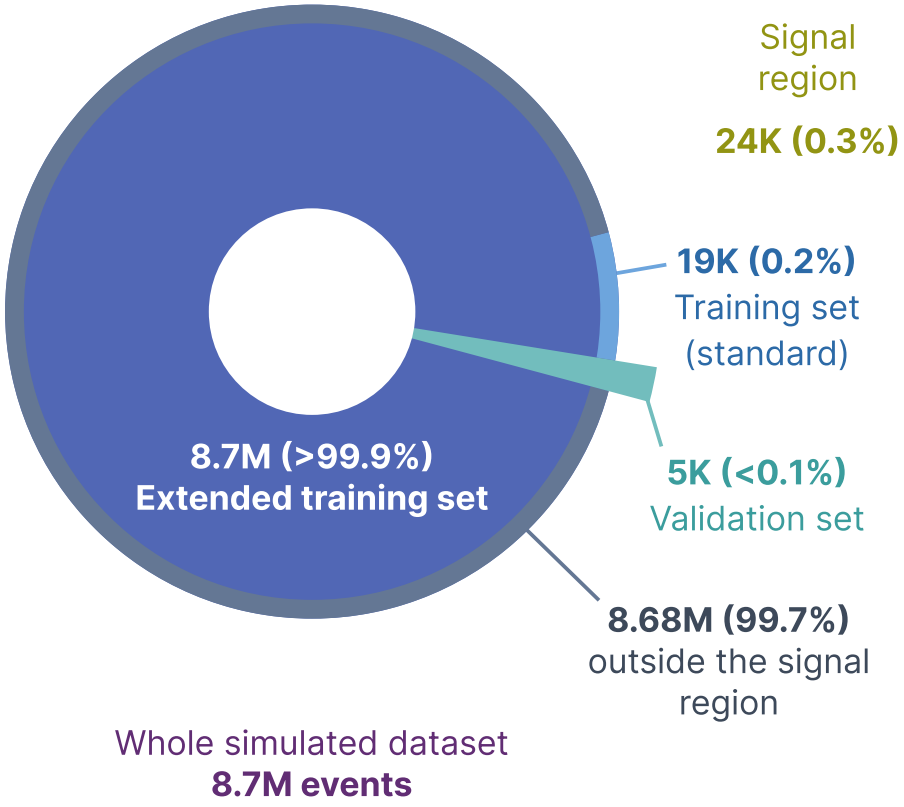


Figure 2.9: Training and validation sets, as well the extended training set. Extended training set is mostly (99.7%) comprised of events outside the SR. The validation set is kept the same as before, to ensure that the evaluation is not affected by the change of the cut expression. The training-validation split is 80-20%.

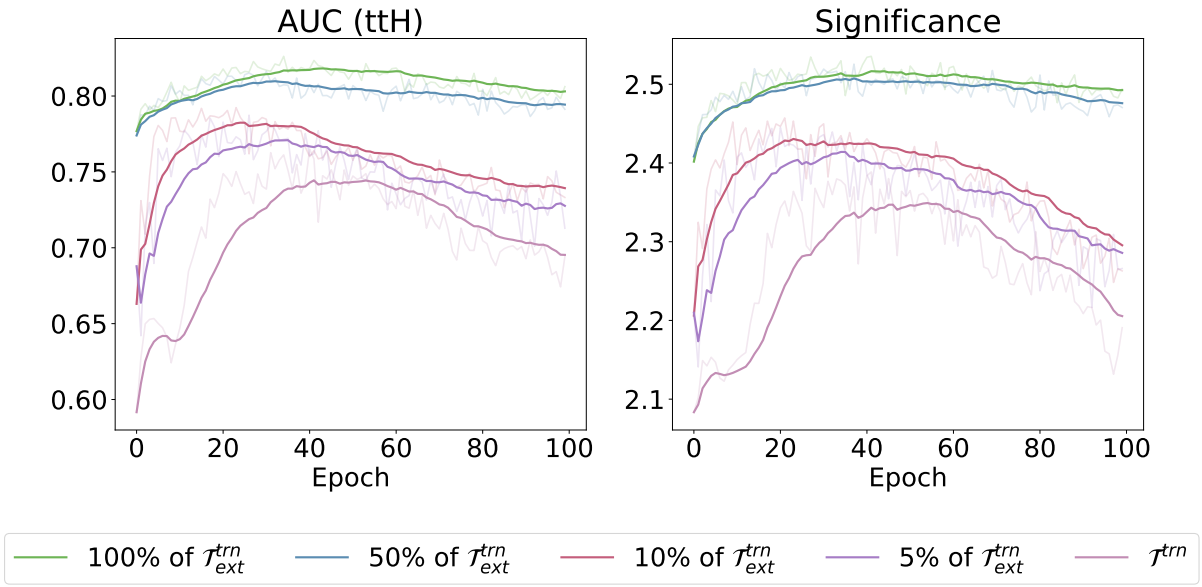


Figure 2.10: Effect of reduced training set size on classifier performance. FT-Transformer with 2 blocks was used. The semi-transparent lines show the real values, while the solid lines show the exponential moving average.

not degrade significantly from 100% and 50%. Such training set size of about 4 million events seem adequate to obtain some sufficiently good results. Of course, for maximum performance, it is best to use the full training set. When dropping to 10% or 5% of the training set, the performance degrades significantly, but is nevertheless much better than when the standard training set $\mathcal{T}^{\text{trn}}$ is used.

2.6 Weights in the Training - Different Approaches

As we described before, each event coming from the MC simulation has an associated weight so that the distribution of the simulated sample matches the distribution of the real data (section 1.6). As described in section 2.2, weights should always be used during evaluation.

Regarding the training, as described in subsection 2.1.4, the Cross-Entropy loss function allows to assign a weight to each class that would put more or less emphasis on it. In [6] the authors trained with using the event weights as in Equation 2.22, however, the weights were per-batch normalized to sum to 1.

In our analysis we have experimented with different approaches:

1. **No weights** - the weights were not used at all.
2. **Per-sample weights** - the weights were used as in [6].

3. **Per-sample weights, not normalized** - the weights were used as in [6] but without the per-sample normalization (Equation 2.22).
4. **Raw class-imbalance weights** - the total number of *raw* samples was divided by the number of raw *samples* in each class. Then the weights were normalized (Equation 2.20).
5. **Weighted class-imbalance weights** - the total number of *weighted* samples was divided by the number of *weighted* samples in each class. Then the weights were normalized (Equation 2.20).
6. **Raw class-imbalance weights & per-sample not normalized weights**: item 3 & item 4
7. **Weighted class-imbalance weights & per-sample not normalized weights**: item 3 & item 5

Results are shown on the Figure 2.11. Any of the weighting methods did not produce any substantial increase in performance. Further investigation of the subject is suggested. We have decided to train without weights for simplicity.

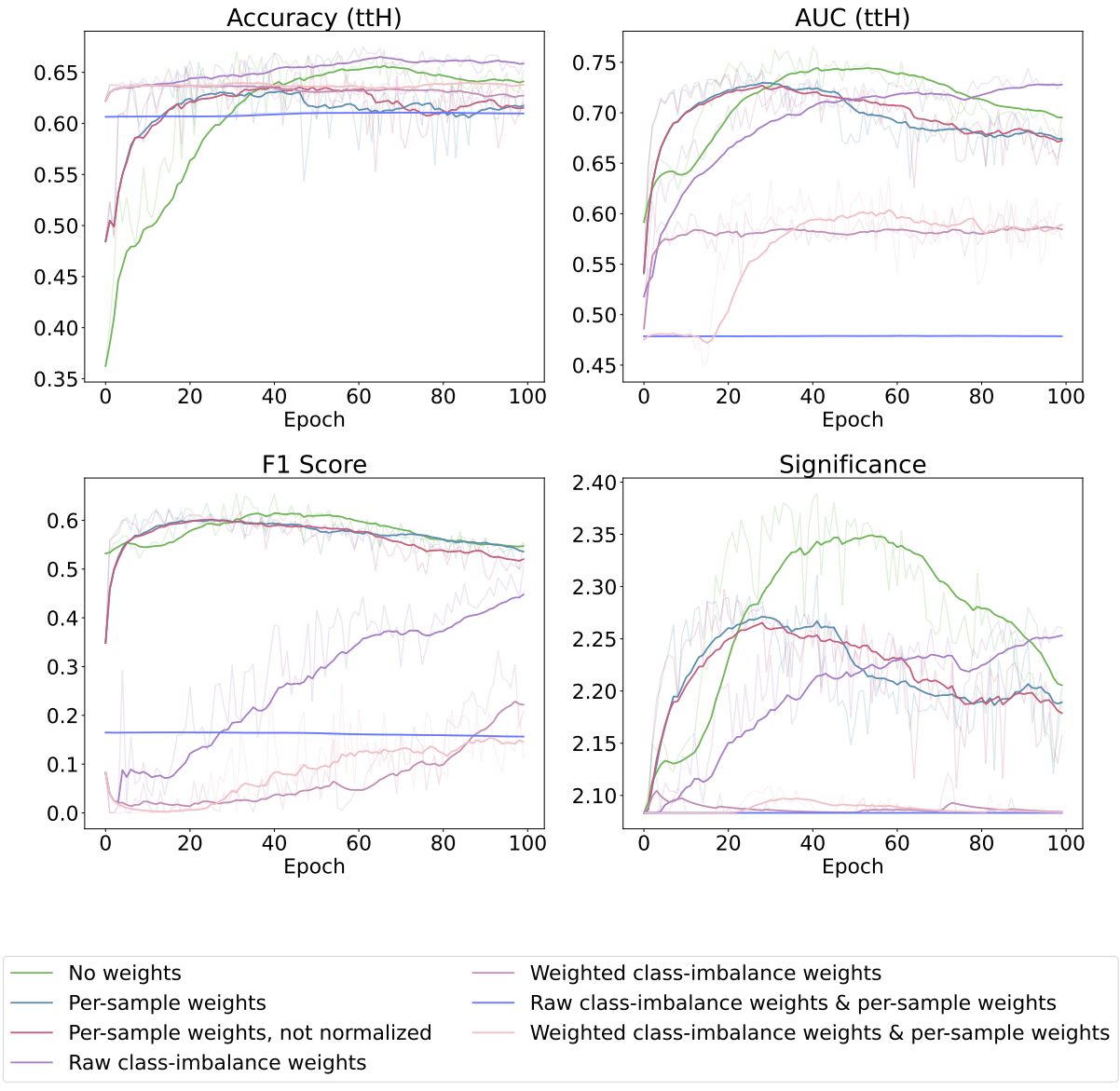


Figure 2.11: Comparison of different weighting strategies. FT-Transformer with 2 blocks was used on the standard training set $\mathcal{T}^{\text{trn}}$. The semi-transparent lines show the real values, while the solid lines show the exponential moving average.

2.7 All Classes versus Signal and Background Only

In [5], the authors experimented with both multiclass and binary classification formulations. The multiclass formulation is a direct approach where the model is trained to differentiate among all classes simultaneously. In contrast, the binary formulation is a specialized approach where the model is trained to distinguish only between signal and background classes. All classes, except for the signal, are treated as background. This approach is motivated by the hypothesis that the model can concentrate more of its resources on the primary task of signal and background discrimination.

For [5], based solely on the reported significance, the results were quite similar[20] for both formulations. However, our observations suggest that when the model is trained to differentiate among all classes, it demonstrates enhanced learning capabilities[21] and can potentially extract more information from the input data (Figure 2.12). This approach, however, also allocates resources towards distinguishing individual background classes, which might not be essential for the primary task of signal and background discrimination.

Consequently, we suggest considering an alternative strategy, which we refer to as "two-phase training," to utilize the model's resources more effectively. Initially, it would involve training the model on all classes. Once satisfactory performance is achieved, one could transition to the binary formulation. In practical terms, this would mean no longer penalizing the model for misclassifying among the background classes (e.g., if the true class is $t\bar{t}W$ and the predicted one is $t\bar{t}Z$, the prediction would still be deemed correct since both processes are background). This strategy could potentially allow the model to focus on the primary task of signal and background discrimination after gleaning sufficient information from the distinctions among all classes. The model would first learn the patterns in the underlying physics comprehensively and then hone its focus on the primary task.

The results are summarized in Figure 2.12, showcasing the progress in AUC and accuracy for differentiating signal from background. It is evident that the multiclass formulation outperforms the binary one, and we hypothesize that this performance could be further enhanced by the two-phase training strategy.

[20] Based solely on significance, it is challenging to conclude which approach is superior.
[21] Based on accuracy, AUC, and significance.

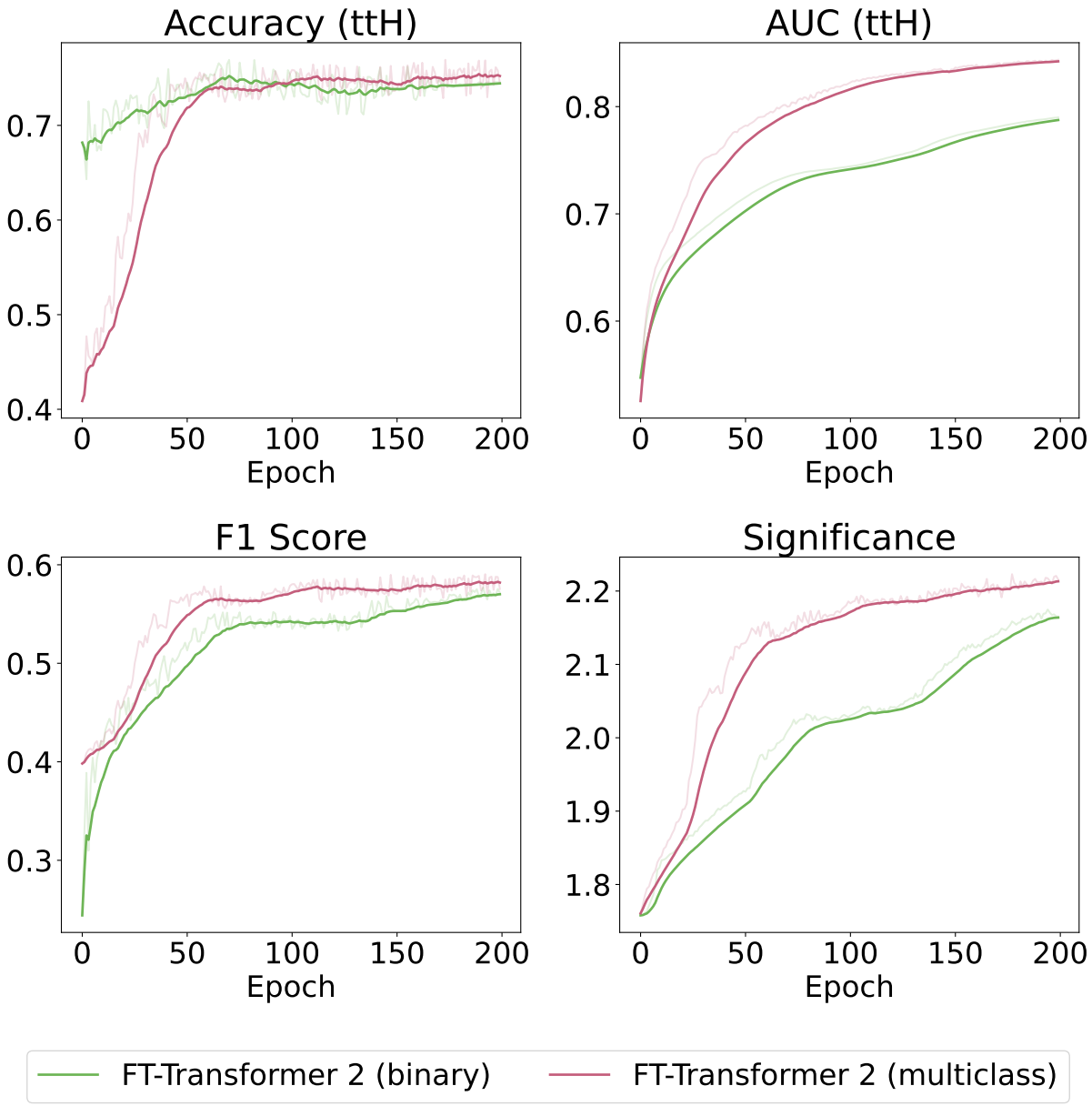


Figure 2.12: Comparison of the multiclass and binary formulations. The semi-transparent lines show the real values, while the solid lines show the exponential moving average.

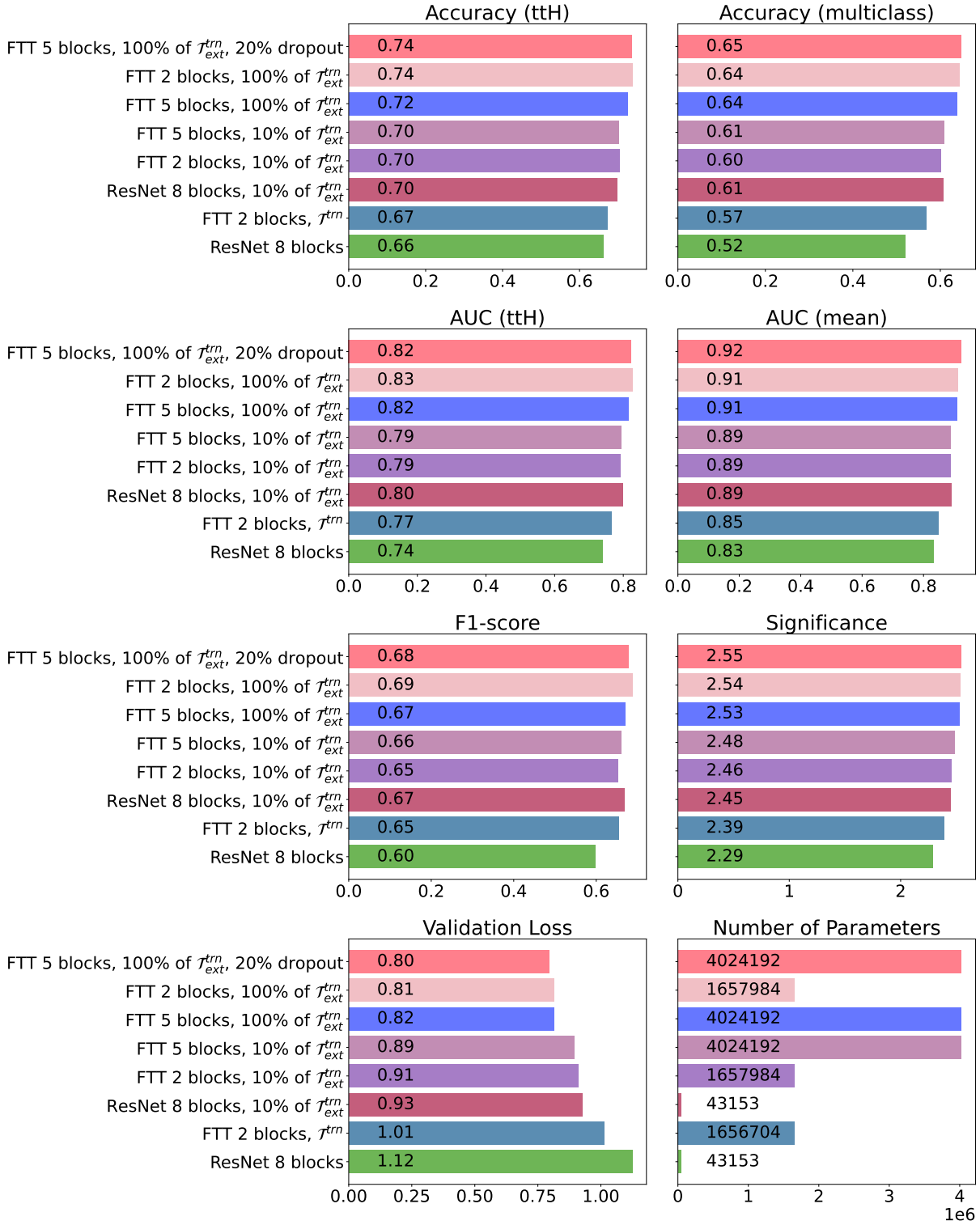


Figure 2.13: Results of the different architectures and configurations. FTT refers to the FT-Transformer.

Chapter 3

Evaluation of Uncertainties

This section presents the estimation of the statistical and systematic uncertainties on the median signal strength μ, where μ is defined as a scale factor, applied to the number of the signal events, such that the sum of the scaled signal s and background b events matches the observed data n:

$$b + \mu \cdot s = n\,. \tag{3.1}$$

Estimation of the uncertainties on the μ is done by performing a fit to the Asimov dataset [3], where we assume $n = b + s$, thus $\mu = 1$. The Asimov dataset is a representative data set that provides a simple method to obtain the median experimental sensitivity of a search or measurement as well as fluctuations about this expectation. It is used to estimate the median significance by replacing the ensemble of simulated data sets by a single representative one.

To perform statistical testing, the binned profile likelihood method is used. The binned profile likelihood is a statistical method commonly used in high-energy physics, especially in the context of searches for new phenomena or precision measurements. It is a variant of the profile likelihood method adapted for histogram-like (binned) data.

Each bin has a count representing the number of events or occurrences in that bin. When applying the profile likelihood method to binned data, one constructs a likelihood based on the expected number of events in each bin and the observed number of events. The expected number of events is typically a function of both the parameters of interest and the nuisance parameters.

The likelihood for each bin i, given a signal strength parameter μ, background b, and observed data n is modeled as a Poisson distribution:

$$L(\mu, \boldsymbol{\theta}) = \prod_{j=1}^{N} \frac{(\mu s_j + b_j)^{n_j}}{n_j!} e^{-(\mu s_j + b_j)} \prod_{k=1}^{M} \frac{u_k(\boldsymbol{\theta})^{m_k}}{m_k!} e^{-u_k(\boldsymbol{\theta})} \tag{3.2}$$

The first part of the equation models the likelihood of observing the parameter of interest μ given the binned distribution. Here N is the number of bins, s_j is the number of signal events in bin j, b_j is the number of background events in bin, $n_j = s_j + b_j$ is the total number of events in bin j. We have used the distribution of the NN output (probability of event being $t\bar{t}H$, formally the posterior $p(y = t\bar{t}H|\boldsymbol{x})$)[1] as the binned distribution (Figure 3.1), however any variable can be used in principle. The automatic binning was used in order to make sure there is a relatively equal number of events in each bin.

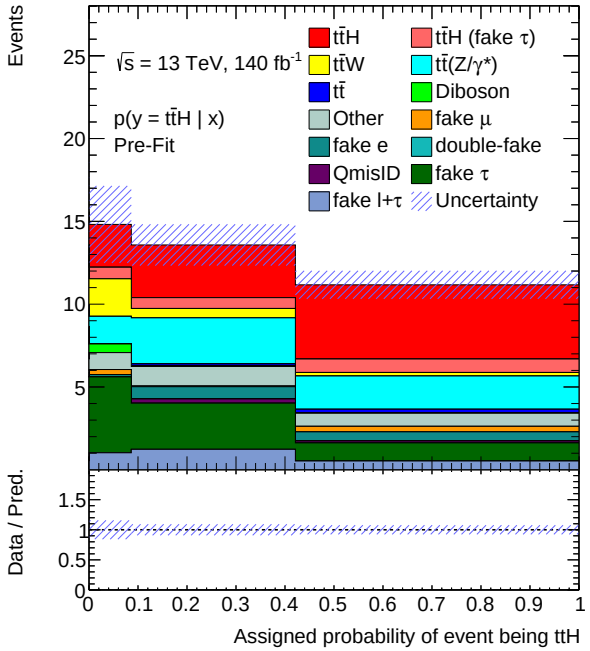


Figure 3.1: NN output for the signal and background events.

The second part of the equation models the likelihood of observing the nuisance parameters $\boldsymbol{\theta}$ given some control sample. Here M is the number of bins in the control sample, m_k is the number of events in bin k, and $u_k(\boldsymbol{\theta})$ are calculable quantities that depend on the nuisance parameters $\boldsymbol{\theta}$ [3]. When maximizing the likelihood with respect to the nuisance parameters, such $\boldsymbol{\theta}$ are taken, which produce the maximum values for the quantities $u_k(\boldsymbol{\theta})$. Essentially, this allows us to generate a control sample (for example with a different MC generator) and use it to estimate the nuisance parameters.

[1] To achieve this, one must either inject the custom feature - NN output in our case - into the existing n-tuples, or use the so-called friend trees. The approach is described in the Appendix A.6

3.1 Statistical Uncertainties

Statistical uncertainties emerge from the inherently stochastic nature of the data collection process, particularly due to the limited size of our data sample. This type of uncertainty, which decreases as we accumulate more data, is tied to our estimate of the significance of the $t\bar{t}H$ signal.

The best fit results when only statistical uncertainties corresponds to the measurement precision shown in Figure 3.2.

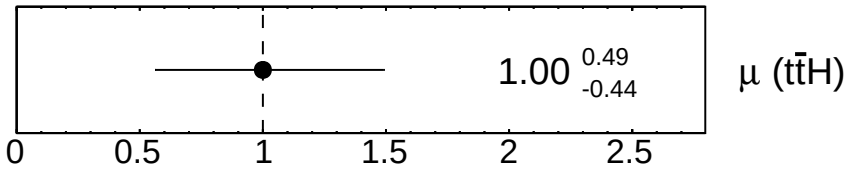


Figure 3.2: Expected uncertainties on the median signal strength μ for the $t\bar{t}H$ process including statistical uncertainties only.

3.2 Systematic Uncertainties

Systematic uncertainties in particle physics originate from a wide range of sources, with several relevant to our analysis:

- Luminosity Uncertainty: The luminosity of the accelerator, or the number of particles within a beam per unit area, is a fundamental parameter in any experiment. Uncertainties in the measurement of luminosity can translate into uncertainties in the overall scale of the data.
- Electron and Muon Uncertainty: The identification, reconstruction, and isolation of electrons and muons can have associated uncertainties. Differences in efficiencies between data and simulation can result in systematic errors.
- Next Leading Order (NLO) Uncertainty: Predictions of the rates for various processes are typically calculated to leading order (LO) or next-to-leading order (NLO) in perturbation theory. The precision of these predictions is limited by the order to which they are calculated, with higher-order terms introducing potential systematic uncertainties.
- Final State Radiation (FSR) and Initial State Radiation (ISR) Uncertainties: These uncertainties are associated with the additional emission of photons from the initial

or final state particles. While these effects are included in simulations, they are based on theoretical models and can have associated uncertainties.

- Modeling Uncertainties for Each Class: Each class or category of events you're analyzing (e.g., $t\bar{t}H$, $t\bar{t}W$, $t\bar{t}Z$, etc.) may have associated modeling uncertainties. These arise due to potential differences between the simulation and the actual experimental data. The uncertainties can stem from the choice of event generator, the specific model assumptions for the process in question, or the parameters used in the simulation.
- Cross-Section Uncertainties: The cross-section of a process is a measure of the likelihood of that process occurring. In particle physics, theoretical calculations predict these cross-sections, but they come with uncertainties. These uncertainties can be due to missing higher-order terms, variations in parton distribution functions, or other theoretical approximations.
- Generator Uncertainties: These are uncertainties associated specifically with the event generators (e.g., Pythia, Sherpa[2]). Event generators use a myriad of theoretical approximations and models to simulate particle collisions. Each generator has its strengths and weaknesses, and switching from one to another or even using different versions/settings of the same generator can yield different results.

When these systematic uncertainties are combined with the statistical ones, the measurement precision shown in Figure 3.3.

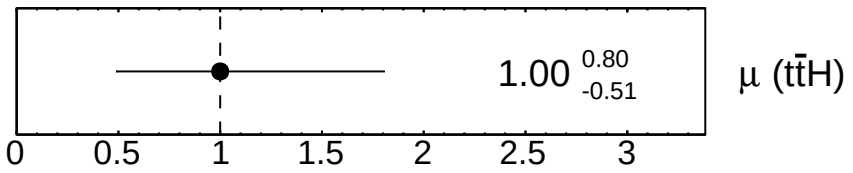


Figure 3.3: Expected uncertainties on the median signal strength μ for the $t\bar{t}H$ process including statistical and systematic uncertainties.

An important aspect of the analysis of systematic uncertainties is the ranking plot. The ranking plot shows the relative impact of each systematic (or statistical - denoted by γ) uncertainty on the final result. The ranking plot for the $t\bar{t}H$ process is shown in Figure 3.4. The uncertainties are ranked in descending order of their impact on the final result. The top 20 uncertainties are shown.

[2]https://sherpa-team.gitlab.io/

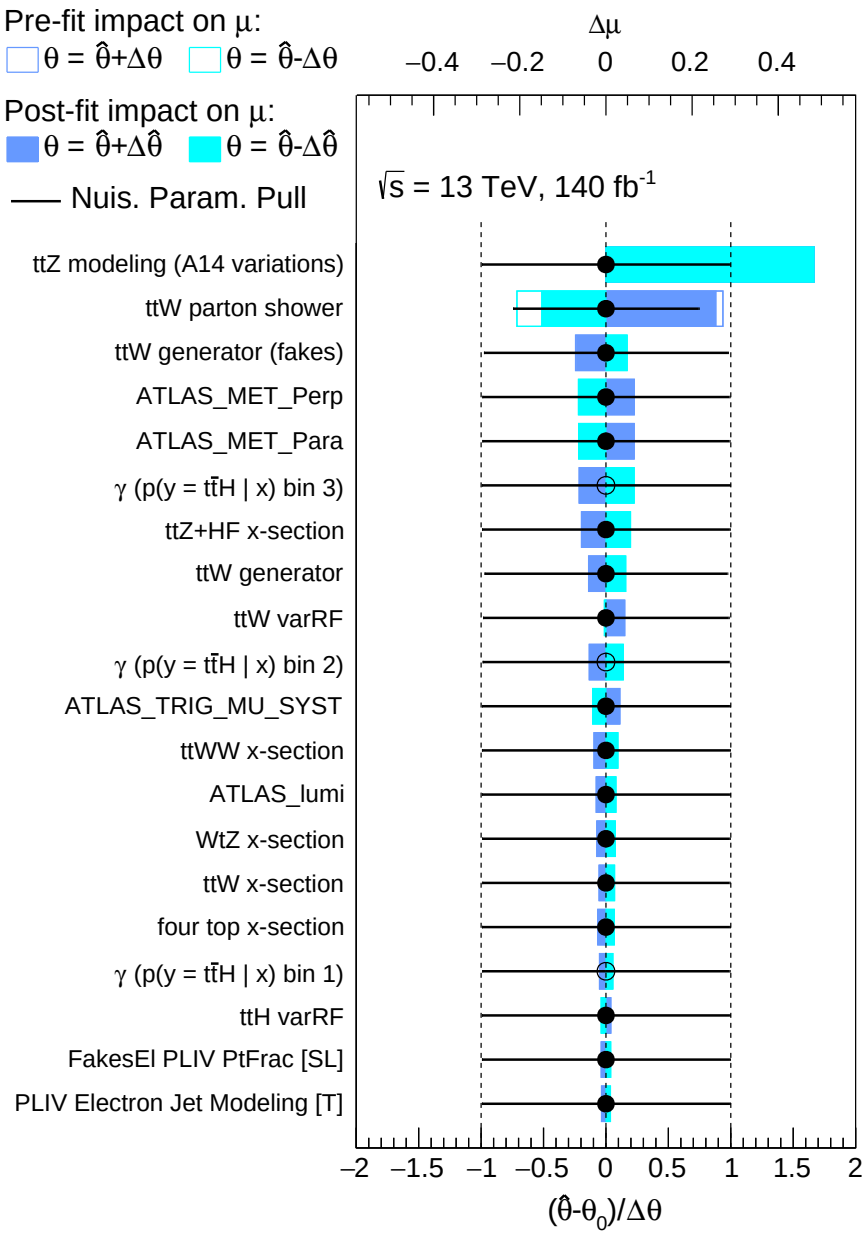


Figure 3.4: Ranking plot for systematic uncertainties.

These results on systematic uncertainties are preliminary and further studies are ongoing.

Conclusions

In this thesis, we have developed and employed advanced machine learning techniques to improve the identification of the $t\bar{t}H$ process. The results of this work advance upon previous efforts in the field, demonstrating the potential of machine learning to enhance the identification of the $t\bar{t}H$ process.

This research explored the usage of a transformer architecture, adapted for the tabular data (FT-Transformer), particularly in the domain of particle physics and event selection.

Additionally, the thesis investigated the impact of dropping the selection cuts and training on the extended training set. Combined with the large NN models, this approach yielded a significant increase in performance, showcasing the importance of a substantial and diverse dataset for model training in particle physics.

The evaluation of systematic uncertainties was a new step in the $t\bar{t}H$ analysis, the ranking of the uncertainties was performed, and the measurement precision on the median strength parameter μ was estimated.

The binary and multi-class prediction problems were investigated, and it was found that differentiating between all the classes yielded better results. Furthermore, the two-phase training process is proposed, where the model is initially trained on the multi-class prediction problem and subsequently switched to binary classification.

The measurement precision on the median signal strength parameter μ was estimated as $\mu = 1 + 0.49 \,/\, - 0.44$ with statistical uncertainties only, and $\mu = 1 + 0.80 \,/\, - 0.51$ when systematic uncertainties are also taken into account. It is noted that statistical and systematic uncertainties contribute about equally to the final result.

Bibliography

[1] P. W. Higgs, "Broken symmetries and the masses of gauge bosons", *Phys. Rev. Lett.*, vol. 13, pp. 508–509, 16 1964. DOI: 10.1103/PhysRevLett.13.508. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.13.508.

[2] K. Cranmer, *Practical statistics for the lhc*, 2015. arXiv: 1503.07622 [physics.data-an].

[3] G. Cowan, K. Cranmer, E. Gross, and O. Vitells, "Asymptotic formulae for likelihood-based tests of new physics", *The European Physical Journal C*, vol. 71, pp. 1–19, 2011.

[4] D. Guest, K. Cranmer, and D. Whiteson, "Deep learning and its application to lhc physics", *Annual Review of Nuclear and Particle Science*, vol. 68, pp. 161–181, 2018.

[5] S. Konig, "Optimization of $t\bar{t}H$ Signal and Background Separation Using Machine Learning in the 2lSS1tau Channel", 2022. [Online]. Available: https://cds.cern.ch/record/2836425.

[6] J. Presperin, "Machine learning for $t\bar{t}H$ mechanism Higgs boson detection from CERN ATLAS data", 2022, Presented 14 Jun 2022. [Online]. Available: https://cds.cern.ch/record/2812400.

[7] L. Grinsztajn, E. Oyallon, and G. Varoquaux, *Why do tree-based models still outperform deep learning on tabular data?*, Jul. 2022. DOI: 10.48550/arXiv.2207.08815.

[8] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system", *CoRR*, vol. abs/1603.02754, 2016. arXiv: 1603.02754. [Online]. Available: http://arxiv.org/abs/1603.02754.

[9] "Random forests", *Machine Learning*, vol. 45, pp. 5–32, Oct. 2001. DOI: 10.1023/A:1010950718922.

[10] OpenAI, *Gpt-4 technical report*, 2023. arXiv: 2303.08774 [cs.CL].

[11] S. Bubeck, V. Chandrasekaran, R. Eldan, *et al.*, *Sparks of artificial general intelligence: Early experiments with gpt-4*, 2023. arXiv: 2303.12712 [cs.CL].

[12] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, *High-resolution image synthesis with latent diffusion models*, 2022. arXiv: 2112.10752 [cs.CV].

[13] "Identification of Jets Containing b-Hadrons with Recurrent Neural Networks at the ATLAS Experiment", CERN, Geneva, Tech. Rep., 2017, All figures including auxiliary figures are available at https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/PUBNOTES/ATL-PHYS-PUB-2017-003. [Online]. Available: https://cds.cern.ch/record/2255226.

[14] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need", *CoRR*, vol. abs/1706.03762, 2017. arXiv: 1706.03762. [Online]. Available: http://arxiv.org/abs/1706.03762.

[15] V. Vapnik, "Principles of risk minimization for learning theory", vol. 4, J. Moody, S. Hanson, and R. Lippmann, Eds., 1991. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/1991/file/ff4d5fbbafdf976cfdc032e3bde78de5-Paper.pdf.

[16] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation", 1986. [Online]. Available: https://api.semanticscholar.org/CorpusID:62245742.

[17] S. Hochreiter and J. Schmidhuber, "Long short-term memory", *Neural computation*, vol. 9, pp. 1735–80, Dec. 1997. DOI: 10.1162/neco.1997.9.8.1735.

[18] D. Kingma and J. Ba, "Adam: A method for stochastic optimization", *International Conference on Learning Representations*, Dec. 2014.

[19] I. Loshchilov and F. Hutter, "Fixing weight decay regularization in adam", *CoRR*, vol. abs/1711.05101, 2017. arXiv: 1711.05101. [Online]. Available: http://arxiv.org/abs/1711.05101.

[20] S. Ö. Arik and T. Pfister, "Tabnet: Attentive interpretable tabular learning", *CoRR*, vol. abs/1908.07442, 2019. arXiv: 1908.07442. [Online]. Available: http://arxiv.org/abs/1908.07442.

[21] Y. Gorishniy, I. Rubachev, V. Khrulkov, and A. Babenko, "Revisiting deep learning models for tabular data", in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34, Curran Associates, Inc., 2021, pp. 18 932–18 943. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2021/file/9d86d83f925f2149e9edb0ac3b49229c-Paper.pdf.

[22] A. F. Agarap, "Deep learning using rectified linear units (relu)", *CoRR*, vol. abs/1803.08375, 2018. arXiv: 1803.08375. [Online]. Available: http://arxiv.org/abs/1803.08375.

[23] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network", *CoRR*, vol. abs/1505.00853, 2015. arXiv: 1505.00853. [Online]. Available: http://arxiv.org/abs/1505.00853.

[24] D. Hendrycks and K. Gimpel, "Bridging nonlinearities and stochastic regularizers with gaussian error linear units", *CoRR*, vol. abs/1606.08415, 2016. arXiv: 1606.08415. [Online]. Available: http://arxiv.org/abs/1606.08415.

[25] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting", *Journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014. [Online]. Available: http://www.cs.toronto.edu/~rsalakhu/papers/srivastava14a.pdf.

[26] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks", *CoRR*, vol. abs/1706.02515, 2017. arXiv: 1706.02515. [Online]. Available: http://arxiv.org/abs/1706.02515.

[27] J. L. Ba, J. R. Kiros, and G. E. Hinton, *Layer normalization*, 2016. arXiv: 1607.06450 [stat.ML].

[28] K. O'Shea and R. Nash, "An introduction to convolutional neural networks", *CoRR*, vol. abs/1511.08458, 2015. arXiv: 1511.08458. [Online]. Available: http://arxiv.org/abs/1511.08458.

[29] A. Sherstinsky, "Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network", *CoRR*, vol. abs/1808.03314, 2018. arXiv: 1808.03314. [Online]. Available: http://arxiv.org/abs/1808.03314.

[30] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate", *CoRR*, vol. abs/1409.0473, 2014. [Online]. Available: https://api.semanticscholar.org/CorpusID:11212020.

[31] R. Xiong, Y. Yang, D. He, *et al.*, "On layer normalization in the transformer architecture", *CoRR*, vol. abs/2002.04745, 2020. arXiv: 2002.04745. [Online]. Available: https://arxiv.org/abs/2002.04745.

[32] M. Reid, "Generalization bounds", in *Encyclopedia of Machine Learning*, C. Sammut and G. I. Webb, Eds. Boston, MA: Springer US, 2010, pp. 447–454, ISBN: 978-0-387-30164-8. DOI: 10.1007/978-0-387-30164-8_328. [Online]. Available: https://doi.org/10.1007/978-0-387-30164-8_328.

[33] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks", in *Proceedings of the 34th International Conference on Machine Learning*, D. Precup and Y. W. Teh, Eds., ser. Proceedings of Machine Learning Research, vol. 70, PMLR, 2017, pp. 3319–3328. [Online]. Available: https://proceedings.mlr.press/v70/sundararajan17a.html.

[34] A. Paszke, S. Gross, F. Massa, *et al.*, "Pytorch: An imperative style, high-performance deep learning library", pp. 8024–8035, 2019. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[35] T. Dao, D. Y. Fu, S. Ermon, A. Rudra, and C. R'e, "Flashattention: Fast and memory-efficient exact attention with io-awareness", *ArXiv*, vol. abs/2205.14135, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:249151871.

Appendix A

Appendix

A.1 List of .root Files Used in V8

$t\bar{t}H$

mc16a/p4498/346343, mc16a/p4498/346344, mc16a/p4498/346345,
mc16d/p4498/346343, mc16d/p4498/346344, mc16d/p4498/346345,
mc16e/p4498/346343, mc16e/p4498/346344, mc16e/p4498/346345.

$t\bar{t}W$

mc16a/p4416/700168, mc16d/p4416/700168, mc16e/p4416/700168.

$t\bar{t}W_{EW}$

mc16a/p4590/700205, mc16d/p4590/700205, mc16e/p4590/700205.

$t\bar{t}Z$

mc16a/p4416/504330, mc16a/p4416/504334, mc16a/p4416/504342,
mc16d/p4416/504330, mc16d/p4416/504334, mc16d/p4416/504342,
mc16e/p4416/504330, mc16e/p4416/504334, mc16e/p4416/504342.

$t\bar{t}$

mc16a/p4308/410470, mc16d/p4308/410470, mc16e/p4308/410470.

VV

mc16a/p4416/364250, mc16a/p4416/364253, mc16a/p4416/364254,
mc16a/p4416/364255, mc16a/p4308/364283, mc16a/p4308/364284,
mc16a/p4308/364285, mc16a/p4308/364286, mc16a/p4308/364287,
mc16a/p4308/363355, mc16a/p4308/363356, mc16a/p4308/363357,
mc16a/p4308/363358, mc16a/p4308/363359, mc16a/p4308/363360,
mc16a/p4308/363489, mc16d/p4416/364250, mc16d/p4416/364253,
mc16d/p4416/364254, mc16d/p4416/364255, mc16d/p4308/364283,
mc16d/p4308/364284, mc16d/p4308/364285, mc16d/p4308/364286,

mc16d/p4308/364287, mc16d/p4308/363355, mc16d/p4308/363356,
mc16d/p4308/363357, mc16d/p4308/363358, mc16d/p4308/363359,
mc16d/p4308/363360, mc16d/p4308/363489, mc16e/p4416/364250,
mc16e/p4416/364253, mc16e/p4416/364254, mc16e/p4416/364255,
mc16e/p4308/364283, mc16e/p4308/364284, mc16e/p4308/364285,
mc16e/p4308/364286, mc16e/p4308/364287, mc16e/p4308/363355,
mc16e/p4308/363356, mc16e/p4308/363357, mc16e/p4308/363358,
mc16e/p4308/363359, mc16e/p4308/363360, mc16e/p4308/363489.

tZ

mc16a/p4308/410560, mc16d/p4308/410560, mc16e/p4308/410560.

WtZ

mc16a/p4308/410408, mc16d/p4308/410408, mc16e/p4308/410408.

tW

mc16a/p4308/410646, mc16a/p4308/410647, mc16d/p4308/410646,
mc16d/p4308/410647, mc16e/p4308/410646, mc16e/p4308/410647.

$t\bar{t}t$ **(three top)**

mc16a/p4308/304014, mc16d/p4308/304014, mc16e/p4308/304014.

$t\bar{t}t\bar{t}$ **(fourTop)**

mc16a/p4308/410080, mc16d/p4308/410080, mc16e/p4308/410080.

$ggVV$

mc16a/p4308/345705, mc16a/p4396/345706, mc16a/p4396/345715,
mc16a/p4396/345718, mc16a/p4396/345723, mc16d/p4308/345705,
mc16d/p4396/345706, mc16d/p4396/345715, mc16d/p4396/345718,
mc16d/p4396/345723, mc16e/p4308/345705, mc16e/p4396/345706,
mc16e/p4396/345715, mc16e/p4396/345718, mc16e/p4396/345723.

VVV

mc16a/p4308/364242, mc16a/p4308/364243, mc16a/p4308/364244,
mc16a/p4308/364245, mc16a/p4308/364246, mc16a/p4308/364247,
mc16a/p4308/364248, mc16a/p4308/364249, mc16d/p4308/364242,
mc16d/p4308/364243, mc16d/p4308/364244, mc16d/p4308/364245,
mc16d/p4308/364246, mc16d/p4308/364247, mc16d/p4308/364248,
mc16d/p4308/364249, mc16e/p4308/364242, mc16e/p4308/364243,
mc16e/p4308/364244, mc16e/p4308/364245, mc16e/p4308/364246,
mc16e/p4308/364247, mc16e/p4308/364248, mc16e/p4308/364249.

VH

mc16a/p4308/342284, mc16a/p4308/342285, mc16d/p4308/342284, mc16d/p4308/342285, mc16e/p4308/342284, mc16e/p4308/342285.

$WttW$

mc16a/p4308/410081, mc16d/p4308/410081, mc16e/p4308/410081.

$tHjb$

mc16a/p4308/346799_AF, mc16d/p4308/346799_AF, mc16e/p4308/346799_AF.

tWH

mc16a/p4308/346678_AF, mc16d/p4308/346678_AF, mc16e/p4308/346678_AF.

A.2 SR Cut Expression

```
custTrigMatch_LooseID_FCLooseIso_DLT
&& (dilep_type && (lep_ID_0*lep_ID_1)>0)
&& ((lep_Pt_0 >= 10e3 && lep_Pt_1 >= 10e3) && (fabs(lep_Eta_0) <= 2.5 && fabs(lep_Eta_1) <= 2.5)
    && ((abs(lep_ID_0) == 13 && lep_isMedium_0 && lep_isolationLoose_VarRad_0 && passPLIVTight_0)
        || ((abs(lep_ID_0) == 11 && lep_isTightLH_0 && lep_isolationLoose_VarRad_0 && passPLIVTight_0
            && lep_ambiguityType_0 == 0 && lep_chargeIDBDTResult_recalc_rel207_tight_0 > 0.7)
            && ((!(!(lep_Mtrktrk_atConvV_CO_0 < 0.1 && lep_Mtrktrk_atConvV_CO_0 >= 0 && lep_RadiusCO_0 > 20)
                && (lep_Mtrktrk_atPV_CO_0 < 0.1 && lep_Mtrktrk_atPV_CO_0 >= 0)))
                && !(lep_Mtrktrk_atConvV_CO_0 <0.1 && lep_Mtrktrk_atConvV_CO_0 >= 0 && lep_RadiusCO_0 > 20))))
        && ((abs(lep_ID_1) == 13 && lep_isMedium_1 && lep_isolationLoose_VarRad_1 && passPLIVTight_1)
            || ((abs(lep_ID_1) == 11 && lep_isTightLH_1 && lep_isolationLoose_VarRad_1 && passPLIVTight_1
                && lep_ambiguityType_1 == 0 && lep_chargeIDBDTResult_recalc_rel207_tight_1 > 0.7)
                && ((!(!(lep_Mtrktrk_atConvV_CO_1 < 0.1 && lep_Mtrktrk_atConvV_CO_1 >= 0 && lep_RadiusCO_1 > 20)
                    && (lep_Mtrktrk_atPV_CO_1 < 0.1 && lep_Mtrktrk_atPV_CO_1 >= 0)))
                    && !(lep_Mtrktrk_atConvV_CO_1 < 0.1 && lep_Mtrktrk_atConvV_CO_1 >= 0 && lep_RadiusCO_1 > 20)))))
&& nTaus_OR==1
&& nJets_OR_DL1r_85>=1
&& nJets_OR>=4
&& ((dilep_type==2) || abs(Mll01-91.2e3)>10e3)
```

We have kept the cuts the same as [5], except for the cut on the `nJets_OR` to `>=4` to keep consistent definition SR definition across the $t\bar{t}H$ $2l_{SS} + 1\tau_{\text{had}}$ analysis.

A.3 Event Weighting

$$\begin{aligned}\text{Channels} = \{&364250, 364253, 364254, 364255, 364283, 364284, 364285,\\ &364286, 364287, 363355, 363356, 363357, 363358, 363359,\\ &363360, 363489, 345705, 345706, 345715, 345718, 345723\}\end{aligned} \tag{A.1}$$

$$\text{Weight}_{\text{nJets}} = \begin{cases} 1.0, & \text{if } \texttt{nJets\_OR} = 0 \\ 0.986980, & \text{if } \texttt{nJets\_OR} = 1 \\ 0.853062, & \text{if } \texttt{nJets\_OR} = 2 \\ 0.785437, & \text{if } \texttt{nJets\_OR} = 3 \\ 0.741692, & \text{if } \texttt{nJets\_OR} = 4 \\ 0.709992, & \text{if } \texttt{nJets\_OR} = 5 \\ 0.685452, & \text{if } \texttt{nJets\_OR} = 6 \\ 0.665613, & \text{if } \texttt{nJets\_OR} \geq 7 \end{cases} \tag{A.2}$$

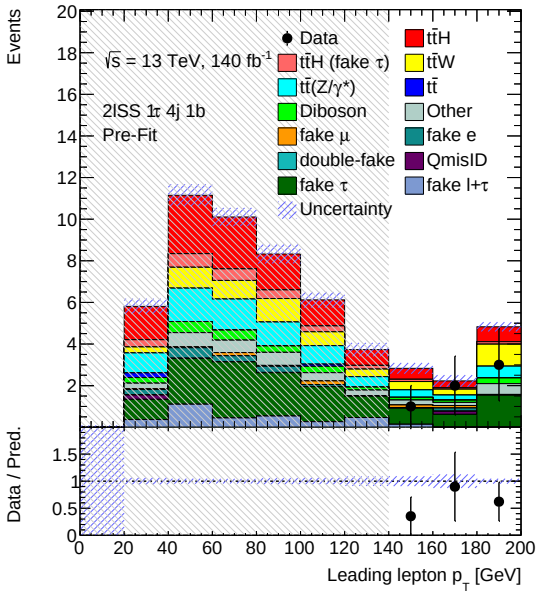
$$\texttt{XXX\_VV\_NJET} = \begin{cases} \text{Weight}_{\text{nJets}}, & \text{if } \texttt{mcChannelNumber} \in \text{Channels} \\ 1.0, & \text{otherwise} \end{cases} \tag{A.3}$$

$$\text{YearLuminosity} = \begin{cases} 36646.74, & \text{if } \texttt{RunYear} = 2015 \text{ or } \texttt{RunYear} = 2016 \\ 44630.6, & \text{if } \texttt{RunYear} = 2017 \\ 58791.6, & \text{if } \texttt{RunYear} = 2018 \end{cases} \tag{A.4}$$

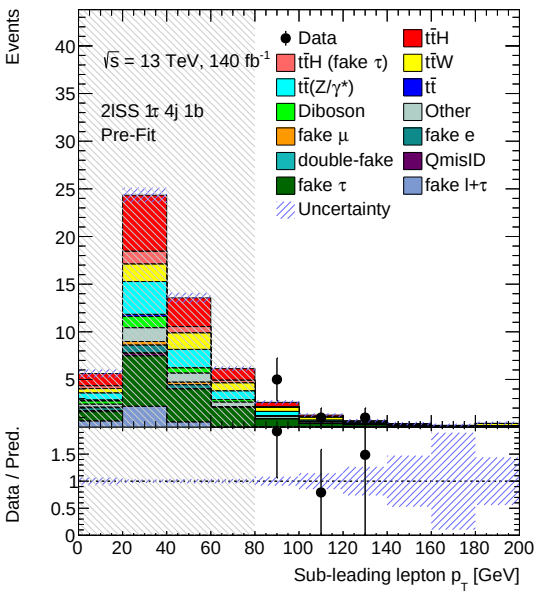
$$\begin{aligned}\text{Event weight} = w = \ &\texttt{YearLuminosity}\\ &\times \frac{\sigma}{\mathcal{L}}\\ &\times \texttt{custTrigSF\_LooseID\_FCLooseIso\_SLTorDLT}\\ &\times \texttt{weight\_pileup}\\ &\times \texttt{jvtSF\_customOR}\\ &\times \texttt{bTagSF\_weight\_DL1r\_85}\\ &\times \texttt{XXX\_VV\_NJET}\\ &\times \texttt{weight\_mc}\\ &\times \texttt{lep\_SF\_CombinedTight\_0}\\ &\times \texttt{lep\_SF\_CombinedTight\_1}\\ &\times \texttt{lepSF\_PLIV\_Prompt\_0}\\ &\times \texttt{lepSF\_PLIV\_Prompt\_1}\\ &\times \frac{1}{\sum_{w\sim\text{MC}} w},\end{aligned} \tag{A.5}$$

where $\mathcal{L}$ = Luminosity = 140068.94, σ is the cross-section (`xs`), and $\sum_{w\sim\text{MC}} w$ is the sum of the weights of all simulated events (`totalEventsWeighted`).

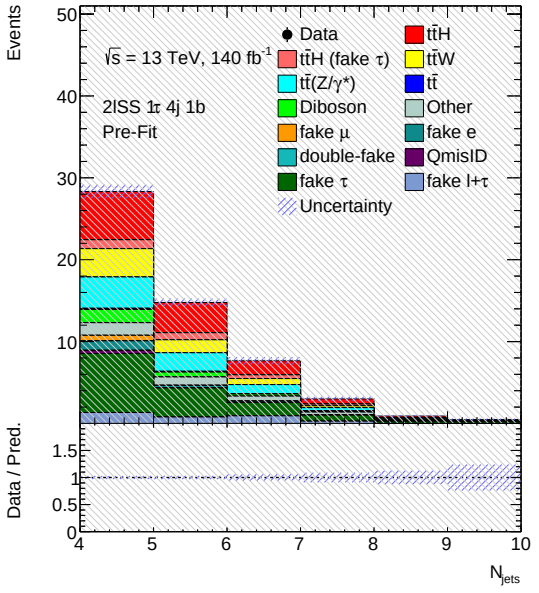
A.4 Yields Plots



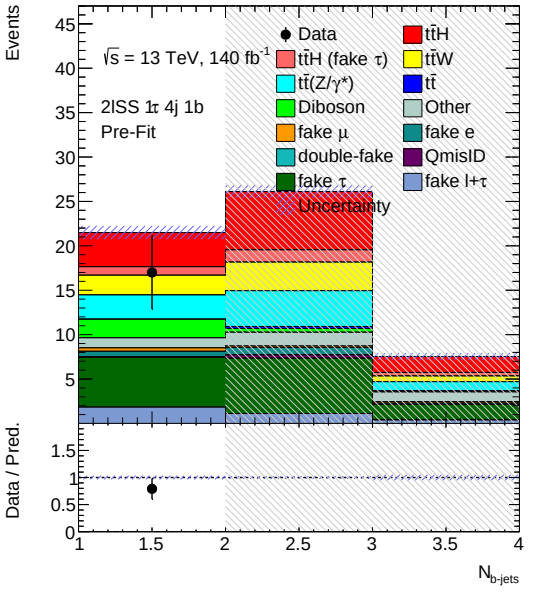
(a) Distribution of the transverse momentum of the leading lepton.



(b) Distribution of the transverse momentum of the subleading lepton.



(c) Distribution of the number of jets.



(d) Distribution of the number of b-jets.

Figure A.1: Distributions of different variables inside the SR. The blinding threshold of 0.3 is applied.

A.5 Why Supervised Learning?

In the context of classifying specific particle interactions like $t\bar{t}H$ events, supervised learning is often favored for several reasons:

1. Labeling Ground Truth: With a well-established theoretical model and simulation techniques, you can generate labeled datasets that represent the intermediate and final states of particle collisions. This allows for direct training on what you want the model to learn.

2. Efficiency and Precision: Supervised learning can leverage this labeled data to create precise and targeted models for the classification problem at hand. This generally results in a more efficient learning process compared to unsupervised methods like clustering, which may not have specific labels to guide the learning.

3. Model Interpretability: By aligning the model with labeled examples, it's often easier to interpret the model's decision process and understand how it correlates the inputs to the desired classification. This can be important in scientific contexts where understanding the model's behavior can be as crucial as its predictive accuracy. In the case of our task, specifically, a particular use-case would be to determine which parts of detectors deserve more attention during improvement, and which one may not be so relevant.

4. Directly Aligned with the Objective: If the main goal is to classify specific types of events, then using supervised learning directly aligns with this objective. It utilizes the available knowledge of the phenomena being studied (in this case, particle physics and the standard model) to create a learning paradigm specifically designed to recognize those events.

5. Potential for Fine-Tuning: Supervised models can often be fine-tuned or adjusted with new data or different structures to continually improve or adapt to new insights. This iterative refinement aligns well with the scientific method of incremental understanding and improvement.

6. Control Over Error and Bias: In the supervised learning framework, you can often have more control over the types of errors and biases that the model may introduce, as you're explicitly defining what constitutes a correct classification.

7. Challenges with Unsupervised Learning: On the contrary, unsupervised learning methods like clustering would require determining similarities between events without clear labels, potentially leading to ambiguous or less accurate classifications. It might not leverage the rich theoretical knowledge available in the field of particle physics.

8. In conclusion, while unsupervised methods might be useful in exploratory phases or when labeled data is not available, the specific nature of $t\bar{t}H$ event classification, combined with the availability of simulated labeled data and theoretical grounding, makes supervised learning a well-suited and likely more effective approach.

Clustering, as an unsupervised learning technique, can be applied to the context of our task, but with some significant caveats. Here is how it might work and the challenges involved:

1. Application: Clustering could be used to group collision events based on similarities in their observable features, without prior knowledge of labels. By identifying patterns in the data, you might uncover groups that correspond to different types of events, such as $t\bar{t}H$ and others.

2. Prediction: Once the clusters have been identified, you could theoretically assign labels to those clusters based on expert knowledge or additional analysis. These labels could then be used to classify new data. However, this is where the challenges and limitations become apparent.

3. Challenges and Limitations: Lack of Ground Truth: Without labeled data to guide the clustering, there's no straightforward way to ensure that the clusters align with the actual underlying physics. The clusters might correspond to some other aspects of the data that aren't relevant to the classification task.

4. Ambiguity: The boundaries between clusters may be ambiguous, leading to uncertainty in how to label the clusters. This could result in less accurate classifications.

5. Model Complexity: Interpreting and validating the clusters might require significant expertise and additional modeling, making it a more complex and potentially less reliable approach than supervised learning.

6. Predictive Power: Without a clear mapping from features to labels (as in supervised learning), the predictive power of a clustering-based model might be lower, especially if the clusters do not correspond well to the underlying physics.

7. Not Directly Aligned with the Task: Clustering is generally used for exploratory data analysis and pattern recognition rather than direct classification. Adapting it to a classification task like identifying $t\bar{t}H$ events may require significant modification and may not be as efficient or effective as using a method specifically designed for classification, such as supervised learning.

A.5.1 Features used

`lep_E_X`

The energy of the Xth lepton.

`DRjj_lead`

ΔR between the two leading jets. ΔR is a distance metric in the $\eta - \phi$ space frequently used in particle physics.

`Ptll01`

The transverse momentum of the dilepton system made up of the two leading leptons.

`lep_nTrackParticles_X`

The number of tracks associated with the Xth lepton.

`custTrigMatch_LooseID_FCLooseIso_DLT`

Custom trigger matching for loosely identified and loosely isolated leptons.

`Mll01`

The invariant mass of the two leading leptons.

`Mlll012`

The invariant mass of the three leading leptons.

`total_charge`

The sum of the electric charges of the particles in the event.

`HT`

The scalar sum of the transverse momenta of all jets in the event.

`HT_lep`

The scalar sum of the transverse momenta of all leptons in the event.

`HT_jets`

The scalar sum of the transverse momenta of all jets (not forward jets) in the event.

`HT_fwdJets`

The scalar sum of the transverse momenta of all forward jets in the event.

`HT_taus`

The scalar sum of the transverse momenta (P_t) of all tau leptons in the event.

`lep_Eta_X`

The pseudorapidity of the Xth lepton.

`nTaus_OR_Pt25`

The number of overlapping-removed taus with a transverse momentum above 25 GeV.

`nFwdJets_OR`

The number of overlapping-removed forward jets.

`MLepMet`

The invariant mass of a lepton and the missing transverse energy vector.

`taus_DL1r_X`

The DL1r score for the Xth tau.

`lep_isolationLoose_VarRad_X`

Indicates whether a lepton (where X refers to the lepton index) passes an isolation cut with a variable radius. Looser isolation cuts allow more nearby activity in the detector.

`lep_EtaBE2_X`

The pseudorapidity of the Xth lepton in the second layer of the electromagnetic calorimeter.

`taus_width_X`

The width of the Xth tau.

`nJets_OR_DL1r_85`

Number of jets that pass overlap removal (OR) and are b-tagged according to the DL1r algorithm at the 85% working point.Overlap removal is a step in particle reconstruction where, for instance, an object identified as both a jet and a tau would be considered only as one or the other.

`lep_nInnerPix_X`

Number of hits in the inner pixel detector associated with the lepton, where X refers to the lepton index.

`met_phi`

The azimuthal angle of the missing transverse energy in the event.

`DeltaR_max_lep_bjet77`

The maximum ΔR value between a lepton and a b-tagged jet.

`MbX`

Invariant mass associated with the leading b-jet in the event

`lep_RadiusCO_X`

Radius of the cone used for isolation of the lepton, or alternatively a parameter associated with the trajectory of the lepton.

`lep_Mtrktrk_atConvV_CO_X`

The invariant mass of track pairs at the conversion vertex for lepton X.

`lep_Pt_X`

The transverse momentum of the Xth lepton.

`mjjMax_frwdJet`

The maximum invariant mass of a pair of forward jets.

`dilep_type`

The type of dilepton event (e.g., ee, μe, $\mu\mu$).

`eta_frwdjet`

The pseudorapidity of the forward jet.

`Mlb`

Invariant mass of a lepton and a b-jet.

`minDeltaR_LJ_X`

The minimum ΔR distance between the Xth lepton and any jet in the event.

`nTaus_OR`

Number of tau leptons that pass overlap removal.

`DeltaR_min_lep_jet`

The minimum ΔR distance between a lepton and a jet in the event.

`lep_sigd0PV_X`

Significance of the transverse impact parameter (d0) of the lepton X with respect to the primary vertex (PV). This is a common variable for distinguishing prompt particles produced in the primary collision from secondary particles produced in a decay.

`taus_eta_X`

The pseudorapidity of the Xth tau.

`lep_Phi_X`

The azimuthal angle (in radians) of the Xth lepton.

`lep_chargeIDBDTResult_recalc_rel207_tight_X`

The outcome of a BDT-based charge identification for a lepton, recalculated with some specific settings, and applying a 'tight' threshold.

`taus_phi_X`

The azimuthal angle (in radians) of the Xth tau.

`taus_passJVT_X`

A boolean flag indicating whether the Xth tau passes the jet vertex tightness (JVT) requirement.

`jets_eta`

The pseudorapidity of the jets (array).

`taus_charge_X`

The charge of the Xth tau.

`passPLIVTight_X`

Boolean flag indicating if a lepton with high transverse momentum passes the "tight" criteria of the Prompt Lepton Veto (PLIV), a tool for identifying non-prompt light leptons.

`lep_Mtrktrk_atPV_CO_X`

The invariant mass of track pairs at the primary vertex for lepton X.

`taus_JetRNNSigMedium_X`

RNN-based score for tau lepton, used to distinguish tau leptons from jets, with 'medium' selection criteria.

`minOSMll`

The minimum invariant mass of oppositely-signed dilepton pairs.

`lep_ID_X`

The identification number for the Xth lepton.

`Mllll0123`

The invariant mass of the four leading leptons.

`custTrigSF_TightElMediumMuID_FCLooseIso_DLT`

Custom trigger scale factor, for events with a tight electron and a medium muon, both of which are loosely isolated.

`best_Z_Mll`

The invariant mass of the dilepton system that is closest to the Z boson mass.

`met_met`

The missing transverse energy in the event.

`MtLep1Met`

Transverse mass between the leading lepton and missing transverse energy. Transverse mass is often used in searches for particles that decay to a lepton and a neutrino.

`lep_ambiguityType_X`

Type of ambiguity for lepton identification, where X refers to the lepton index. Ambiguity could arise from several factors, such as a single track matching with multiple reconstructed particles.

`jets_phi`

The azimuthal angle (in radians) of the jets (array).

`lep_isMedium_X`

Boolean flag indicating if a lepton passes the 'medium' selection criteria.

`taus_RNNJetScore_X`

RNN-based score for tau lepton, used to distinguish tau leptons from jets.

`MtLepMet`

The transverse mass of a lepton and the missing transverse energy vector.

`DeltaR_min_lep_jet_fwd`

The minimum ΔR distance between a lepton and a forward jet in the event.

`jets_e`

The energy of the jets (array).

`minOSSFMll`

The minimum invariant mass of oppositely-signed, same-flavor dilepton pairs.

`nJets_OR`

The number of overlapping-removed jets.

`total_leptons`

The total number of leptons in the event.

`taus_numTrack_X`

The number of tracks associated with the Xth tau.

`taus_passEleOLR_X`

A boolean flag indicating whether the Xth tau passes the electron overlap removal.

`DRll01`

The ΔR distance between the two leading leptons.

`taus_JetRNNSigLoose_X`

RNN-based score for tau lepton, used to distinguish tau leptons from jets, with 'loose' selection criteria.

`taus_pt_X`

The transverse momentum of the Xth tau.

`flag_JetCleaning_LooseBad`

A flag variable indicating whether a jet passes a loose cleaning cut to remove bad or noisy jets from the analysis.

`taus_fromPV_X`

A boolean flag indicating whether the Xth tau comes from the primary vertex.

`best_Z_other_MtLepMet`

The transverse mass between the lepton and missing transverse energy for the event that best reconstructs a Z boson using other criteria.

`nJets_OR_DL1r_77`

Count of jets that pass overlap removal (OR) and are b-tagged according to the DL1r algorithm at the 77% working point.

`jets_pt`

The transverse momentum of the jets (array).

`lep_isTightLH_X`

Boolean flag indicating if a lepton passes the 'tight' Likelihood-based identification criteria.

`taus_JetRNNSigTight_X`

RNN-based score for tau lepton, used to distinguish tau leptons from jets, with 'tight' selection criteria.

`sumPsbtag`

The sum of b-tagging weights for jets in the event.

`taus_decayMode_X`

The decay mode of the Xth tau.

`dEta_maxMjj_frwdjet`

The maximum difference in pseudorapidity (η) between two forward jets.

`max_eta`

The maximum pseudorapidity among all particles in the event.

`best_Z_other_Mll`

The invariant mass of the dilepton system that is closest to the Z boson mass, not considering the leading leptons.

`taus_passEleBDT_X`

Flag indicating if a tau lepton passes the Electron Boosted Decision Tree discriminator.

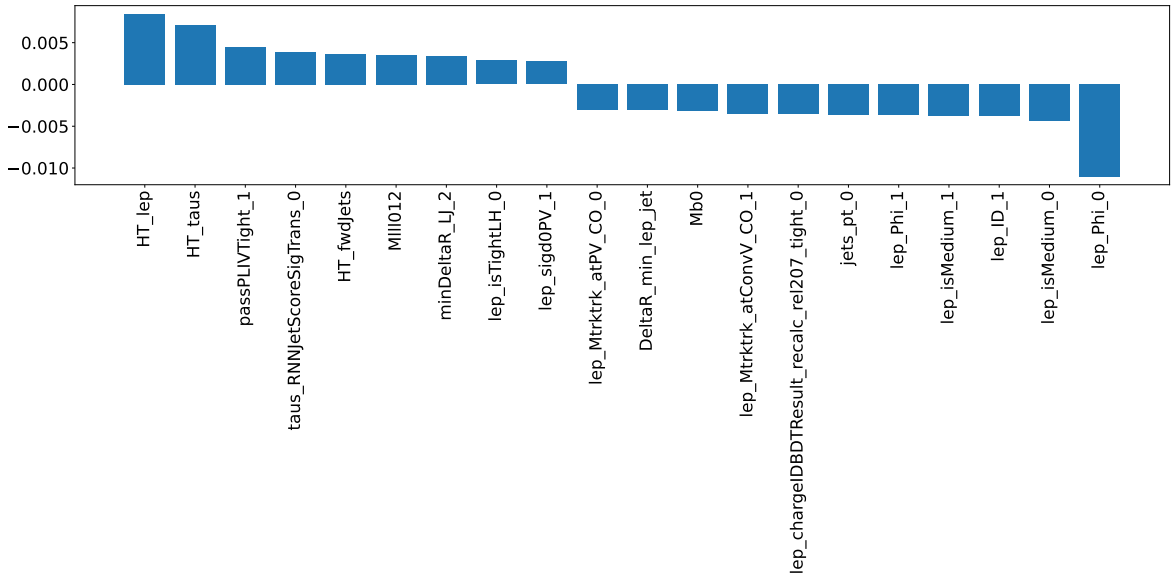


Figure A.2: Feature importance of the top 20 most important features. The feature importance was calculated using the Integrated Gradients (IG) method [33].

A.6 Friend Trees

To be able to perform the analysis of the impact of the systematic uncertainties in `TRExFitter` using the custom classifier's output variable, one must either inject the variable in all the root files, or create the so-called friend n-tuples, where each file only contains the classifier's output variable. The latter is the preferred option, as it does not require modifying or copy the original files (which are ≈ 2.7 TB in size).

First, one must essentially process all the events with the classifier. However, to make the processing shorter, the cuts can be applied to all the systematic files first, which reduces the total size to $\approx$ 40 GB. Then, the classifier should be applied to all the events to produce the output variable. For each input file, a new "friend" file is produced, which contains only the classifier's output variable.

Once the friend trees are produced, `TRExFitter` should be made aware of them.

To do so, `UseFriend: TRUE` should be added across all the `Sample` blocks. Then, in the `Job` block definition, `FriendPath: XXX_FriendPaths` should be added, with the `XXX_FriendPaths` pointing to the `nominal` directory of the trees.

Furthermore, for the systematic blocks definitions, where the `NtuplePathsUp` or `NtuplePathsDown` are defined, the `FriendPathsUp` and `FriendPathsDown` should be added, respectively.

This enables `TRExFitter` to access both the original trees containing all the systematics-related information, and the friend trees containing only the classifier's output variable.

A.7 Training Optimizations

Here we list some additional details that do not influence the quality of the optimization itself.

As we have used large NNs and a large training set, the training times are quite long. Without some necessary optimization, experiments take extremely long time to complete. We have used the following techniques to speed up the training:

A.7.1 Mixed Precision Training

Mixed precision training is a technique in deep learning that leverages the benefits of both low-precision and high-precision numerical representations to accelerate model training and improve overall efficiency. It involves using a combination of reduced-precision (such as 16-bit) and full-precision (such as 32-bit) floating-point computations during the training process. By employing reduced precision for certain computations, such as matrix multiplications, mixed precision training can significantly speed up the training process while maintaining a comparable level of accuracy. This approach is especially useful when training large-scale models with massive amounts of data, as it reduces memory usage, allows for faster computations, and enables the use of larger batch sizes. Overall, mixed precision training is a valuable technique that helps us achieve faster and more efficient deep learning models, leading to quicker iteration cycles and advancements in various fields, including computer vision, natural language processing, and reinforcement learning.

A.7.2 PyTorch 2.0 and torch.compile()

`torch.compile()` is a feature introduced in PyTorch 2.0 [34] that aims to improve the performance of PyTorch code by JIT-compiling it into optimized kernels. It allows PyTorch code to run faster while requiring minimal code changes. The `torch.compile()` supports arbitrary PyTorch code, control flow, and mutation, and comes with experimental support for dynamic shapes. By using `torch.compile()`, developers can optimize their PyTorch code without sacrificing flexibility or ease of use. This feature is particularly useful for boosting the performance of PyTorch models during training and inference.

A.7.3 FlashAttention

FlashAttention [35] is a fast and memory-efficient exact attention algorithm that aims to improve the training speed and quality of models with long sequences in machine learning applications. It incorporates IO-awareness, which involves dividing operations between faster and slower levels of GPU memory to optimize performance. By reordering the attention computation and leveraging classical techniques such as tiling and recomputation, FlashAttention significantly speeds up the attention process and reduces memory usage from quadratic to linear in sequence length. This algorithm outperforms other exact attention algorithms in terms of training speed and model quality, especially when dealing with long sequences. It achieves faster end-to-end training time and higher quality models by accounting for GPU memory reads and writes, resulting in improved performance and reduced compute complexity. FlashAttention is a valuable tool for researchers and

practitioners working with attention mechanisms in machine learning, enabling them to train models more efficiently and effectively.