

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



**Analysis of the  $t\bar{t}H$  Process in Same-Sign Dilepton  
Events with Hadronic Tau Decay Using  
Self-Attention Architectures**

Bachelor's Thesis

*David William Erwin*

Bachelor's program: Electrical Engineering and Computer Science  
Supervisor: doc. Dr. André Sopczak

Prague, May 2025

**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

# Declaration

I hereby declare I have written this bachelor's thesis independently and quoted all the sources of information used in accordance with methodological instructions on ethical principles for writing an academic thesis.

In Prague, May 2025

.....  
David William Erwin

# Abstract

The associated production of a Higgs boson with a top-antitop quark pair ( $t\bar{t}H$ ) is a rare but important process studied using the ATLAS detector at CERN. Separating this signal from the vast sea of background data is an impossible task for humans, and a challenging task for machines. In this work, we apply attention-based neural networks, also called transformers, to classify simulated events in the same-sign dilepton plus hadronic  $\tau$  channel. We start with a simple transformer model and progress to implementing state-of-the-art techniques and concepts. Throughout the process, we analyze the inner mechanisms of the models in detail, including activation histograms and detailed head-specific attention maps. We achieve a final ROC AUC score of 0.804 and an  $F_1$ -score of 0.704. Using the CERN-developed TRExFitter, we produce normalized model probability distributions to illustrate the separation between signal and background predictions and assess the stability of our classifier across different background processes. We then quantify the intrinsic statistical uncertainty in the expected  $t\bar{t}H$  coupling measurement and find that our model reduces the  $\pm 1\sigma$  uncertainty from  $\mu = 1 \pm 0.28$  to  $\mu = 1 \pm 0.24$ .

**Keywords:** CERN, ATLAS, Particle Physics, Higgs Boson, Machine Learning

# Acknowledgements

First, my heartfelt thanks go to my supervisor, doc. Dr. André Sopczak, whose expert advice, generosity with his time, and eager encouragement made this thesis possible. I especially appreciate his push for me to speak on my topic at DPG-Frühjahrstagungen, Göttingen, which marked my first-ever conference presentation and was a transformative experience.

I would also like to thank the kind people of the CERN community for their generosity and support, from taking the time to answer a fellow student's questions to providing invaluable resources such as the LXPLUS machine. My experience being part of this esteemed organization has been nothing short of life-changing, and I am incredibly grateful to have contributed, in my small way, to the exciting field of particle physics.

I am especially grateful for my partner, Alenka. From late-night writing sessions to skipped meals and workouts, her unfaltering kindness was a constant source of support during this journey, and I will always be grateful to both her and her family for their generosity.

Last, but certainly not least, I would like to express my eternal gratitude to my family in Florida. Thank you for always being there for me, for supporting me through every difficult decision or moment, for having the confidence in me that you have, and for always being excited to hear how my day went. Thank you, a million times over, for everything.

# List of Figures

1.1	Standard Model of Particle Physics. . . . .	2
1.2	Higgs boson coupling to other elementary particles [1]. . . . .	4
1.3	Illustration of the transformer model introduced by the <i>Attention is All You Need</i> paper. . . . .	7
1.4	Activation functions commonly found in ML applications. . . . .	12
1.5	Representative Feynman diagrams for the four main Higgs boson production mechanisms at the LHC: (a) gluon fusion ( $gg \rightarrow H$ ), (b) vector-boson fusion (VBF), (c) associated $VH$ production ( $q\bar{q}' \rightarrow VH$ ), and (d) $t\bar{t}H$ production ( $gg \rightarrow t\bar{t}H$ ) [17]. . . . .	16
2.1	CERN accelerator complex. . . . .	20
2.2	ATLAS detector under construction. . . . .	22
3.1	Feynman diagram of gluon-gluon fusion creating the $t\bar{t}H$ signal and subsequent decay into two light leptons and a hadronically decaying tau. . . . .	25
5.1	Prenormalized <code>dEta_maxMjj_frwdjet_NOSYS</code> . . . . .	35
5.2	<code>dEta_maxMjj_frwdjet_NOSYS</code> , normalized via Z-scoring. . . . .	35
5.3	Prenormalized <code>taus_pt_0_NOSYS</code> (in MeV). . . . .	36
5.4	<code>taus_pt_0_NOSYS</code> , normalized via Z-scoring. . . . .	37
5.5	<code>taus_pt_0_NOSYS</code> , normalized via log + Z-scoring. . . . .	37
5.6	Correlation heatmap of the pre-normalized feature set, with the “_NOSYS” tag removed from each feature for readability. . . . .	39
5.7	Correlation heatmap of the feature set, post feature-wise scaling, with the “_NOSYS” tag removed from each feature for readability. . . . .	40
5.8	ROC curve of the baseline model trained on the feature-wise scaled dataset. . . . .	44
5.9	Training and validation loss over 50 epochs, baseline model. . . . .	46
5.10	Activation values for the input projection layer after 50 epochs, baseline model. . . . .	47
5.11	Activation values for the last encoder block after 50 epochs, baseline model. . . . .	48
5.12	Average attention heatmap of the last encoder block after 50 epochs, baseline model, feature-wise scaled dataset. . . . .	49
5.13	Attribution map for the signal class — signal event scenario, baseline model. . . . .	50
5.14	Attribution map for the background class — background event scenario, baseline model. . . . .	50
5.15	Attribution map for the signal class — background event scenario, baseline model. . . . .	51
5.16	Attribution map for the background class — signal event scenario, baseline model. . . . .	51

5.17	Distribution of the posterior probability $P(y = t\bar{t}H \mid x)$ as estimated by the TRExFitter, baseline model trained on the feature-wise scaled dataset.	52
5.18	Average attention heatmap of the last encoder block after 50 epochs, baseline model, z-score standardized dataset. . . . .	54
6.1	Diagram of the proposed model, where $N = L \times d_{model}$ . . . . .	56
6.2	Activation values for the input projection layer after 40 epochs, proposed model. . . . .	61
6.3	Activation values for the last encoder block after 40 epochs, proposed model.	62
6.4	Attention heatmap of head 0 in the last encoder block after 40 epochs, proposed model, feature-wise scaled dataset. . . . .	63
6.5	Average attention heatmap of the last encoder block after 40 epochs, proposed model, feature-wise scaled dataset. . . . .	64
6.6	ROC curve of the proposed model trained on the feature-wise scaled dataset.	65
7.1	Normalization factor for the single-bin TRExFitter setup: $\mu_1 = 1 \pm 0.28$ . .	68
7.2	Normalization factor for the ten-bin TRExFitter setup with our proposed model, trained on the z-score standardized dataset, applied: $\mu_{10} = 1 \pm 0.24$	69

# List of Tables

1.1	Partial widths and branching ratios for the main Higgs-boson decay channels at $m_H \simeq 125$ GeV [20]. . . . .	18
3.1	Primary $\tau$ lepton decay channels at $m_\tau \simeq 1.777$ GeV, with branching ratios and an indication of hadronic modes [26]. . . . .	24
3.2	Parton-level partial widths and branching ratios for the primary hadronic $\tau^-$ decays into quark-antiquark pairs plus a neutrino [26]. . . . .	24
5.1	Input features used in the analysis. . . . .	33
5.2	Normalization method applied to each input feature in the z-score standardized dataset. . . . .	36
5.3	Normalization method applied to each input feature in the feature-wise scaled dataset. . . . .	38
5.4	Classification report for background versus signal, baseline model trained on the feature-wise scaled dataset. . . . .	45
6.1	Hyperparameters optimized by Optuna for the proposed model using both the feature-wise scaled dataset and the z-score standardized dataset. . . . .	60
6.2	Classification report for background versus signal, proposed model trained on the feature-wise scaled dataset. . . . .	66
6.3	Classification report for background versus signal, proposed model trained on the z-score standardized dataset. . . . .	67
A.1	Full list of all hyperparameters for the proposed model using both the feature-wise optimally scaled dataset and the z-score standardized dataset. . . . .	75
A.2	Project dependencies. . . . .	79



# Acronyms & Glossary

**API** Application Programming Interface: A connection between computers or between computer programs. It is a type of software interface, offering a service to other pieces of software. A document or standard that describes how to build such a connection or interface is called an API specification. A computer system that meets this standard is said to implement or expose an API. The term API may refer either to the specification or to the implementation. 30

**CERN** Conseil Européen pour la Recherche Nucléaire (European Council for Nuclear Research): CERN is an intergovernmental organization that operates the largest particle physics laboratory in the world. Established in 1954, it is based in Meyrin, western suburb of Geneva, on the France–Switzerland border. It comprises 24 member states. CERN is an official United Nations General Assembly observer. v, 20, 29–31

**DNN** Deep Neural Network: An artificial neural network with multiple layers between the input and output layers. There are different types of neural networks but they always consist of the same components: neurons, synapses, weights, biases, and functions. These components as a whole function in a way that mimics functions of the human brain, and can be trained like any other machine learning algorithm. 5, 6, 13

**HEP** High-Energy Physics: Also known as Particle Physics, is the study of fundamental particles and forces that constitute matter and radiation. The field also studies combinations of elementary particles up to the scale of protons and neutrons, while the study of combinations of protons and neutrons is called nuclear physics. xiii, 6, 18–20, 23, 27, 30, 31, 44, 70

**LHC** Large Hadron Collider: The LHC is the world’s largest and highest-energy particle accelerator. It was built by the European Organization for Nuclear Research (CERN) between 1998 and 2008, in collaboration with over 10,000 scientists, and hundreds of universities and laboratories across more than 100 countries. It lies in a tunnel 27 kilometres (17 mi) in circumference and as deep as 175 metres (574 ft) beneath the France–Switzerland border near Geneva. vi, 16, 17, 20, 21

**LXPLUS** Linux Public Login User Service: An interactive Linux-based computing cluster provided by CERN’s IT Department for all CERN users. It serves as a general-purpose platform for interactive work, offering large data storage and access to both CPU and GPU nodes. v, 29, 31

- MCS** Monte Carlo Simulation: A computational technique that uses repeated random sampling to model and analyze complex systems or processes. It generates multiple scenarios by assigning random values to uncertain variables, allowing for the estimation of outcomes and their probabilities. This method is widely used in fields like finance, engineering, and risk analysis to assess uncertainty and make informed decisions. 20, 26
- ML** Machine Learning: A field of study in artificial intelligence concerned with the development and study of statistical algorithms that can learn from data and generalize to unseen data, and thus perform tasks without explicit instructions. vi, 4, 11–13, 23, 29, 30, 32, 44, 46, 55
- MLP** Multilayer Perceptron: In deep learning, a multilayer perceptron is a name for a modern feedforward neural network consisting of fully connected neurons with non-linear activation functions, organized in layers, notable for being able to distinguish data that is not linearly separable. 5, 6, 57
- NLP** Natural Language Processing: A subfield of computer science and especially artificial intelligence. It is primarily concerned with providing computers with the ability to process data encoded in natural language and is thus closely related to information retrieval, knowledge representation and computational linguistics, a subfield of linguistics. Major tasks in natural language processing are speech recognition, text classification, natural-language understanding, and natural-language generation. 6, 9, 11
- NN** Neural Network: In machine learning, a neural network (also artificial neural network or neural net, abbreviated ANN or NN) is a model inspired by the structure and function of biological neural networks in animal brains. It consists of connected units or nodes called artificial neurons, which loosely model the neurons in the brain. 6, 30
- QCD** Quantum Chromodynamics: In particle physics, quantum chromodynamics is the study of the strong interaction between quarks mediated by gluons. Quarks are fundamental particles that make up composite hadrons such as the proton, neutron and pion. QCD is a type of quantum field theory called a non-abelian gauge theory, with symmetry group  $SU(3)_C$ . The QCD analog of electric charge is a property called color. Gluons are the force carriers of the theory, just as photons are for the electromagnetic force in quantum electrodynamics. 2, 17, 25
- QED** Quantum Electrodynamics: In particle physics, quantum electrodynamics is the relativistic quantum field theory of electrodynamics. In essence, it describes how light and matter interact and is the first theory where full agreement between quantum mechanics and special relativity is achieved. QED mathematically describes all phenomena involving electrically charged particles interacting by means of exchange of photons and represents the quantum counterpart of classical electromagnetism giving a complete account of matter and light interaction. 2, 3
- SM** Standard Model: The Standard Model of particle physics is the theory describing three of the four known fundamental forces (electromagnetic, weak and strong interactions – excluding gravity) in the universe and classifying all known elementary particles. 1, 14, 15, 17, 18

- SR** Signal Region: A subset of collision events defined by a specific set of selection criteria (cuts) chosen to maximize the fraction of true signal processes while suppressing background contributions. In high-energy physics analyses, the signal region is the primary phase-space where one measures the rate or properties of the physics process of interest and compares data to expectations. 25, 26
- VENV** Virtual Environment: In Python, a virtual environment is an isolated space where you can work on your Python projects separately from your system-installed Python. This allows you to set up your own libraries and dependencies without affecting the system Python installation. 29

# Contents

<b>Abstract</b>	<b>iv</b>
<b>Acknowledgements</b>	<b>v</b>
<b>List of Figures</b>	<b>vi</b>
<b>List of Tables</b>	<b>viii</b>
<b>Acronyms &amp; Glossary</b>	<b>ix</b>
<b>1 Introduction</b>	<b>1</b>
1.1 The Standard Model & the Higgs Boson . . . . .	1
1.1.1 Quarks . . . . .	2
1.1.2 Leptons . . . . .	3
1.1.3 Bosons . . . . .	3
1.2 Machine Learning Foundations for Particle Physics . . . . .	4
1.2.1 Multilayer Perceptron . . . . .	5
1.2.2 Gradient Descent . . . . .	5
1.2.3 Loss Function . . . . .	6
1.2.4 Transformer Architecture . . . . .	6
1.2.5 Embedding & Positional Encoding . . . . .	7
1.2.6 Attention . . . . .	10
1.2.7 Softmax . . . . .	11
1.2.8 Activation Functions . . . . .	11
1.2.9 Layer Normalization . . . . .	13
1.2.10 Regularization . . . . .	14
1.3 Introduction to $t\bar{t}H$ Analysis in Particle Physics . . . . .	14
1.3.1 Lagrangian . . . . .	14
1.3.2 Yukawa Coupling & Mass Generation . . . . .	15
1.3.3 Importance of $t\bar{t}H$ Production in Particle Physics . . . . .	16
1.3.4 Higgs Decay Channels . . . . .	17
1.3.5 Summary . . . . .	18
1.4 Research Objectives and Thesis Overview . . . . .	18
<b>2 The ATLAS Experiment</b>	<b>20</b>
2.1 Large Hadron Collider . . . . .	21
2.1.1 Luminosity . . . . .	21
2.2 ATLAS Detector . . . . .	22

<b>3</b>	<b>Theoretical Framework</b>	<b>23</b>
3.1	The $2\ell_{\text{SS}} + \tau_{\text{had}}$ Channel . . . . .	23
3.2	Signal Region . . . . .	25
3.3	Monte Carlo Simulation . . . . .	26
3.4	Foundations for HEP Classification . . . . .	27
3.4.1	Binary Classification Metrics . . . . .	27
3.4.2	Cross-Validation . . . . .	27
3.4.3	Confusion Matrix . . . . .	28
3.4.4	Efficiency & Background Rejection . . . . .	28
3.4.5	F <sub>1</sub> -Score . . . . .	28
3.4.6	Overtraining & Validation Strategies . . . . .	28
<b>4</b>	<b>Software Infrastructure</b>	<b>29</b>
4.1	Python . . . . .	29
4.2	LXPLUS . . . . .	29
4.3	At the Root of it All . . . . .	30
4.3.1	ROOT . . . . .	30
4.3.2	Uproot . . . . .	30
4.4	Machine Learning . . . . .	30
4.4.1	PyTorch . . . . .	30
4.4.2	Optuna . . . . .	30
4.5	TRExFitter . . . . .	31
<b>5</b>	<b>Experimental Setup &amp; Baseline Model</b>	<b>32</b>
5.1	Data Splits & Cross-Validation . . . . .	32
5.2	Feature & Dataset . . . . .	32
5.3	Normalization . . . . .	34
5.3.1	Z-score Standardization . . . . .	34
5.3.2	Feature-wise Scaling . . . . .	36
5.3.3	Correlation Heatmap . . . . .	38
5.4	Baseline Transformer Implementation . . . . .	40
5.4.1	Sequence . . . . .	40
5.4.2	Transformer Encoder . . . . .	41
5.4.3	Output . . . . .	42
5.4.4	Weight Initialization . . . . .	43
5.5	Baseline Performance . . . . .	43
5.5.1	Performance Metrics . . . . .	44
5.5.2	Model Interpretability & Internal Dynamics . . . . .	47
5.6	Network Output . . . . .	52
5.7	Performance Comparison . . . . .	53
<b>6</b>	<b>Proposed Model</b>	<b>55</b>
6.1	Overfitting . . . . .	56
6.2	Weight Initialization . . . . .	57
6.3	Training & Hyperparameters . . . . .	58
6.4	Hyperparameter Search . . . . .	59
6.5	Internal Dynamics Improvement . . . . .	60
6.5.1	Activations . . . . .	60
6.5.2	Attention . . . . .	62

6.6	Performance Metrics . . . . .	65
6.6.1	Feature-wise Scaled Dataset . . . . .	66
6.6.2	Z-score Standardized Dataset . . . . .	67
<b>7</b>	<b>Results and Discussion</b>	<b>68</b>
7.1	Final Model Performance . . . . .	68
7.1.1	Statistical Uncertainty . . . . .	68
7.1.2	Network Output . . . . .	69
7.2	Dataset Considerations . . . . .	70
7.3	Summary of Experimental Findings . . . . .	70
7.4	Implications & Future Directions . . . . .	71
<b>A</b>	<b>Appendix</b>	<b>73</b>
A.1	Event Weighting and TRExFitter NormFactor . . . . .	73
A.2	Hyperparameters . . . . .	75
A.3	Normalized Feature Distributions . . . . .	76
A.4	Software Dependencies . . . . .	79
	<b>Bibliography</b>	<b>81</b>

# Chapter 1

## Introduction

“In our current description of Nature, every particle is a wave in a field. The most familiar example of this is light: light is simultaneously a wave in the electromagnetic field and a stream of particles called photons.” [1]

### 1.1 The Standard Model & the Higgs Boson

The Standard Model of Particle Physics (SM) is a comprehensive framework that describes the forces and elementary particles governing the quantum realm. As the prevailing theory of fundamental interactions, it demonstrates remarkable theoretical self-consistency and has become the foundation of modern particle physics and high-energy experimental research worldwide. Although its predictive power is extraordinary, the model does not account for all observed phenomena and remains incomplete, falling short of a true unified field theory.

The SM consists of two main groups: fermions—the matter particles—and bosons—the force carriers. Fermions are characterized by their half-integer spin and mass properties and are divided into two groups: quarks and leptons. Bosons, on the other hand, hold integer spin and mediate the forces between fermions [1]. Fermions and bosons also have antiparticle counterparts, although bosons such as the gluon, photon,  $Z^0$  boson, and Higgs boson are considered self-conjugate particles, meaning they are their own antiparticles. Antiparticle versions of standard particles differ in additive quantum numbers, such as electric charge and lepton or baryon number, which reverse signs [2].

# Standard Model of Elementary Particles

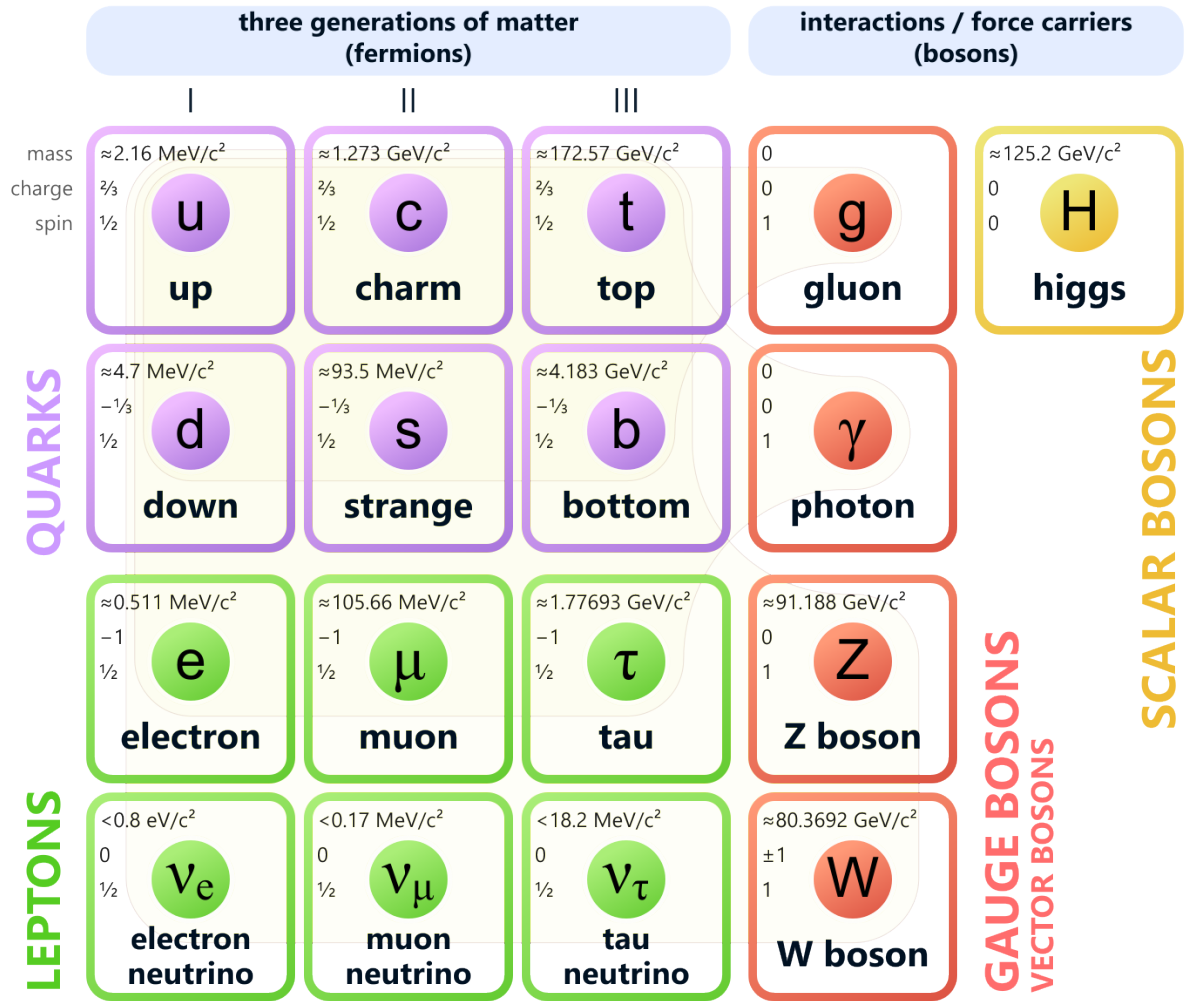


Figure 1.1: Standard Model of Particle Physics.

## 1.1.1 Quarks

Quarks obey the strong force, as described by QCD, a quantum field theory that shares many similarities with quantum electrodynamics (QED). The QCD analog to electric charge is a property called color, mediated by a force carrier called a gluon. In contrast to photons, gluons carry color charge themselves, which contributes to the complexity of QCD, as gluons can interact with each other. Each quark carries one of three color charges—red, green, or blue—and can exchange these charges dynamically through gluon interactions. The property of color leads to a set of interesting phenomena, one of which is that color symmetry dictates the way quarks combine. In simple terms, hadrons, which are composite particles consisting of quarks bound together by the strong force, are required to be color neutral, meaning that the individual quark color charges combine to



cancel one another out, forming an overall color singlet state. The requirement for color neutrality also facilitates the confinement phenomenon, where quarks are never found in isolation but only within hadrons. Quarks carry electric charge, specifically in  $\frac{2}{3}$  and  $-\frac{1}{3}$  variations, which subject them to electromagnetic interactions. They also possess intrinsic angular momentum; each quark has a spin of  $\frac{1}{2}$  and a characteristic mass that distinguishes it among the various quarks [3]. We categorize the quarks into six distinct flavors organized into three generations: the first generation comprises the up and down quarks, the second includes the charm and strange quarks, and the third consists of the top and bottom quarks.

### 1.1.2 Leptons

Leptons do not carry color charge and thus do not interact with the strong force. Instead, they are manipulated by the weak force. Neutrinos, a type of lepton lacking electric charge, engage only in weak force interactions mediated by the  $W^\pm$  and  $Z^0$  bosons. The  $W^\pm$  bosons facilitate charged current processes, in which a neutrino is converted to a charged lepton (or vice versa), and the  $Z^0$  boson mediates neutral current processes, where the neutrino remains unchanged. Charged leptons, coming in only one variation of charge,  $-1$ , participate in both weak interactions and electromagnetic interactions, the latter of which is mediated by the photon as described by QED. Similar to quarks, leptons exhibit a spin of  $\frac{1}{2}$ , including the neutrinos. Leptons are also categorized into 6 distinct flavors among 3 generations: generation one includes the electron and the electron neutrino, the second generation is comprised of the muon and the muon neutrino, and finally, the third generation consists of the tau and the tau neutrino [4].

### 1.1.3 Bosons

As mentioned earlier, bosons act as force mediators for fermions and are characterized by integer spin. Gluons mediate the strong force, photons mediate the electromagnetic force,  $W^\pm$  and  $Z^0$  bosons mediate the weak force. The Higgs boson is not a force mediator in the traditional sense; instead, it is the quantum excitation of the Higgs field. Unlike a vector field, the Higgs field is special because it is the first scalar field found in particle physics [5]. It is responsible for giving mass to elementary particles such as quarks and leptons. In other words, the mass of these particles is proportional to the strength of their coupling to the Higgs field. However, most of the mass of hadrons, such as protons and neutrons, does not come from the Higgs mechanism but from the binding energy of the strong interaction. Particles like the photon that do not interact directly with the Higgs field are therefore massless [1].

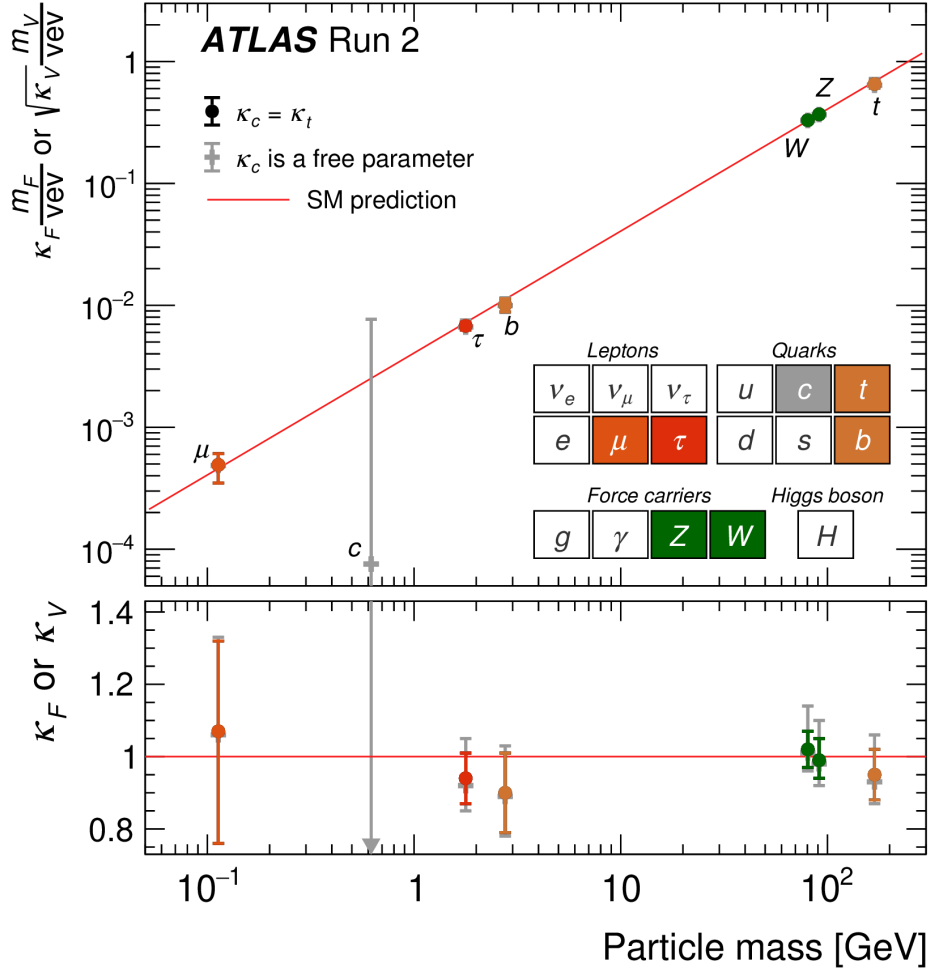


Figure 1.2: Higgs boson coupling to other elementary particles [1].

Figure 1.2 was produced by the ATLAS experiment at CERN in July of 2022. It visualizes the relationship between the mass of a particle (horizontal axis) and its coupling strength to the Higgs boson (vertical axis). The measured Higgs couplings for a range of particles (both quarks and leptons) align extremely closely with the linear relationship predicted by the Standard Model. The figure above is a plot generated by the TRExFitter software [6]. Plots such as Figure 1.2 visually compare the observed events to those predicted by theoretical models, highlighting the degree of agreement with the Standard Model expectations. Examining the Higgs boson's decay channels, such as  $H \rightarrow b\bar{b}$ ,  $H \rightarrow WW^*$ , or  $H \rightarrow \tau^+\tau^-$  final states, provides additional validation of its linear coupling behavior and confirms the mass generation mechanism prediction.

## 1.2 Machine Learning Foundations for Particle Physics

Machine Learning (ML) is a field of study in artificial intelligence focused on approximating complex functions and finding patterns in data. It is used extensively in practical

physics research due to the presence of large and highly complex datasets that are challenging for humans to interpret and extract value from.

### 1.2.1 Multilayer Perceptron

A basic formulation of a multilayer perceptron (MLP), a simple neural network used to predict a continuous target variable, consists of a set of  $n$  observations given as pairs  $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ , where  $\mathbf{x}_i \in \mathbb{R}^d$  represents a  $d$ -dimensional input vector and  $y_i \in \mathbb{R}$  the target variable. The goal is to learn parameters for a network that best predicts new data the model has not seen. A standard MLP with one hidden layer is defined by

$$\mathbf{z}_i = \mathbf{W}^{(1)} \mathbf{x}_i + \mathbf{b}^{(1)}, \quad (1.1)$$

$$\mathbf{h}_i = \phi(\mathbf{z}_i), \quad (1.2)$$

$$\hat{y}_i = \mathbf{W}^{(2)\top} \mathbf{h}_i + b^{(2)}, \quad i = 1, \dots, n, \quad (1.3)$$

where

- $\mathbf{W}^{(1)} \in \mathbb{R}^{m \times d}$  and  $\mathbf{b}^{(1)} \in \mathbb{R}^m$  are the weights and biases of the hidden layer of size  $m$ ,
- $\phi(\cdot)$  is an element-wise activation function (Subsection 1.2.8) that introduces non-linearity [7],
- $\mathbf{W}^{(2)} \in \mathbb{R}^{1 \times m}$  and  $\mathbf{b}^{(2)} \in \mathbb{R}$  are the weights and bias of the output layer.

The MLP transforms each input  $\mathbf{x}_i$  through a series of steps to produce a predicted output  $\hat{y}_i$ . The input is first combined with weights and biases in the hidden layer, then passed through an activation function to capture nonlinear patterns, and finally processed by the output layer to generate the prediction. The activation function is essential for modeling complex relationships in the data [7]. To train the MLP, the parameters  $\mathbf{W}^{(1)}, \mathbf{b}^{(1)}, \mathbf{W}^{(2)}, \mathbf{b}^{(2)}$  must be tuned to make the predictions  $\hat{y}_i$  as close as possible to the true targets  $y_i$ . This is achieved by defining a loss function, often the mean squared error, which computes the difference between predicted and actual values. The parameters are then optimized using gradient descent, an algorithm that iteratively adjusts the parameters to minimize the loss by following the direction of steepest descent [8].

### 1.2.2 Gradient Descent

Solving for the optimal parameters in neural networks is a non-trivial task, especially when dealing with DNNs. Gradient descent is used to iteratively update the weights

of the NN by computing the gradient of the loss and moving in the steepest direction, effectively trying to reduce the loss for the next run. The general equation for gradient descent is

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla_{\theta} L(\theta^{(t)}) \quad (1.4)$$

where  $\eta$  is the learning rate,  $\theta$  are the learnable weights, and  $L(\theta)$  is the loss [9].

### 1.2.3 Loss Function

For classification tasks we use the cross-entropy loss, which measures the difference between the true labels and the model's predicted probability distribution. Let  $\{(x_i, y_i)\}_{i=1}^n$  be our training set, with  $y_i$  being the true class label  $y_{i,k} \in \{0, 1\}$  over  $K$  classes, in our case two, and let

$$\hat{p}_{i,k} = \text{softmax}(z_i)_k \quad (1.5)$$

be the predicted probability for class  $k$  on input  $x_i$ . Thus the (average) cross-entropy loss is

$$L(\theta) = -\frac{1}{n} \sum_{i=1}^n \sum_{k=1}^K y_{i,k} \log \hat{p}_{i,k}. \quad (1.6)$$

Minimizing this loss via gradient descent is how we can train a network to favor correct labeling [10].

### 1.2.4 Transformer Architecture

Standard MLPs often struggle to capture the intricate feature correlations present in particle-physics data. A wide variety of models with different architectures have been used for high-energy physics (HEP), but a recent contender that has been dominating in the natural language processing (NLP) field is quickly gaining popularity in multiple domains [8].

The transformer is a deep neural network (DNN) architecture, visualized in Figure 1.3, that is characterized by the attention mechanism. It was originally developed by Google and released in 2017 with the release of the *Attention is All You Need*<sup>1</sup> paper. Although it was originally designed for NLP, it has since been adapted to be used in almost every domain previously dominated by more traditional architectures.

---

<sup>1</sup><https://arxiv.org/abs/1706.03762>

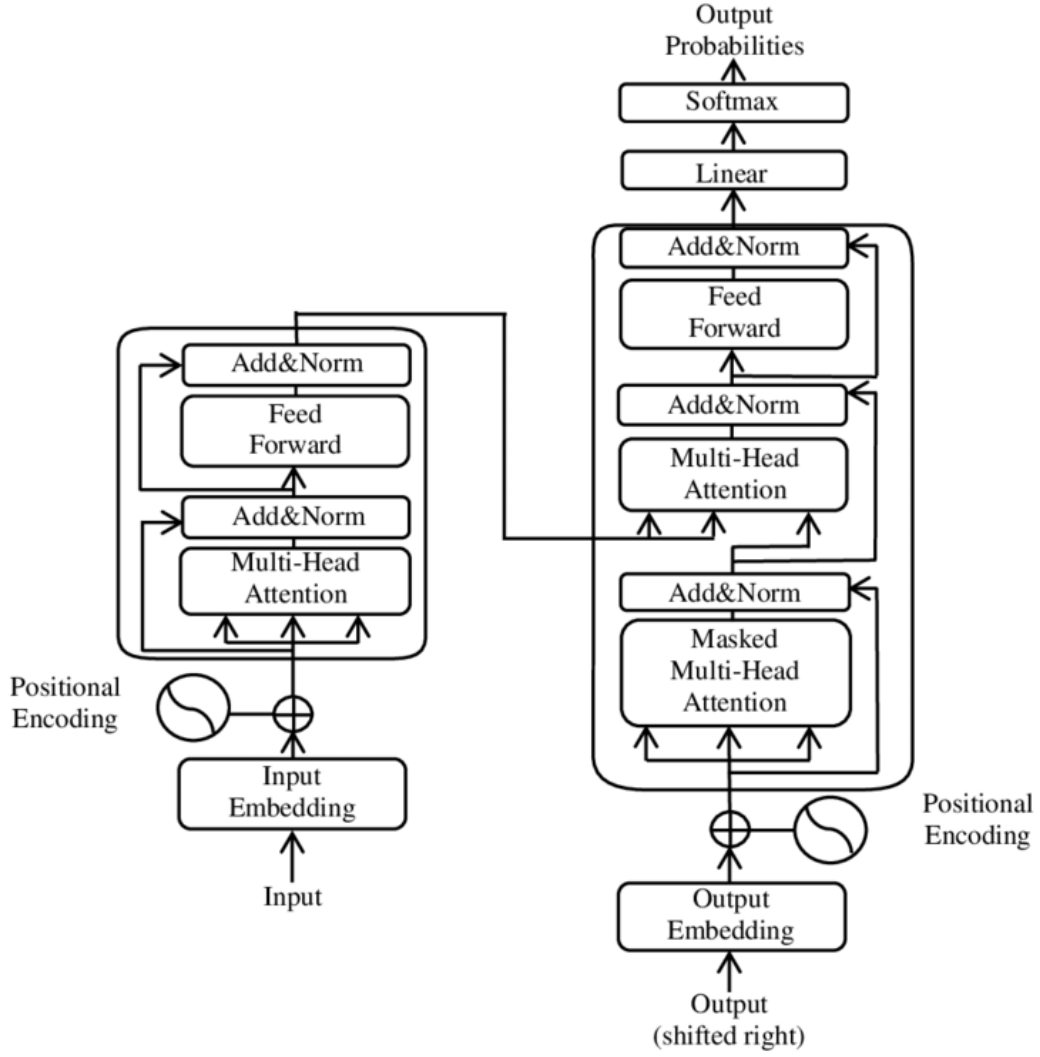


Figure 1.3: Illustration of the transformer model introduced by the *Attention is All You Need* paper.

### 1.2.5 Embedding & Positional Encoding

The shining star of the transformer is its attention mechanism. Although early versions have existed since the fifties, modern applications started with the paper *Neural Machine Translation by Jointly Learning to Align and Translate*<sup>2</sup>. But working with attention can be trickier than traditional convolution or linear layers. Transformers often use an input embedding strategy that casts an input sequence into a more desirable shape.

Let the input sequence be represented as

$$\mathbf{x} = \{x_1, x_2, \dots, x_n\} \quad (1.7)$$

where each element, also called a token,  $x_j$  is mapped to a continuous vector through

<sup>2</sup><https://arxiv.org/abs/1409.0473>

an embedding function

$$\mathbf{e}(x_j) \in \mathbb{R}^{d_{model}}. \quad (1.8)$$

Thus, for the entire sequence, we form the embedded representation

$$\mathbf{E} = \{\mathbf{e}(x_1), \mathbf{e}(x_2), \dots, \mathbf{e}(x_n)\} \in \mathbb{R}^{n \times d_{model}} \quad (1.9)$$

where  $d_{model}$  is the model dimensionality. The embedding function can be a fixed or a learned embedding. In *Attention is All You Need*, the authors use a straightforward yet effective embedding function to map discrete tokens into continuous  $\mathbb{R}^{d_{model}}$  space. This process is handled via a learned embedding matrix, which acts as a lookup table. Below is an explanation of the embedding function used in the 2017 paper.

Let  $V$  denote the vocabulary size and define the embedding matrix as

$$\mathbf{E} \in \mathbb{R}^{V \times d_{model}} \quad (1.10)$$

where each row  $\mathbf{E}_i$  corresponds to the  $d_{model}$ -dimensional embedding of the  $i$ -th token from the vocabulary. This matrix is learned during training, meaning that the model adjusts the values in  $\mathbf{E}$  so that tokens with similar contexts acquire similar representations.

For an input sequence of tokens

$$\mathbf{x} = \{x_1, x_2, \dots, x_n\}, \quad \text{with } x_j \in \{1, 2, \dots, V\} \quad (1.11)$$

the embedding function maps each token  $x_j$  to its corresponding embedding vector by using  $\mathbf{E}$  as a lookup table:

$$\mathbf{e}(x_j) = \mathbf{E}_{x_j} \in \mathbb{R}^{d_{model}}. \quad (1.12)$$

This means that for every token, its raw integer index is replaced by a continuous vector that captures semantic information in  $d_{model}$ -dimensional space.

Since the scale of the embedding vectors might vary, and to ensure stable gradients during training, the model scales each embedding by the factor  $\sqrt{d_{model}}$ :

$$\tilde{\mathbf{e}}(x_j) = \sqrt{d_{model}} \cdot \mathbf{e}(x_j). \quad (1.13)$$

This scaling serves to normalize the magnitude of the embeddings, which is important when working with layers that are sensitive to the absolute scale of the inputs, such as attention and softmax.

Therefore, the final embedded representation for the input sequence is assembled by

concatenating these scaled vectors:

$$\mathbf{E}_{\mathbf{x}} = [\tilde{\mathbf{e}}(x_1), \tilde{\mathbf{e}}(x_2), \dots, \tilde{\mathbf{e}}(x_n)] \in \mathbb{R}^{n \times d_{model}}. \quad (1.14)$$

Each row in  $\mathbf{E}_{\mathbf{x}}$  now represents a token from the original sequence in a continuous and normalized vector space. This approach efficiently converts discrete tokens into continuous vectors that serve as the input, but it leaves out important information about the input sequence that transformer models need to be effective [11].

For most applications, the attention mechanism is paired with a positional encoding of the input. This is because the attention is inherently permutation invariant, meaning it cannot extract information from the order in which the input vector is arranged. For tasks such as NLP and many others, positional data must be encoded into inputs. A clever way of doing this is to compute a frequency function that incorporates the token's position. For all even indices in the input embedding, we calculate

$$\mathbf{PE}_{(pos, 2i)} = \sin \left( \frac{pos}{10000^{\frac{2i}{d_{model}}}} \right) \quad (1.15)$$

and for all odd

$$\mathbf{PE}_{(pos, 2i+1)} = \cos \left( \frac{pos}{10000^{\frac{2i}{d_{model}}}} \right) \quad (1.16)$$

where

- $pos = 0, 1, \dots, N-1$  is the position index in the sequence (with  $N$  being the sequence length), and
- $i = 0, 1, \dots, \frac{d_{model}}{2} - 1$  indexes the dimensions [12].

The resulting Positional Encoding matrix,  $\mathbf{PE}$ , has dimensions  $N \times d_{model}$ . Now, to finally produce viable inputs for our transformer model, we can sum the terms by performing an element-wise sum:

$$\text{Input Matrix} = \text{Embedding Matrix} + \mathbf{PE}, \quad \text{with Input Matrix} \in \mathbb{R}^{N \times d_{model}}. \quad (1.17)$$

In this thesis, we forgo using any kind of positional encoding. The order of the features is arbitrary and remains constant, and unnecessary encodings may taint the sensitive correlations between features, but it is helpful to understand how attention-based models are position-invariant and how researchers get around this limitation.

### 1.2.6 Attention

In Figure 1.3, we see the multi-head version of attention being used. To form a solid understanding, we will start with single-head self-attention, which we will just call self-attention, where the inputs to the attention layer all come from the same sequence. The basis of the self-attention mechanism starts by computing 3 separate matrices from our input and 3 learned weight matrices:

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V, \quad (1.18)$$

where  $W_Q$ ,  $W_K$ , and  $W_V$  are learned weight matrices, query ( $Q$ ), key ( $K$ ) and value ( $V$ ) matrices, respectively, that get updated and tweaked through backpropagation, and  $X$  is the input matrix calculated in Subsection 1.2.5. With the inputs calculated, we can define the attention mechanism as

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V, \quad (1.19)$$

where  $d_k$  is the dimensionality of the key vectors. The term  $d_k$  is derived from

$$d_k = \frac{d_{\text{model}}}{h} \quad (1.20)$$

where  $d_{\text{model}}$  is the dimension of the embedding space and  $h$  is the number of attention heads, in the case of single-head attention, one.

For multi-head attention, instead of a single projection, we split the input into  $h$  subspaces and learn separate weight matrices for each head. For each head  $i$  ( $i = 1, \dots, h$ ), we compute:

$$Q_i = XW_Q^i, \quad K_i = XW_K^i, \quad V_i = XW_V^i. \quad (1.21)$$

Then, each head computes its attention output:

$$\text{head}_i = \text{Attention}(Q_i, K_i, V_i) = \text{softmax}\left(\frac{Q_i K_i^T}{\sqrt{d_k}}\right) V_i. \quad (1.22)$$

The outputs from all heads are concatenated and projected:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h) W_O, \quad (1.23)$$

where  $W_O$  is an extra learned weight matrix, known as the output projection matrix. This approach allows the model to capture different aspects of the input data simultaneously and allows for more complex relationship building [11].

In this thesis, the attention mechanism is applied feature-wise, meaning that attention



is computed across the feature dimension for each individual sample in the batch. Each attention head operates over the set of input features, treating the feature axis as the sequence dimension. This enables the model to learn relationships between different features within each sample, rather than across time steps or token positions as is typical in NLP applications.

### 1.2.7 Softmax

The softmax function is a key function used in many ML applications. It is described by the equation

$$\text{softmax}(z)_i = \frac{e^{z_i}}{\sum_{j=1}^k e^{z_j}}, \quad i = 1, 2, \dots, k. \quad (1.24)$$

The softmax function converts a vector of  $k$  real numbers into a probability distribution over  $k$  items, each in the interval  $(0, 1)$ , such that the sum of the probabilities equals one. It is an extremely useful function that is used extensively in ML, especially to convert a model's raw outputs, known as logits, into a normalized probability distribution over all possible classes, thereby enabling the extraction of a final decision based on the highest probability. In the context of the attention mechanism, softmax normalizes the scaled dot-product scores into a set of attention weights. These weights highlight the relative importance of different elements in the input sequence [13].

We use softmax to extract the *signal* versus *background* probabilities from model logits in this thesis.

### 1.2.8 Activation Functions

Neural networks work because they are composed of many artificial neurons, usually working together in layers. The artificial neuron can be considered the building block and the workhorse of the neural network. In its vectorized form, the artificial neuron is defined as

$$y = \phi(\mathbf{w}^\top \mathbf{x} + b), \quad (1.25)$$

where

- $\mathbf{x} \in \mathbb{R}^n$  is the input vector,
- $\mathbf{w} \in \mathbb{R}^n$  is the weight vector,
- $b \in \mathbb{R}$  is the bias, a scalar value, and

- $\phi(\cdot)$  denotes the scalar activation function.

Activation functions are the nonlinear transformations applied to a neuron’s pre-activation that introduce nonlinearity into the network. Having nonlinear activation functions is extremely important because they are essential for learning complex patterns.

The idea for nonlinear activation functions came from nature, as real neurons “spike” or “fire” when an input threshold is reached. A historically important activation function is the Heaviside function, defined as

$$\phi(z) = \begin{cases} 1, & z \geq 0, \\ 0, & z < 0, \end{cases} \quad (1.26)$$

where  $z$  is the term in Equation 1.25 inside the parenthesis

$$z = \mathbf{w}^\top \mathbf{x} + b. \quad (1.27)$$

Also known as a step function, the Heaviside function is nondifferentiable, making gradient descent impossible and creating a need for smooth activation functions.

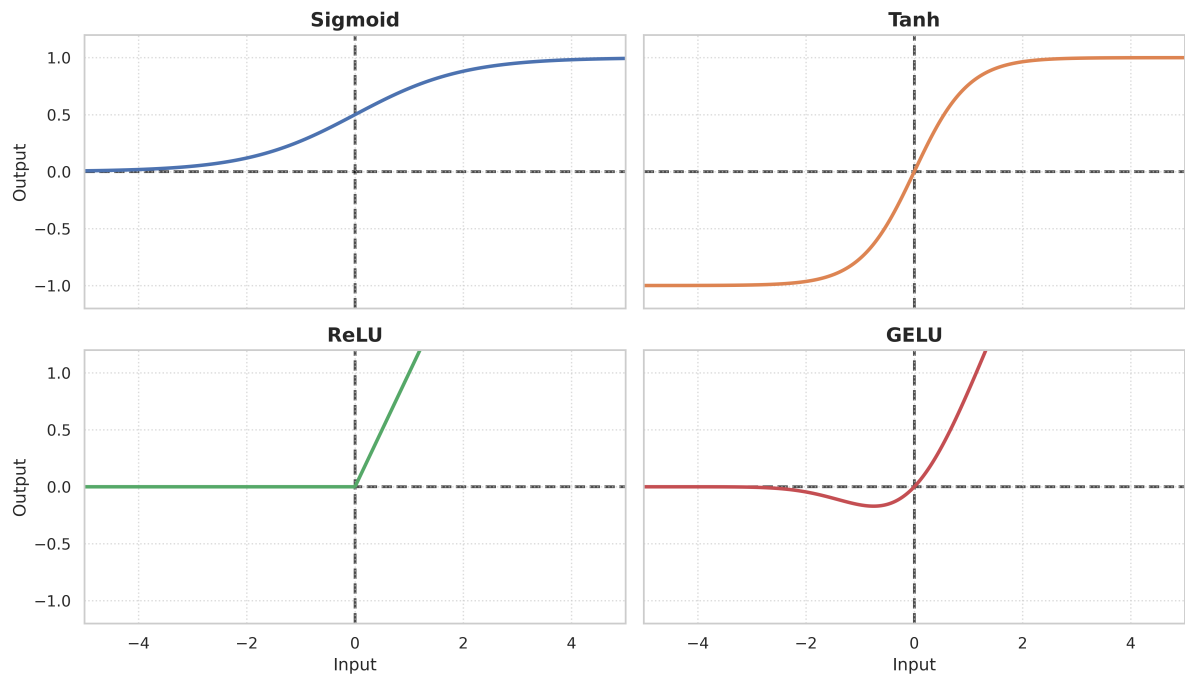


Figure 1.4: Activation functions commonly found in ML applications.

Figure 1.4 shows a collection of popular activation functions for ML projects of today. All functions in the figure have been used successfully to produce impressive models, but the popularity of ReLU and the performance of GELU stand out above the rest. ReLU [7] is a simple function, defined as

$$\text{ReLU}(z) = \max(0, z). \quad (1.28)$$

Equation 1.2.8 is easy to compute, but can lead to “dead” neurons when  $z < 0$ . Modifications to ReLU exist, but in 2016, Hendrycks & Gimpel proposed a new function that leads to smoother gradients<sup>3</sup>. It has since been used in popular transformer-based models like BERT. Named the GELU function, it is defined as

$$\text{GELU}(z) = z \Phi(z) \quad \text{where} \quad \Phi(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^z e^{-t^2/2} dt. \quad (1.29)$$

GELU uses the Gaussian CDF ( $\Phi(z)$ ) to gate activations. It is smooth everywhere, provides continuous gradients, and generally improves model performance, especially in DNNs. Due to its complexity, it is 10-20% slower than ReLU, but we have selected it for this thesis due to its stability, faster convergence, and superior accuracy in transformer-style models.

### 1.2.9 Layer Normalization

Another important feature of modern ML models is layer normalization. First introduced in 2016 by the *Layer Normalization*<sup>4</sup> paper, it stabilizes training by normalizing the activations of each layer (the output vector) across its hidden units for each input vector. The normalization starts by computing the mean ( $\mu$ ) and the variance ( $\sigma^2$ ):

$$\mu = \frac{1}{H} \sum_{i=1}^H x_i, \quad \sigma^2 = \frac{1}{H} \sum_{i=1}^H (x_i - \mu)^2, \quad (1.30)$$

Then the distribution parameters are used to compute the individual output values of the input vector. Here  $\epsilon$  is a small constant for numerical stability:

$$\hat{x}_i = \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}}. \quad (1.31)$$

This is usually paired with

$$y_i = \gamma_i \hat{x}_i + \beta_i, \quad (1.32)$$

where  $\gamma_i, \beta_i$  are learnable scale and shift parameters that allow the model to restore any needed scale and shift after normalization. Layer normalization has since become a

---

<sup>3</sup><https://arxiv.org/abs/1606.08415>

<sup>4</sup><https://arxiv.org/abs/1607.06450>

standard component in modern deep architectures, especially transformers, for its ability to stabilize and accelerate training.

### 1.2.10 Regularization

Regularization is a collection of techniques such as weight decay, data augmentation, and early stopping, designed to prevent a model from overfitting to its training data. Common penalty-based methods include L1 regularization, which encourages sparsity by penalizing the absolute values of weights, and L2 regularization, which discourages large weights by penalizing their squared magnitude.

Dropout is a simple but effective regularization method in which, during training, each neuron’s output is zeroed with probability  $p$  to prevent overfitting. At test time, outputs are scaled by  $1 - p$  to account for the missing activations. Regularization techniques aim to mend situations where the model performs extremely well on training data but fails to generalize well enough to perform well on testing data.

Residual connections, first popularized by ResNet<sup>5</sup>, add an identity skip so that each block learns a residual function  $F(x)$  and outputs  $y = F(x) + x$ . This design mitigates the vanishing-gradient problem in very deep networks and makes it easier to optimize over hundreds of layers. In simpler terms, they let the network pass the original input straight through and only learn the small tweaks needed on top, so each layer does not have to reinvent the entire transformation from scratch.

Together, dropout and residual connections have become standard tools: dropout to improve generalization and residuals to enable the training of extremely deep architectures without degradation of accuracy.

## 1.3 Introduction to $t\bar{t}H$ Analysis in Particle Physics

The top quark, the biggest fermion particle in the SM, has a strong Yukawa coupling with the Higgs boson [14]. The Higgs boson and top-antitop quark pair ( $t\bar{t}H$ ) production provides a direct test of the SM’s predictions for the Higgs sector and is a unique probe of the mass generating mechanism. In this section, we briefly explain the significance of the  $t\bar{t}H$  channel and discuss the role of the Yukawa coupling.

### 1.3.1 Lagrangian

In order to fully understand Yukawa coupling, it is essential to first grasp the framework that encapsulates all of the dynamics of a physical system — the Lagrangian.

---

<sup>5</sup><https://arxiv.org/abs/1512.03385>

The Lagrangian density, usually just called the Lagrangian, is a compact way to encode how fields behave and interact. It depends on the fields  $\phi_i$  and their spacetime derivatives  $\partial_\mu \phi_i$ :

$$\mathcal{L} = \mathcal{L}(\phi_i, \partial_\mu \phi_i). \quad (1.33)$$

By integrating over all spacetime (three spatial dimensions and one time dimension),

$$S = \int \mathcal{L} d^4x, \quad (1.34)$$

we obtain the action  $S$ . Demanding that the action be stationary under small variations of the fields, we arrive at the equation

$$\delta S = 0, \quad (1.35)$$

which leads to the Euler-Lagrange equations,

$$\frac{\partial \mathcal{L}}{\partial \phi_i} - \partial_\mu \left( \frac{\partial \mathcal{L}}{\partial (\partial_\mu \phi_i)} \right) = 0, \quad (1.36)$$

which are simply the equations of motion for each field [15]. In practice, one builds  $\mathcal{L}$  out of three types of terms:

- *Kinetic terms* involving  $\partial_\mu \phi_i$ ,
- *Mass terms* like  $m^2 \phi_i^2$ ,
- *Interaction terms*, which tell us how fields interact with one another.

### 1.3.2 Yukawa Coupling & Mass Generation

The next step is to introduce the simplest fermion-Higgs interaction, the Yukawa coupling, which is responsible for giving fermions their masses.

The Yukawa coupling is a fundamental parameter in the SM, governing the strength of the interaction between the Higgs field and the fermions. The interaction term in the Lagrangian for a fermion  $\psi$  with the Higgs field  $\phi$  is given by

$$\mathcal{L}_{\text{Yukawa}} = -y_f \bar{\psi}_L \phi \psi_R + \text{h.c.} \quad (1.37)$$

Here,  $y_f$  is the Yukawa coupling constant for fermion  $f$ ,  $\bar{\psi}_L$  and  $\psi_R$  are the left-handed and right-handed components of the fermion field, and “h.c.” denotes the Hermitian conjugate. When the Higgs field acquires its vacuum expectation value  $v$ , this term generates a fermion mass

$$m_f = y_f \frac{v}{\sqrt{2}}. \quad (1.38)$$

Since the mass of a fermion is directly proportional to its Yukawa coupling, heavier fermions are associated with larger Yukawa couplings (Figure 1.2). The top quark, with its mass of approximately 173 GeV, implies a Yukawa coupling  $y_t$  of order one [16]. This strong coupling makes the  $t\bar{t}H$  production channel a direct and sensitive probe of the top-Higgs interaction.

### 1.3.3 Importance of $t\bar{t}H$ Production in Particle Physics

When analyzing Feynman diagrams of decay chains, researchers often include loops. These loop diagrams—closed circuits of virtual particles—represent quantum fluctuations where particles pop in and out of existence for an instant. These instantaneous particles, called virtual particles, tweak the probability of a process happening and make theoretical predictions more accurate, but are also more challenging to calculate. These virtual particle loops drastically increase the complexity of the calculations.

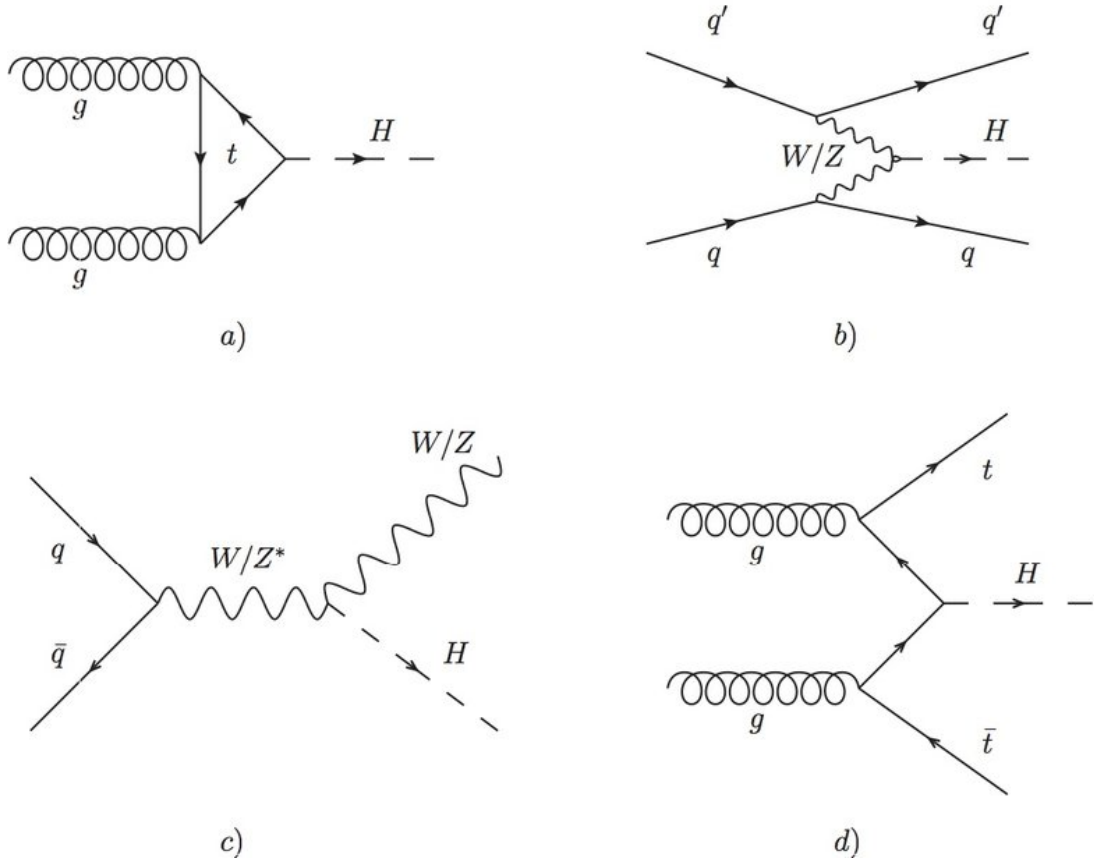


Figure 1.5: Representative Feynman diagrams for the four main Higgs boson production mechanisms at the LHC: (a) gluon fusion ( $gg \rightarrow H$ ), (b) vector-boson fusion (VBF), (c) associated  $VH$  production ( $q\bar{q}' \rightarrow VH$ ), and (d)  $t\bar{t}H$  production ( $gg \rightarrow t\bar{t}H$ ) [17].

The Higgs boson at the LHC can be produced through several distinct mechanisms, as shown in Figure 1.5:

- **Gluon fusion (ggF):** involves virtual quark loops—requires complex loop integrals and renormalization.
- **Vector-boson fusion (VBF):** a tree-level electroweak exchange of  $W/Z$  bosons—no loops, so the calculation is straightforward.
- **Associated  $VH$  production:** tree-level  $q\bar{q}' \rightarrow VH$  process—direct emission of  $H$  with no hidden virtual loops.
- **Associated  $t\bar{t}H$  production:** our chosen production signal, tree-level radiation of  $H$  off a top quark—minimal virtual effects and easiest to interpret [18].

Of all the Higgs boson production modes at the LHC,  $t\bar{t}H$  stands out as a tree-level process that couples directly to the top-Higgs Yukawa interaction without the need for complicated virtual loops. Its clear Feynman topology makes theoretical calculations and experimental measurements more straightforward, while its sensitivity to the heaviest SM fermion, the top particle, provides a uniquely direct probe of the mass generation mechanism. For these reasons, the  $t\bar{t}H$  production signal forms the cornerstone of our analysis.

### 1.3.4 Higgs Decay Channels

The Higgs particle does not last very long (around  $2.1 \times 10^{-22}$  s or 210 yoctoseconds) [19] and decays into a number of different end states. For this reason, confirming the presence of a  $t\bar{t}H$  signal relies on careful analysis of the end state of the event. To be able to determine what end states, or final states, we should be looking for, we need to understand the decay channels of the Higgs boson.

Known decay channels provide us with a way to intelligently select what final states have a chance of being produced with the help of a Higgs boson. Table 1.1 lists the largest decay fraction of the Higgs is into  $b\bar{b}$ , but this channel suffers from overwhelming QCD backgrounds in a busy  $t\bar{t}H$  environment [21]. The  $W^*W$  mode offers a significant rate and clean leptonic signatures, but the presence of multiple neutrinos complicates full event reconstruction. In contrast, the  $\tau^+\tau^-$  ( $\tau\bar{\tau}$ ) channel, with a branching ratio of about 6.3%, provides a compelling compromise:

- The hadronic decays of the tau produce narrow, low-multiplicity jets (one- or three-prong) that can be identified with high purity.

Decay mode	Partial width [MeV]	Branching ratio [%]
$H \rightarrow b\bar{b}$	2.38	58.2
$H \rightarrow W^*W$	0.88	21.5
$H \rightarrow gg$	0.34	8.6
$H \rightarrow \tau^+\tau^-$	0.26	6.3
$H \rightarrow c\bar{c}$	0.12	2.9
$H \rightarrow ZZ^*$	0.11	2.6
$H \rightarrow \gamma\gamma$	0.0086	0.23
$H \rightarrow Z\gamma$	0.0063	0.15
$H \rightarrow \mu^+\mu^-$	0.00089	0.022

Table 1.1: Partial widths and branching ratios for the main Higgs-boson decay channels at  $m_H \simeq 125$  GeV [20].

- Taus leave distinctive, narrow jets in the detector, which makes them easier to identify.
- When paired with the charged leptons from the top-quark decays, the  $\tau^+\tau^-$  mode leads to distinctive final states such as two same-sign leptons plus a hadronically decaying  $\tau$  ( $2\ell_{\text{SS}} + \tau_{\text{had}}$ ), which is the focus of our analysis.

By targeting the  $\tau^+\tau^-$  decay of the Higgs in association with the top-quark pair, we can exploit both the moderate branching fraction and the clean experimental signature.

### 1.3.5 Summary

Fundamentally, the significance of  $t\bar{t}H$  analysis lies in its direct access to the top-Higgs Yukawa coupling, which is a cornerstone of the mass generation mechanism in the SM. By probing  $t\bar{t}H$  production, researchers can test the SM's predictions and investigate potential signals of new physics.

## 1.4 Research Objectives and Thesis Overview

The differentiation of signal from background in HEP remains a challenging task, yet it is pivotal for advancing our understanding of fundamental particles and interactions. In this thesis, we focus on one of the most important signals, the  $t\bar{t}H$  process, as it provides a direct probe into the top-Higgs Yukawa coupling and, potentially, physics beyond the SM.

The main technical focus is to explore and optimize various transformer architectures and hyperparameters, test different normalization techniques and attention mechanisms, and experiment with novel activation functions and regularization methods to efficiently



and accurately differentiate  $t\bar{t}H$  signals from overwhelming background processes. Specifically, the research objectives are:

- **Signal Differentiation:** Combine optimized transformer architectures with novel activation and regularization schemes to robustly distinguish the  $t\bar{t}H$  signal from background events.
- **Model and Technique Optimization:** Systematically explore transformer variants, normalization methods, attention mechanisms, activation functions, and regularization approaches to maximize classification performance.
- **Impact on High-Energy Physics:** Enhance the sensitivity and precision of  $t\bar{t}H$  analyses, enabling deeper probes of the Higgs sector and potential new physics beyond the Standard Model.

This work applies advanced machine learning techniques to experimental particle physics, with the goal of improving the accuracy and computational efficiency of HEP analyses. Notably, this is the first  $t\bar{t}H$  analysis to utilize ATLAS software Release 22, the latest major version of the ATLAS reconstruction and analysis framework. Release 22 incorporates a suite of enhanced calibrations, including refined jet energy scale corrections and significantly improved b-tagging algorithms for more accurate heavy-flavor jet identification.

## Chapter 2

# The ATLAS Experiment

CERN, the European Organization for Nuclear Research, is a world-renowned research organization located in Geneva, Switzerland. Established on September 29<sup>th</sup>, 1954, by 12 countries in Western Europe, CERN leads the world in nuclear and HEP research and experimentation. It also facilitates the Large Hadron Collider (LHC), which is the world's largest particle accelerator and has been performing collisions since 2010 [22]. The LHC has been pivotal in numerous breakthroughs, most notably the discovery of the Higgs boson, and it serves as the foundation for the Monte Carlo simulations (MCS) that emulate the Run 2 data used in our analysis.

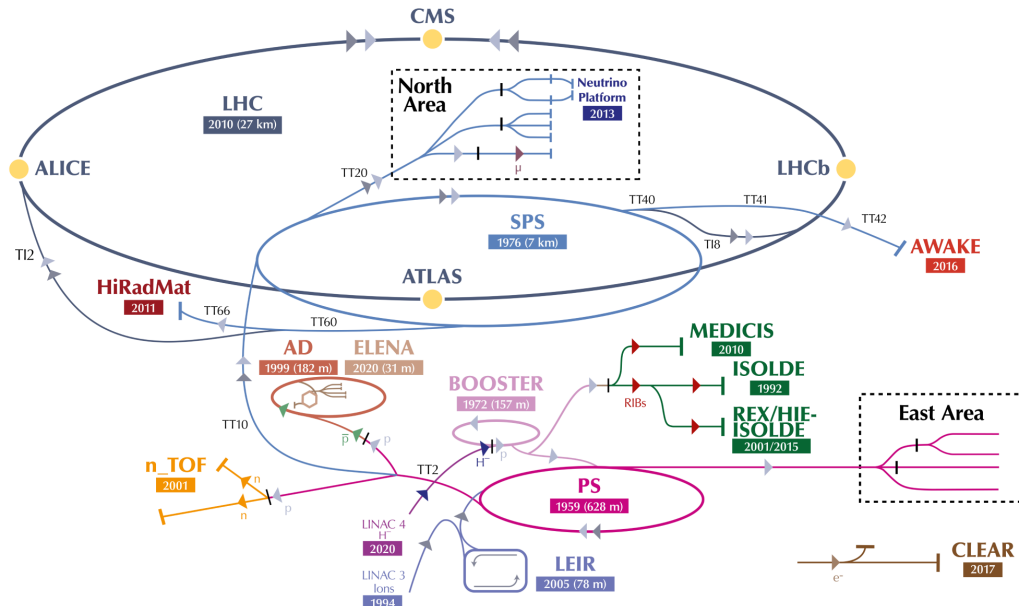


Figure 2.1: CERN accelerator complex.

## 2.1 Large Hadron Collider

The LHC, illustrated in Figure 2.1, is a 27-kilometer-long feat of engineering. Made from thousands of superconducting magnets and filled with dozens of tons of superfluid helium, it is a testament to human engineering.

The LHC is capable of delivering center-of-mass energies up to  $\sqrt{s} = 13$  TeV (tera-electronvolts) and achieving impressive peak luminosities exceeding  $2 \times 10^{34} \text{ cm}^{-2} \text{ s}^{-1}$  [23]. It consists of four major experiments—ATLAS and CMS, the two general-purpose detectors, as well as LHCb and ALICE, which specialize in heavy-flavor and heavy-ion physics, respectively. In this analysis, we focus on data recorded by the ATLAS detector, whose inner tracking system, calorimeters, and muon spectrometer provide the reconstruction of  $t\bar{t}H$  final states and robust identification of b-jets via advanced b-tagging algorithms.

### 2.1.1 Luminosity

In collider physics, the instantaneous luminosity, denoted as  $L$ , quantifies the rate at which particle collisions occur. From the instantaneous luminosity, we can calculate the collision rate,  $R$ , by using

$$R = L \times \sigma. \quad (2.1)$$

where  $\sigma$  is the cross section of our beam.

The integrated luminosity  $\mathcal{L}$ , measured in inverse femtobarns ( $\text{fb}^{-1}$ ), is the time-integral of the instantaneous luminosity

$$\mathcal{L} = \int L(t) dt. \quad (2.2)$$

It represents the total data sample available, so a run with cross section  $\sigma$  will yield

$$N = \sigma \mathcal{L} \quad (2.3)$$

events in that dataset.

High integrated luminosities are crucial for observing rare processes like  $t\bar{t}H$  production because they directly scale the number of signal events collected. Run 2 at ATLAS recorded approximately  $140 \text{ fb}^{-1}$  of proton-proton collision data, with 90% certified as good for physics analysis [24].

## 2.2 ATLAS Detector

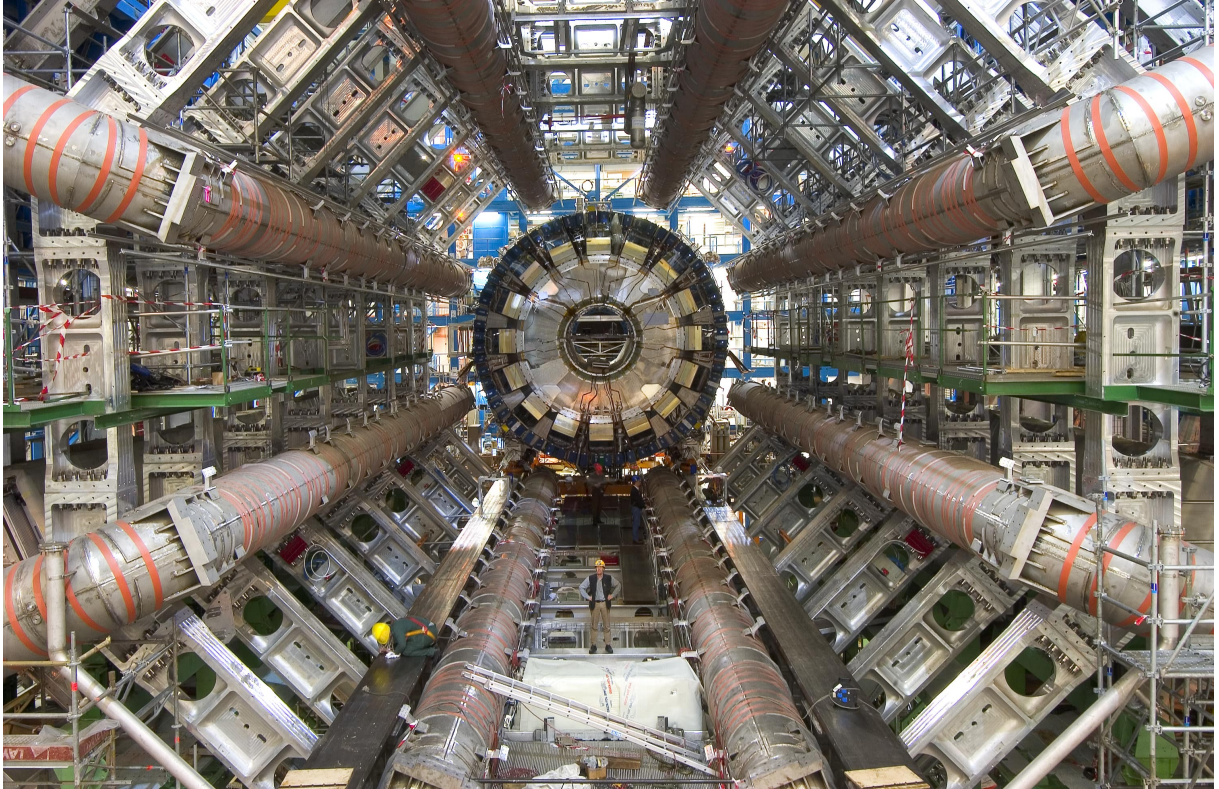


Figure 2.2: ATLAS detector under construction.

The ATLAS detector (ATLAS humorously standing for A Toroidal LHC ApparatuS) is the largest particle detector in the world (see Figure 2.2). Weighing over 7000 tons and sitting under 100 meters of Swiss soil, the ATLAS detector records millions of events per second of valuable experimental physics data [25].

ATLAS is built as a series of concentric “onion” layers around the collision point, each optimized to record a different kind of signature:

- **Inner detector:** very fine silicon sensors that trace charged-particle trajectories
- **Calorimeters:** absorb and measure the energy of electrons, photons and hadronic jets
- **Muon system:** large air-core magnets and tracking chambers that catch muons escaping the calorimeters
- **Trigger & readout:** real-time electronics that select  $\sim 1000$  out of 40 million collisions per second for storage

Together these subsystems deliver a high-dimensional snapshot of each collision, energy deposit, and timing information that we can use to create a robust dataset to probe through the physics.

# Chapter 3

## Theoretical Framework

HEP analysis, especially when combined with machine learning, requires a solid theoretical understanding before practical implementation. It is important to understand the type of data being analyzed, the rationale behind the data selection, and the methodological options available, including their strengths and drawbacks.

Given the difficulty in differentiating the  $t\bar{t}H$  signal from substantial background noise, this thesis will leverage augmented datasets and explore a variety of machine learning models. The technical focus lies on experimenting with diverse transformer architectures, including adjustments to layers, embeddings, normalization techniques, and hyperparameters, to determine the optimal configuration for robust signal extraction. This theoretical framework not only delineates the physics aspects of the  $t\bar{t}H$  process but also supports our innovative approach to model training and performance evaluation.

In this chapter, we dissect the  $2\ell_{\text{SS}} + \tau_{\text{had}}$  channel, establish the dataset foundation that underpins our  $t\bar{t}H$  analysis, and establish important ML metrics that are commonly utilized in HEP analysis.

### 3.1 The $2\ell_{\text{SS}} + \tau_{\text{had}}$ Channel

One promising channel for  $t\bar{t}H$  signal extraction is the same-sign dilepton ( $2\ell_{\text{SS}}$ ) plus a hadronically decaying tau ( $\tau_{\text{had}}$ ) channel.

- **$2\ell_{\text{SS}}$ :** This part of the channel definition specifies that the final state must contain two leptons, either electrons, muons, or a combination, that have the same electric charge.
- **$\tau_{\text{had}}$ :** The second part of the channel definition sets the requirement for one hadronically decaying tau lepton.

Hadronically, in this context, means that the lepton undergoes a weak force decay into one or more hadrons, a hadron being a composite particle made from multiple quarks held together by the strong force. Common hadronic decay channels for the tau lepton include the neutral pion ( $\pi^0$ ) and its charged variants ( $\pi^-$  and  $\pi^+$ ). The most common decay channels for the tau lepton are listed in Table 3.1.

Decay mode	Branching ratio [%]	Hadronic
$\tau^- \rightarrow e^- \bar{\nu}_e \nu_\tau$	17.8	No
$\tau^- \rightarrow \mu^- \bar{\nu}_\mu \nu_\tau$	17.4	No
$\tau^- \rightarrow \pi^- \nu_\tau$	10.8	Yes
$\tau^- \rightarrow \pi^- \pi^0 \nu_\tau$	25.5	Yes
$\tau^- \rightarrow \pi^- 2\pi^0 \nu_\tau$	9.3	Yes
$\tau^- \rightarrow \pi^- \pi^+ \pi^- \nu_\tau$	9.8	Yes

Table 3.1: Primary  $\tau$  lepton decay channels at  $m_\tau \simeq 1.777$  GeV, with branching ratios and an indication of hadronic modes [26].

Pions then decay into muons and neutrinos (for charged pions, e.g.  $\pi^- \rightarrow \mu^- \bar{\nu}_\mu$ ) or into photon pairs (for neutral pions,  $\pi^0 \rightarrow \gamma\gamma$ ), which are the actual signatures reconstructed in the detector. Because the tau initially decays into quarks, that almost instantly form hadrons, we can illustrate the parton-level decay in Table 3.2, and a representative Feynman diagram for the  $2\ell_{\text{SS}} + \tau_{\text{had}}$  channel is shown in Figure 3.1.

Decay mode	Partial width [MeV]	Branching ratio [%]
$\tau^- \rightarrow d \bar{u} \nu_\tau$	$1.26 \times 10^{-9}$	55.4
$\tau^- \rightarrow s \bar{u} \nu_\tau$	$6.7 \times 10^{-11}$	2.9

Table 3.2: Parton-level partial widths and branching ratios for the primary hadronic  $\tau^-$  decays into quark-antiquark pairs plus a neutrino [26].

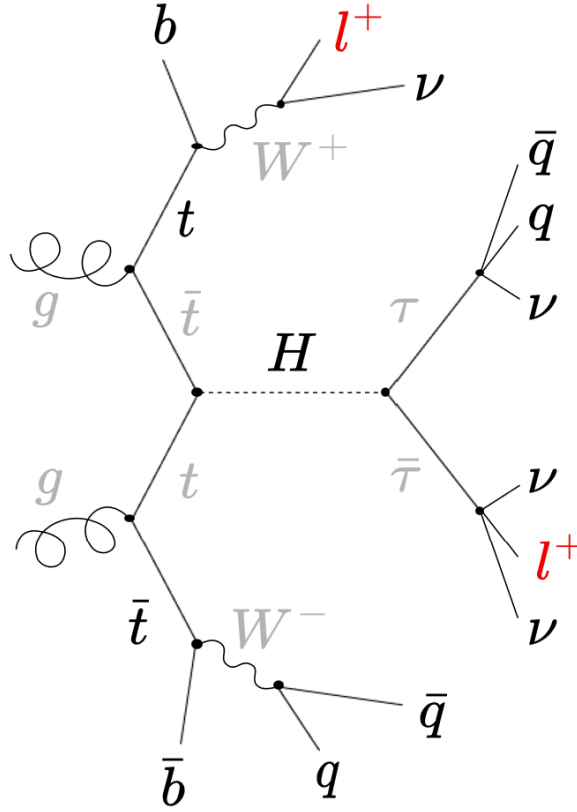


Figure 3.1: Feynman diagram of gluon-gluon fusion creating the  $t\bar{t}H$  signal and subsequent decay into two light leptons and a hadronically decaying tau.

The two same-sign leptons and hadronically decaying tau we are looking for are highlighted in Figure 3.1. It also illustrates the parton-level decay

$$\tau^- \rightarrow W^{*-} \rightarrow q\bar{q}' + \nu_\tau, \quad (3.1)$$

with  $q\bar{q}' = d\bar{u}$  or  $s\bar{u}$  prior to hadronization into pions.

The  $2\ell_{\text{SS}} + \tau_{\text{had}}$  channel offers a particularly clean signature for  $t\bar{t}H$ . The same-sign dilepton signature effectively suppresses common opposite-charge dilepton backgrounds, and the  $\tau_{\text{had}}$  signature appears as a narrow jet with low track multiplicity, making it relatively easy to distinguish from QCD backgrounds.

## 3.2 Signal Region

To compile our dataset and maintain a consistent framework for both testing and validation, we define a dedicated Signal Region (SR) optimized for  $t\bar{t}H$  signal extraction. All event selection, performance evaluations, and model training are carried out within this SR. The full expression used to create our dataset is

```

Leptons_NOSYS.size() = 2
    & &
(Leptons_NOSYS[0].charge × Leptons_NOSYS[1].charge) = 1
    & &
Taus_NOSYS.size() = 1
    & &
nbJets77_NOSYS > 0
    & &
Jets_NOSYS.size() > 3.

```

The SR cut expression selects for the number of leptons to be equal to two, the charge of those leptons to be the same, the number of taus to be one, at least one jet composed of a b-quark, and the total number of jets to be greater than three.

### 3.3 Monte Carlo Simulation

Accurate Monte Carlo simulations, from now on to be referred to as MCS, in ATLAS depend critically on detailed measurements of the detector’s response. Collision data is used to improve and validate the GEANT4-based detector model [27]. By comparing kinematic and topological distributions in well-understood control regions (e.g.  $Z \rightarrow \ell\ell$ ,  $W \rightarrow \ell\nu$ ) between data and MCS, the modeling of particle interactions is iteratively tuned. This feedback loop—using real detector data to refine the simulation—ensures that both signal and background processes are reproduced with high fidelity, thereby reducing systematic uncertainties in physics measurements.

MCS provide the simulated collision events used to train and validate our machine-learning models. In ATLAS, MCS data are produced by:

- **Event generation:** Hard-scatter processes (e.g.  $t\bar{t}H$ , backgrounds) are generated with matrix-element tools and showered into stable particles.
- **Detector response:** A GEANT4-based model emulates the passage of particles through the ATLAS detector, producing digitized signals.
- **Reconstruction & truth labeling:** Standard algorithms build physics objects (jets, leptons, etc.) and match them to their “true” generator origins, yielding the



ground-truth labels needed for supervised learning [28].

MC samples are validated in well-understood control regions to ensure realistic detector performance. The resulting labeled datasets form the foundation of our transformer-based  $t\bar{t}H$  signal versus background classifiers.

## 3.4 Foundations for HEP Classification

This section introduces the key concepts and tools used to assess the performance of binary classifiers in HEP, with an emphasis on metrics that capture both signal sensitivity and background rejection. The following methods and strategies are used in this thesis and are shown in more detail in Chapter 5 where the model performance discussion takes place.

### 3.4.1 Binary Classification Metrics

We quantify classifier performance using ROC curves and their area under the curve (AUC) to gauge overall discrimination power. The Receiver Operating Characteristic (ROC) curve plots the True Positive Rate (TPR) against the False Positive Rate (FPR) at various classification thresholds, providing a comprehensive view of a classifier's performance.

The True Positive Rate and False Positive Rate are defined as:

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \quad \text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \quad (3.2)$$

where TP, FP, FN, and TN stand for true positives, false positives, false negatives, and true negatives, respectively. The AUC summarizes the ROC curve into a single value between 0 and 1. An AUC of 0.5 corresponds to random guessing, while an AUC of 1.0 indicates perfect classification. ROC curves are one of the main methods of visualizing model performance and will be used in the later sections.

### 3.4.2 Cross-Validation

Data are split into training and validation sets to ensure unbiased performance estimates. K-fold cross-validation is employed to maximize data usage and stabilize metric evaluations. In this thesis, we use five-fold cross-validation, meaning our data loader will shuffle and split the data into five equal partitions:

$$\{D_1, D_2, D_3, D_4, D_5\} \quad \text{partition of the full dataset } D.$$

We then train five separate models, each time using a new dataset partition as the validation set.

### 3.4.3 Confusion Matrix

The confusion matrix compiles true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN) to provide a complete view of classification outcomes. Visualizing this matrix helps select operating thresholds and understand error composition.

### 3.4.4 Efficiency & Background Rejection

Signal efficiency measures the fraction of true signal kept, while background rejection quantifies background suppression. Working points on the ROC curve balance these two aims.

$$\varepsilon_S = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (3.3)$$

$$1 - \varepsilon_B = 1 - \frac{\text{FP}}{\text{FP} + \text{TN}} \quad (3.4)$$

### 3.4.5 F<sub>1</sub>-Score

The F<sub>1</sub>-score provides a single-value metric sensitive to class imbalance. It complements AUC by penalizing extreme trade-offs between purity and efficiency.

$$F_1 = 2 \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \quad (3.5)$$

### 3.4.6 Overtraining & Validation Strategies

The more complex models in this thesis use learning curves and early stopping to guard against overfitting by monitoring training and validation performance. They also use techniques like dropout and weight decay to serve as regularization and careful tracking of metrics to help the model generalize well.

# Chapter 4

## Software Infrastructure

Advancements in software frameworks and infrastructure have allowed for fast iterative development, smarter workflows, and more complex analysis. A common programming language used extensively in ML and analysis is Python.

### 4.1 Python

Python<sup>1</sup> has become the go-to language for many ML researchers and practitioners. Its ease of use, world-class support, and vibrant community have propelled it to where it is today in a relatively short period of time. The Python ecosystem includes thousands of community developed frameworks and libraries that extend its capabilities to do virtually anything with high performance.

In this thesis, we leverage community-driven frameworks and libraries such as NumPy, Pandas, and PyTorch along with many others listed in Appendix A.4. We use them with Python 3.9.21 in a virtual environment (VENV) we have created on the LXPLUS machine at CERN.

### 4.2 LXPLUS

All data storage and computational work for this thesis has been done on the LXPLUS<sup>2</sup> machine thanks to CERN's generous user policies. LXPLUS has provided over 100 hours of GPU time as well as dozens more of CPU time.

---

<sup>1</sup><https://www.python.org/>

<sup>2</sup><https://linux-training.web.cern.ch/lxplus/introduction/>

## 4.3 At the Root of it All

### 4.3.1 ROOT

HEP analysis requires large amounts of data and real-time access to all of it. To mend this pain point, CERN developed the ROOT<sup>3</sup> framework to be the backbone of its data processing needs. Written in C++, ROOT offers a variety of tools and features to help researchers analyze and present data. It is extensively used at CERN and provides the infrastructure needed to process the data we are working with in this thesis.

### 4.3.2 Uproot

Because ROOT is written in C++ and is only packaged with a command line interface, we use a Python wrapper, Uproot, to read and process its files. Uproot<sup>4</sup> is used to handle and analyze every ROOT file in this thesis. Subsequent analysis uses NPY or CSV files for convenience and efficiency.

## 4.4 Machine Learning

### 4.4.1 PyTorch

PyTorch<sup>5</sup> is the workhorse of this thesis. It acts as a full, well-maintained, and polished tool belt on the hip of ML researchers. We utilize dozens of methods and functions from PyTorch, such as the various dataloaders it provides (for both conventional and K-Fold dataset creation), tensor operation functions, statistical methods, and of course, the NN classes we use to create the various transformers analyzed.

Specifically, this thesis builds and trains all models using PyTorch’s NN application programming interface (API). Each model inherits from `torch.nn.Module`, and common building blocks such as `nn.LayerNorm` for normalization, `nn.MultiheadAttention` for self-attention, and `nn.Linear` for affine projections are used extensively throughout Chapter 5.

### 4.4.2 Optuna

Optuna<sup>6</sup> is a state-of-the-art hyperparameter optimization framework that provides efficient sampling algorithms and pruning strategies to automate the search for optimal model parameters. A hyperparameter is a configuration variable set before training a ML

---

<sup>3</sup><https://root.cern/about/>

<sup>4</sup><https://uproot.readthedocs.io/en/latest/index.html>

<sup>5</sup><https://pytorch.org/>

<sup>6</sup><https://optuna.org/>

model that is not learned from the data. In the proposed model, Optuna was used to tune critical hyperparameters such as learning rate, warm-up epochs, dropout rates, and scheduler settings during training.

## 4.5 TRExFitter

HEP requires powerful statistical frameworks to interpret complex data. One such tool is TRExFitter<sup>7</sup> (Top Related Experiment Fitter), which is widely used within the community. Being built into the LXPLUS machines and based on the ROOT framework, TRExFitter provides efficient statistical modeling, fitting, and the generation of publication-quality plots and tables used at CERN. TRExFitter is instrumental to this thesis. Some plots, such as Figure 5.17, Figure 7.3a, and Figure 7.3b are created by TRExFitter.

We use TRExFitter to turn our ntuples (our event data in the .root format) into histograms that we then normalize, perform fits on, and plot.

---

<sup>7</sup><https://gitlab.cern.ch/TRExStats/TRExFitter>

# Chapter 5

## Experimental Setup & Baseline Model

We start our analysis with a baseline model. A relatively simple attempt at classification that is meant to point us in the right direction and gain insight. We train the same model on two separate datasets, differentiated by the normalization techniques used to create them.

### 5.1 Data Splits & Cross-Validation

As mentioned in Subsection 3.4.2, we use cross-validation to get an accurate and unbiased look at our models' performance. For every training run, we produce five separately trained models, each on a different fold.

This, by extension, also means we are running an 80/20 split where we use 80% of our dataset for training and 20% for validation. This is a common setup in ML and has been proven to balance dataset size and prediction confidence.

### 5.2 Feature & Dataset

From our robust signal region cut expression defined in Section 3.2, we create a feature set based on particle kinematic data. Our dataset consists of 160,194 events, 36.61% of which are our  $t\bar{t}H$  signal (58,644) and the rest, the background (101,550). Below are the features used in this analysis.

Features
DeltaR_min_lep_jet_fwd_NOSYS
HT_NOSYS
MLepMet_NOSYS
MtLepMet_NOSYS
dEta_maxMjj_frwdjet_NOSYS
jets_pt_0_NOSYS
lep_Z0SinTheta_1_NOSYS
minDeltaR_LJ_1_NOSYS
sumPsbtag_NOSYS
taus_RNNJetScoreSigTrans_0_NOSYS
taus_pt_0_NOSYS

Table 5.1: Input features used in the analysis.

- **DeltaR\_min\_lep\_jet\_fwd\_NOSYS**: The minimum angular separation  $\Delta R$  between the selected lepton and the leading jet in the event.
- **HT\_NOSYS**: The scalar sum of the transverse momenta of all selected jets in the event,  $H_T = \sum_{\text{jets}} p_T$ ,
- **MLepMet\_NOSYS**: The invariant mass of the system formed by the selected lepton and the missing transverse momentum,  $M(\ell, \text{MET})$ .
- **MtLepMet\_NOSYS**: The transverse mass of the selected lepton and the missing transverse momentum,  $M_T(\ell, \text{MET}) = \sqrt{2p_T^\ell \text{MET} (1 - \cos \Delta\phi)}$ .
- **dEta\_maxMjj\_frwdjet\_NOSYS**: The pseudorapidity difference  $\Delta\eta$  between the two forward jets that form the dijet pair with the maximum invariant mass.
- **jets\_pt\_0\_NOSYS**: The transverse momentum  $p_T$  of the leading (highest- $p_T$ ) jet in the event.
- **lep\_Z0SinTheta\_1\_NOSYS**: The product of the longitudinal impact parameter  $z_0$  and  $\sin \theta$  for the second-selected lepton.
- **minDeltaR\_LJ\_1\_NOSYS**: The minimum angular separation  $\Delta R$  between the second-selected lepton and any jet in the event.
- **sumPsbtag\_NOSYS**: The sum of the per-jet  $b$ -tagging discriminant scores from the chosen algorithm.

- `taus_RNNJetScoreSigTrans_0_NOSYS`: The transformed RNN-based  $\tau$ -jet discriminant score for the leading  $\tau$  candidate.
- `taus_pt_0_NOSYS`: The transverse momentum  $p_T$  of the leading  $\tau$  candidate in the event.

All features were computed without systematic uncertainties.

## 5.3 Normalization

Normalization is an extremely important part of transformer model analysis. We will test two dataset normalization schemes, one where every feature is normalized by the z-score technique and one where each feature gets a normalization technique that is best for its specific distribution.

### 5.3.1 Z-score Standardization

A popular and effective normalization technique that we use in this thesis is Z-scoring. We compute the dataset-wide mean and standard deviation for each feature we are utilizing using

$$\mu = \frac{1}{N} \sum_{i=1}^N x_i \quad (5.1)$$

and

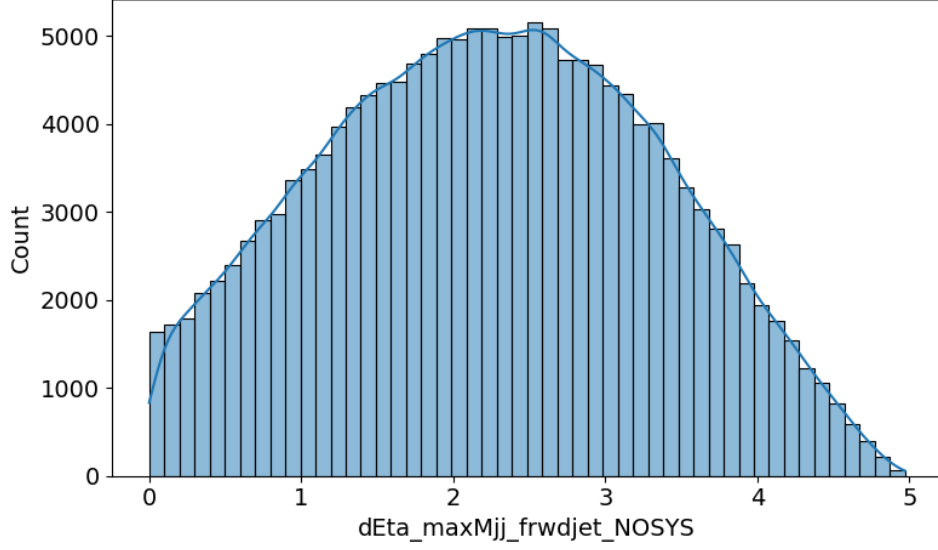
$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^N (x_i - \mu)^2}. \quad (5.2)$$

Then, we use these new variables' specific values to calculate the new value of each feature in every event:

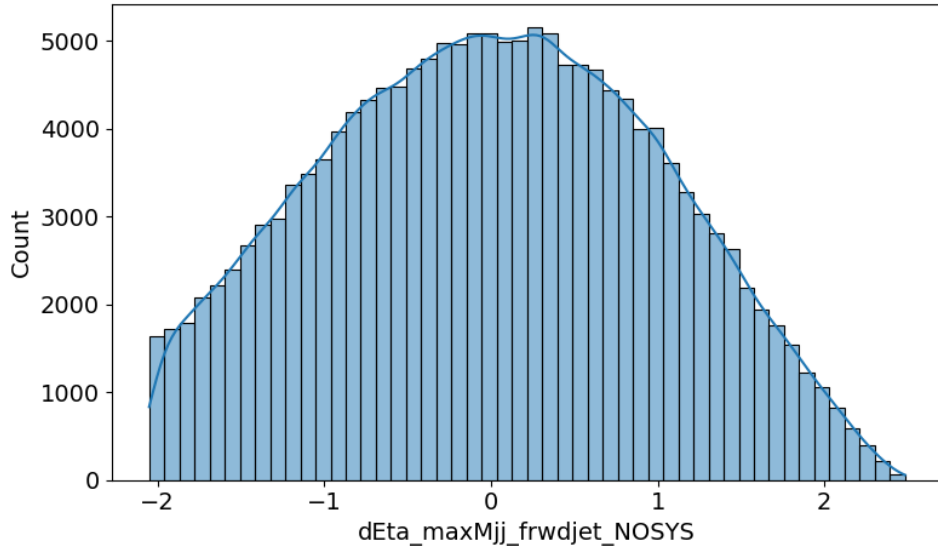
$$x'_i = \frac{x_i - \mu}{\sigma}, \quad i = 1, 2, \dots, N. \quad (5.3)$$

We apply this technique to every feature in our feature set, but it favors some distributions over others. An example of an ideal distribution that benefits the most from Z-scoring is:



Figure 5.1: Prenormalized `dEta_maxMjj_frwdjet_NOSYS`.

The distribution for `dEta_maxMjj_frwdjet_NOSYS` in Figure 5.1 is an excellent candidate for z-score normalization. After processing, our new distribution for `dEta_maxMjj_frwdjet_NOSYS` is

Figure 5.2: `dEta_maxMjj_frwdjet_NOSYS`, normalized via Z-scoring.

The shape of the distribution in Figure 5.2 has not changed at all, preserving detail in the original dataset, but the mean has been shifted to zero. This new distribution is ideal for our transformer models and is ready to be used for our training and validation loops.

We apply Z-scoring to every feature, creating our z-score standardization map.

Feature	Normalization
DeltaR_min_lep_jet_fwd_NOSYS	Z-score
HT_NOSYS	Z-score
MLepMet_NOSYS	Z-score
MtLepMet_NOSYS	Z-score
dEta_maxMjj_frwdjet_NOSYS	Z-score
jets_pt_0_NOSYS	Z-score
lep_Z0SinTheta_1_NOSYS	Z-score
minDeltaR_LJ_1_NOSYS	Z-score
sumPshtag_NOSYS	Z-score
taus_RNNJetScoreSigTrans_0_NOSYS	Z-score
taus_pt_0_NOSYS	Z-score

Table 5.2: Normalization method applied to each input feature in the z-score standardized dataset.

### 5.3.2 Feature-wise Scaling

As mentioned in Subsection 5.3.1, not every distribution is an ideal candidate for Z-scoring. The distribution of `taus_pt_0_NOSYS` is a prime example.

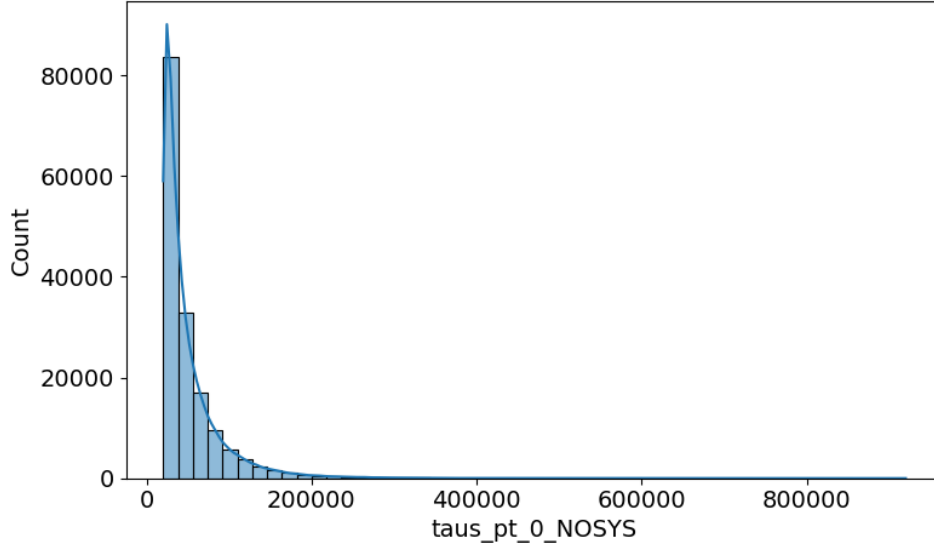
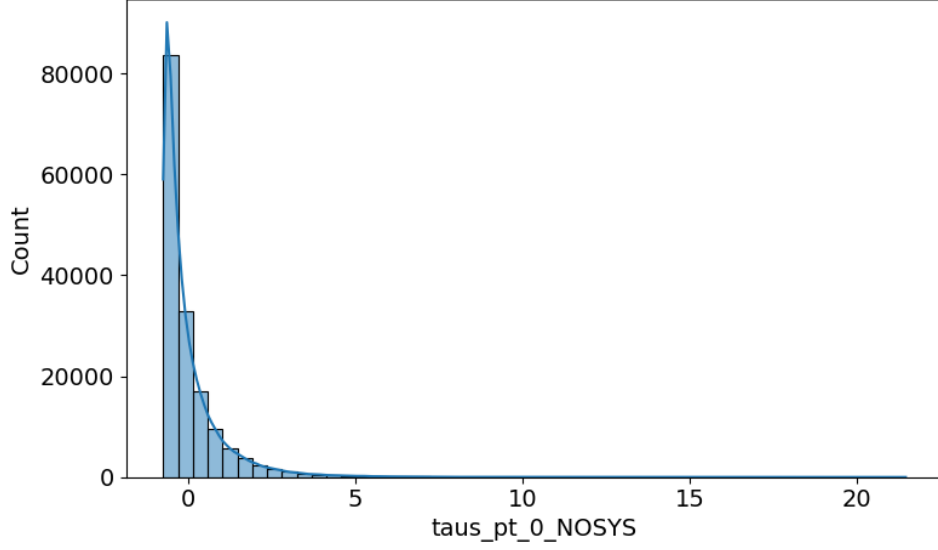


Figure 5.3: Prenormalized `taus_pt_0_NOSYS` (in MeV).

In Figure 5.3, we see an extremely skewed distribution that would fare poorly in the training data of a transformer-based analysis. After applying Z-scoring, Figure 5.4 does not show us much of an improvement.

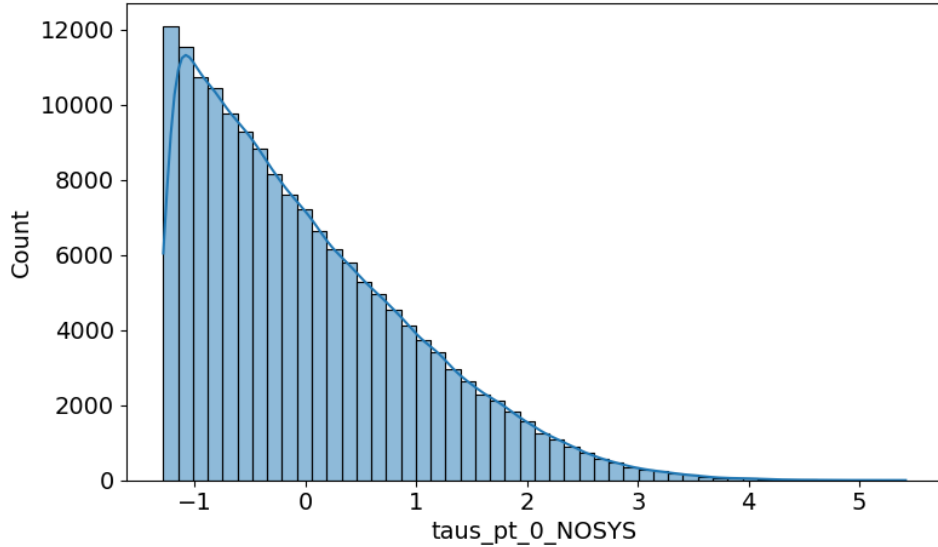
Figure 5.4: `taus_pt_0_NOSYS`, normalized via Z-scoring.

This brings us to the log function. Instead of using  $x_i$  as we did in Equation 5.3.1 to yield  $x'_i$ , we do

$$x'_i = \frac{\log(x_i) - \mu}{\sigma}, \quad i = 1, 2, \dots, N \quad (5.4)$$

for values we want to normalize with log + z-score. We use log + z-score for most of the skewed distributions.

After applying log + z-score, we obtain the distribution in Figure 5.5.

Figure 5.5: `taus_pt_0_NOSYS`, normalized via log + Z-scoring.

This form is much nicer for our transformer model, but it loses some intrinsic information about the outliers.

We apply a similar methodology to the other ten features, creating a normalization map for our feature-wise scaling, as listed in Table 5.3.

Feature	Normalization
DeltaR_min_lep_jet_fwd_NOSYS	Z-score
HT_NOSYS	Log + Z-score
MLepMet_NOSYS	Log + Z-score
MtLepMet_NOSYS	Log + Z-score
dEta_maxMjj_frwdjet_NOSYS	Z-score
jets_pt_0_NOSYS	Log + Z-score
lep_Z0SinTheta_1_NOSYS	Z-score
minDeltaR_LJ_1_NOSYS	Z-score
sumPshtag_NOSYS	Z-score
taus_RNNJetScoreSigTrans_0_NOSYS	Z-score
taus_pt_0_NOSYS	Log + Z-score

Table 5.3: Normalization method applied to each input feature in the feature-wise scaled dataset.

### 5.3.3 Correlation Heatmap

This brings us to another avenue we can explore to analyze our feature set, the correlation heatmap. We generated a heatmap of the Pearson correlation matrix across our 11 features. This allows a rapid visual assessment of pairwise linear relationships, highlighting both potential redundancies and inverse associations. In simpler terms, when we see a high positive correlation ( $r > 0.7$ ) between two features, that means as  $X_i$  increases,  $X_j$  also increases in a roughly linear fashion. A large negative value ( $r < -0.7$ ) also indicates a linear correlation, just in this case, as one feature increases, the other decreases.

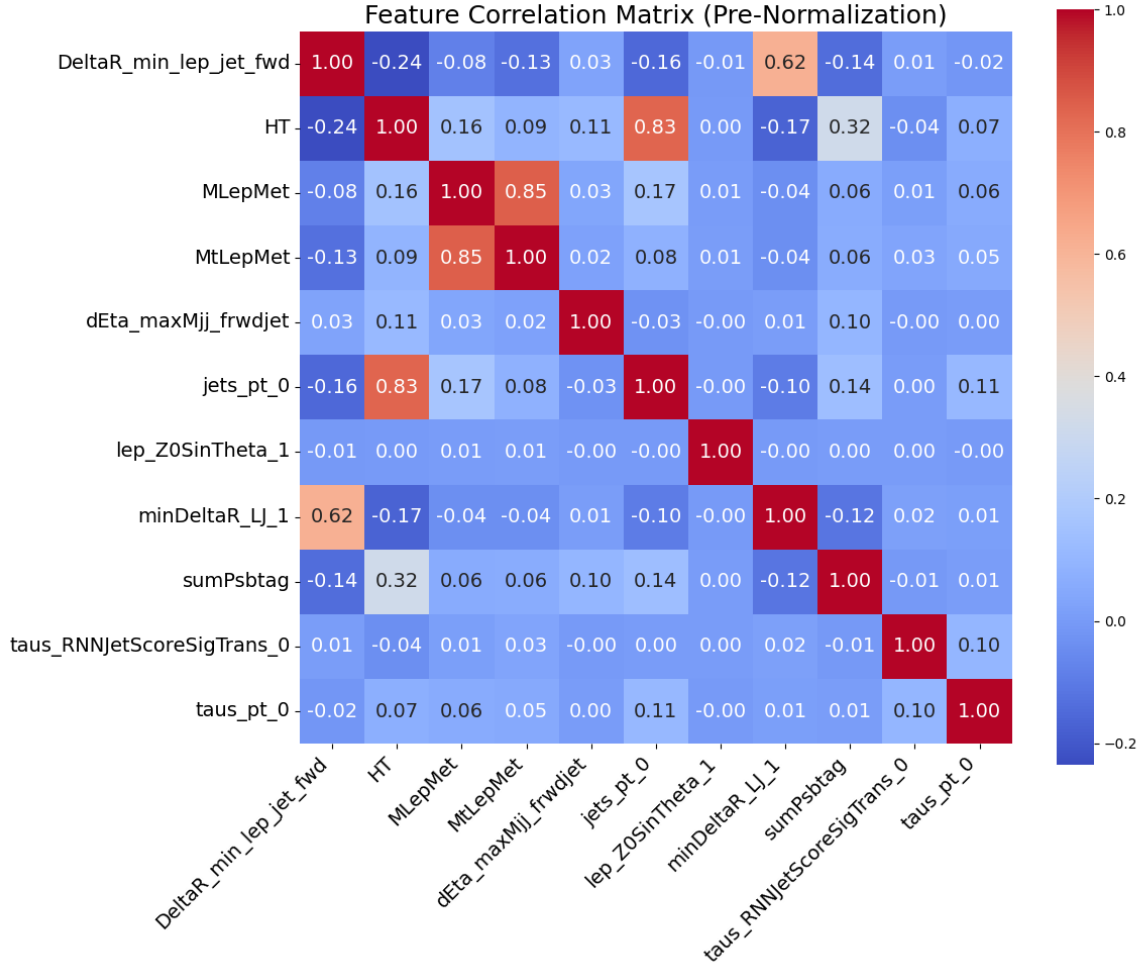


Figure 5.6: Correlation heatmap of the pre-normalized feature set, with the “\_NOSYS” tag removed from each feature for readability.

The pairwise correlation in Figure 5.6 shows that most features have a zero or a near-zero correlation, with the exception of some of our **Met** and **jet** features (**jets\_pt\_0** versus **HT**) which share an  $r \approx 0.83$  and the two **LepMet** features, which share an  $r \approx 0.85$ . This is overall a suitable feature set with a small number of features being strongly correlated.

In our z-score standardized dataset, the pre- and post-normalization heatmaps are exactly the same since we are using the z-score normalization on everything. In the feature-wise scaling dataset, the logarithmic nature of our transformations alters the correlations between features, resulting in the correlation profile shown in Figure 5.7

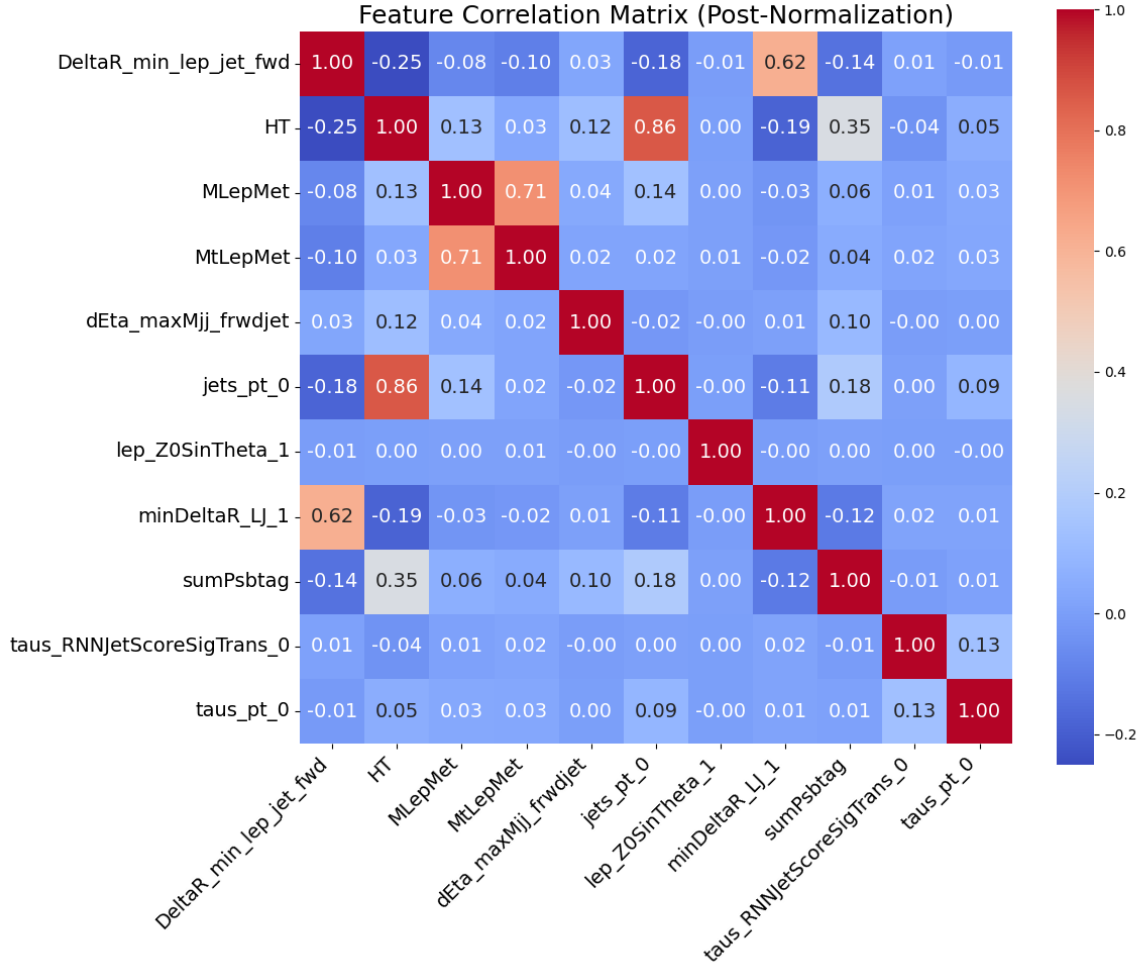


Figure 5.7: Correlation heatmap of the feature set, post feature-wise scaling, with the “\_NOSYS” tag removed from each feature for readability.

One of the most notable differences observable in Figure 5.7 is the contrast between the two *LepMet* features. Since both of these distributions are highly skewed and require log normalization, we reduced the correlation coefficient by approximately 0.14. We see how impactful this is when we compare model performance between the two datasets.

## 5.4 Baseline Transformer Implementation

### 5.4.1 Sequence

Fitting transformer models for tabular data requires more careful consideration than traditional approaches. Transformers require the input in a sequence. A suitable approach, popularized by the *TabTransformer*<sup>1</sup> paper, is to treat each feature as a token in the input sequence, so for our set of 11 features, our sequence length would be 11. Let  $L$  be the

<sup>1</sup><https://arxiv.org/abs/2012.06678>

sequence length, then we can formally define our sequence as

$$\mathbf{x} = [x_1, x_2, \dots, x_L]^\top \in \mathbb{R}^L. \quad (5.5)$$

We then cast the sequence into a higher-dimensional representation using a linear layer of artificial neurons, which naturally use a learnable weight ( $\mathbf{W} \in \mathbb{R}^{d_{model} \times 1}$ ) and bias ( $\mathbf{b} \in \mathbb{R}^{d_{model}}$ ). We choose  $d_{model} = 64$  for the initial transformer because it provides sufficient representational capacity and is cleanly divisible by common attention head counts. The input vector is cast into shape  $(L, d_{model})$ . From this point on, we refer to that tensor as the hidden-state matrix

$$\mathbf{e}_i = \mathbf{W} x_i + \mathbf{b} \in \mathbb{R}^{d_{model}} \quad (i = 1, \dots, L). \quad (5.6)$$

The value of  $d_{model}$  is one of the most important hyperparameters, and it will follow our hidden-state matrix throughout the model.

This now leaves us with a hidden-state matrix prime for additional enhancements such as positional encoding

$$\mathbf{z}_i = \mathbf{e}_i + \mathbf{p}_i. \quad (5.7)$$

As described in Subsection 1.2.5, we skip any kind of positional encoding and are left with

$$\mathbf{Z} = \begin{bmatrix} \mathbf{z}_1^\top \\ \mathbf{z}_2^\top \\ \vdots \\ \mathbf{z}_n^\top \end{bmatrix} \in \mathbb{R}^{L \times d_{model}}. \quad (5.8)$$

The hidden-state matrix is now ready to be processed by the encoder blocks that follow the input projection layer.

### 5.4.2 Transformer Encoder

The repeating encoder of our transformer is relatively simple in the baseline model. We use four encoder blocks in the model. The encoder consists of

- Normalization layer,
- Multi-head self-attention layer, where the embedding dimension is  $d_{model}$  and number of heads is eight ( $h = 8$ ),

- Residual connection, where the input vector to the encoder is summed with the output vector of the attention layer.

The attention mechanism works from the formula defined in Subsection 1.2.6,

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V, \quad (5.9)$$

but now we implement the multi-head attention version, defined as

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V), \quad i = 1, \dots, h, \quad (5.10)$$

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h) W^O, \quad (5.11)$$

where  $W^O$  is an extra learnable weight matrix that is used only in the multi-head version. The normalization layer is implemented exactly as specified in Subsection 1.2.9.

### 5.4.3 Output

After the encoder stack produces a sequence of hidden states

$$X = [x_1, x_2, \dots, x_L] \in \mathbb{R}^{L \times d_{\text{model}}}, \quad x_i \in \mathbb{R}^{d_{\text{model}}}, \quad (5.12)$$

we flatten them into a single large vector of dimension  $L \times d_{\text{model}}$ , which we will call dimension  $N$ :

$$x_{\text{flat}} = \text{reshape}(X) = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} \in \mathbb{R}^N. \quad (5.13)$$

After this operation, we are left with one expanded vector that carries information about all other vectors in our sequence. We then extract model probabilities by applying a series of linear layers and activation functions.



$$\begin{aligned}
& \mathbf{x} \in \mathbb{R}^{d_{model}} \\
& \downarrow \\
& W_1, b_1 \\
& \downarrow \\
& \mathbf{h}_1 = \text{GELU}(W_1 \mathbf{x} + b_1) \in \mathbb{R}^{d_{model}/2} \\
& \downarrow \\
& W_2, b_2 \\
& \downarrow \\
& \mathbf{h}_2 = \text{GELU}(W_2 \mathbf{h}_1 + b_2) \in \mathbb{R}^{d_{model}/4} \\
& \downarrow \\
& W_3, b_3 \\
& \downarrow \\
& \mathbf{y} = W_3 \mathbf{h}_2 + b_3 \in \mathbb{R}^k
\end{aligned} \tag{5.14}$$

This casts the sequence down to a  $K$ -dimensional vector. These are our raw model outputs. We have  $K = 1$ , so this output layer produces a single scalar value. This final value is called a logit and is used for our binary cross-entropy loss function that is defined in Subsection 1.2.3. To extract a probability from the logit, we apply the sigmoid function and extract the probability.

#### 5.4.4 Weight Initialization

For our baseline weight initialization, we use PyTorch's default initialization for simplicity and safety. All weight matrices are initialized from a uniform distribution, and all biases are set to zero:

- **All biases:** set to zero.
- **Linear layers:** weights uniformly in  $[-1/\sqrt{n}, 1/\sqrt{n}]$ , with  $n = \text{input features for the layer}$ .
- **Multi-Head Attention (Q, K, V, W):** weights uniformly in  $[-\sqrt{6/(2n)}, \sqrt{6/(2n)}]$  with  $n = d_{model}$ .
- **Normalization layers:**  $y = \gamma x + \beta$  where  $\gamma = 1$ ,  $\beta = 0$ .

### 5.5 Baseline Performance

We have many tools available at our disposal to poke and prod at our model. ROC curves, precision, recall, and F<sub>1</sub>-scores, loss and validation graphs, activation histograms,

attention heat maps, and many others.

In this section, we evaluate the performance of our two baseline models, both identical, but training on different datasets to gain insight into the best normalization techniques for HEP ML. We begin with analyzing the model trained on the feature-wise scaled dataset.

### 5.5.1 Performance Metrics

We will use the ROC curve as the main judge of performance, supported by metrics such as the  $F_1$ -score and data derived from the confusion matrix. We choose the ROC curve as our primary performance indicator because it evaluates true-positive versus false-positive rates across thresholds and is less sensitive to class imbalance.

After cross-validation using 5 folds, we saw very little variance in model performance ( $0.711 \leq \text{ROC score} \leq 0.723$ ) so we analyze the best fold, illustrated in Figure 5.8.

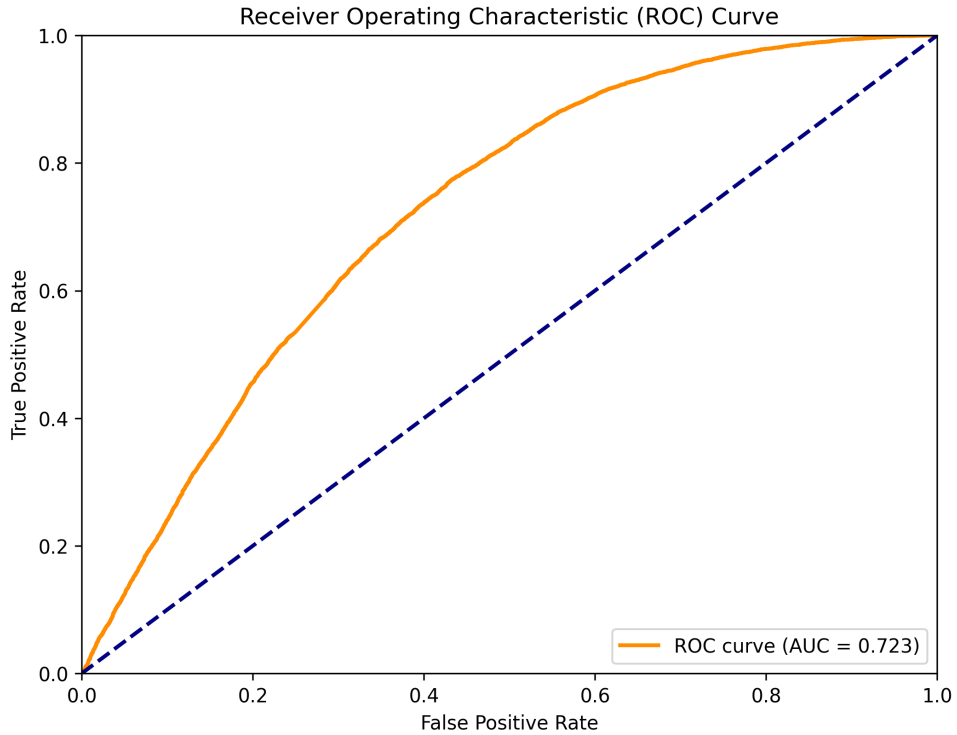


Figure 5.8: ROC curve of the baseline model trained on the feature-wise scaled dataset.

An ROC score of 0.723 is considered *moderate* (0.5 corresponds to random guessing). This is a welcome sight for an initial baseline model. The dashed diagonal line in the ROC curve represents the performance of a random classifier. We compile more statistics from validation runs to yield a classification report.

Class	Precision	Recall	F <sub>1</sub> -score	Support
background	0.7929	0.6589	0.7197	20 381
signal	0.5397	0.6992	0.6092	11 658
accuracy			0.6736	32 039
macro avg	0.6663	0.6790	0.6644	32 039
weighted avg	0.7008	0.6736	0.6795	32 039

Table 5.4: Classification report for background versus signal, baseline model trained on the feature-wise scaled dataset.

Table 5.4 summarizes the classification report for our baseline model on the *background* versus *signal* task. The model achieves an overall accuracy of 67.36%, with an averaged F<sub>1</sub>-score of 0.6644. It performs better on the majority *background* class (Precision 0.7929, Recall 0.6589, F<sub>1</sub>-score 0.7197) than on the minority *signal* class (Precision 0.5397, Recall 0.6992, F<sub>1</sub>-score 0.6092). This difference tells us that, while the model is cautious when predicting *background* (few false positives), it still misses a substantial fraction of true *background* events. It also captures most *signal* events but at the expense of a high false positive rate. From Table 5.4, we can conclude that the model displayed moderate performance with clear room for improvement.

Performance was never the goal of this model. For better insights into our data and feature set, we need to study the model in detail.

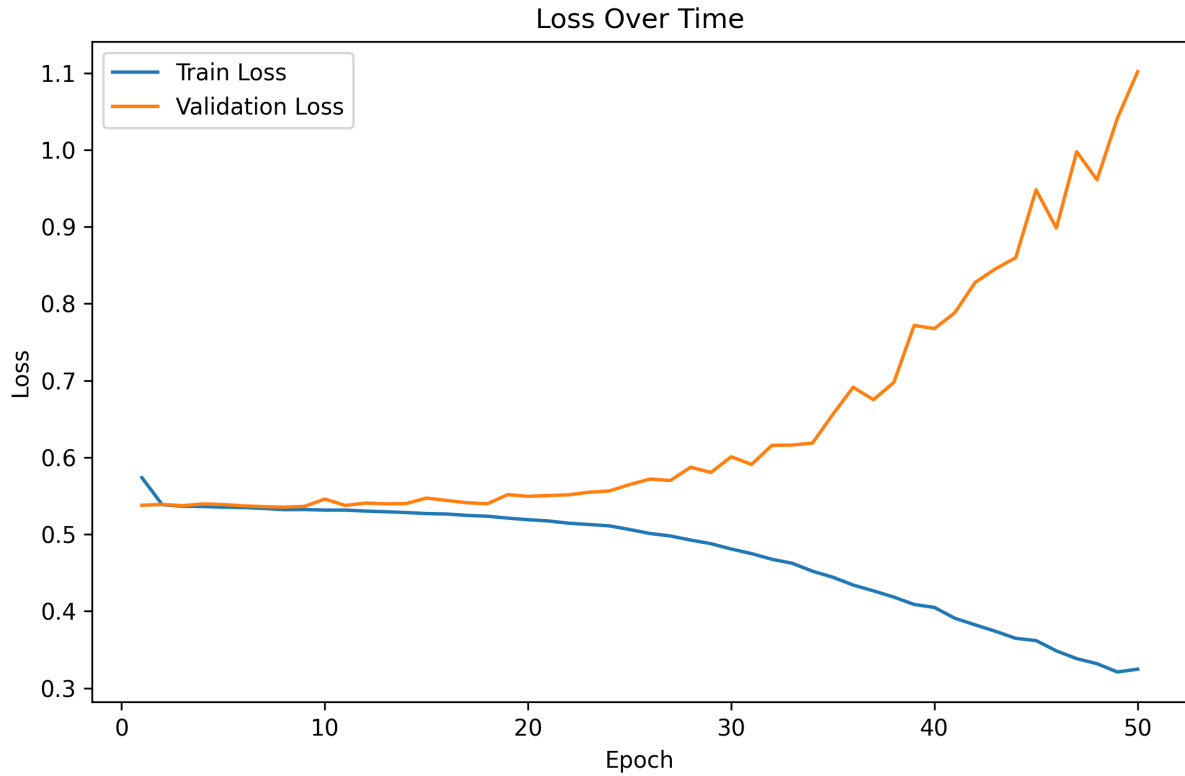


Figure 5.9: Training and validation loss over 50 epochs, baseline model.

From the loss-over-epochs graph illustrated in Figure 5.9, we can make a few important observations. The model quickly settles into a plateau after the start of training and finds it hard to make meaningful progress. It then finds correlation present only in the training data and exploits it, quickly dumping training loss as it progresses while equally adding onto the validation loss. This is a textbook example of overfitting, a problem present in every ML model, but more prevalent in transformer architectures because of the powerful attention mechanism. The model fails to generalize well, but the indication of the training loss going down rapidly is an indicator that the model has the capacity to learn and the attention mechanism is indeed working as we are expecting it to.

To gain further insight into our model, we can look into the activations of our input projection layer, attention layers, and encoder blocks.

### 5.5.2 Model Interpretability & Internal Dynamics

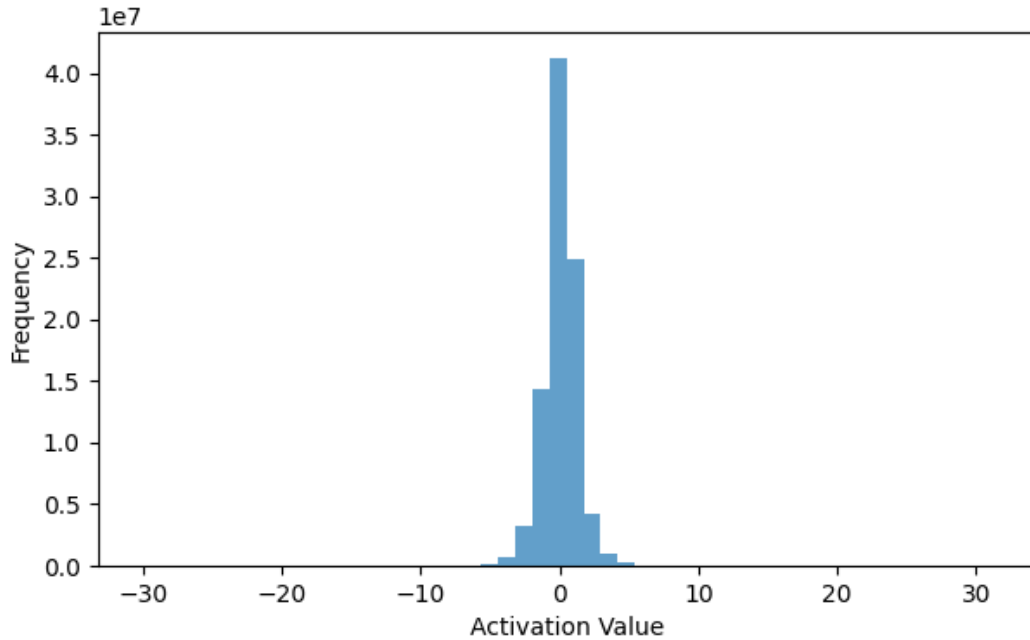


Figure 5.10: Activation values for the input projection layer after 50 epochs, baseline model.

Figure 5.10 shows another sign of overfitting. The range of our activations is extremely large, which is indicative of the model over-specializing to the training set rather than learning smooth, generalizable patterns. However, we also have an approximately symmetric distribution centered around zero, indicating that our normalization scheme successfully prevents a nonzero mean in the activations, helping to balance gradient updates.

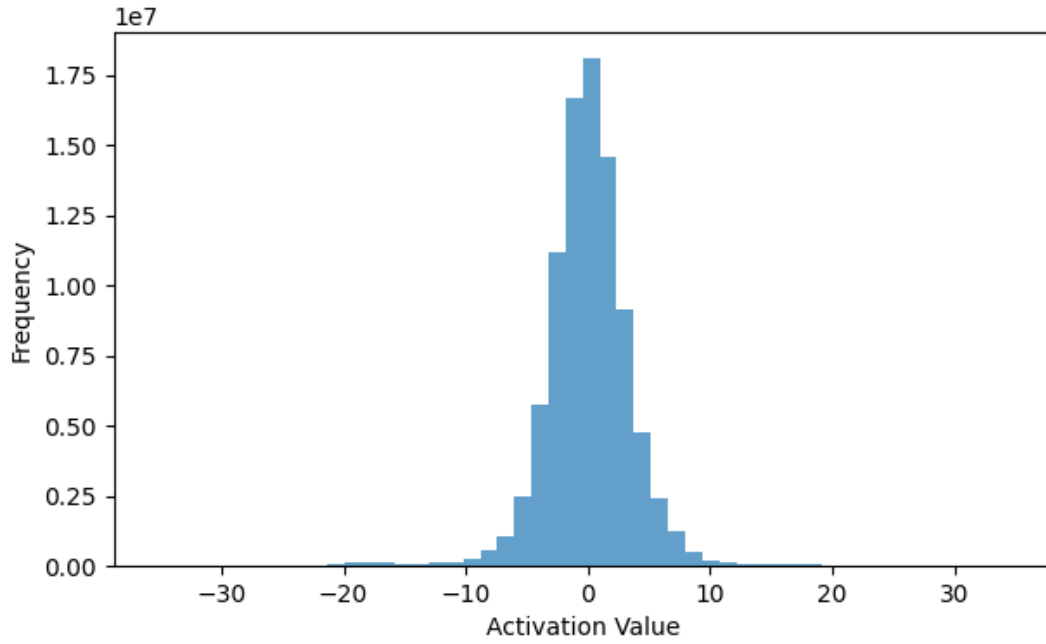


Figure 5.11: Activation values for the last encoder block after 50 epochs, baseline model.

The activation histogram for the last encoder block shows a similar situation but in an even more prominent way, meaning this pattern is consistent throughout the model. Next, we study the attention mechanism.

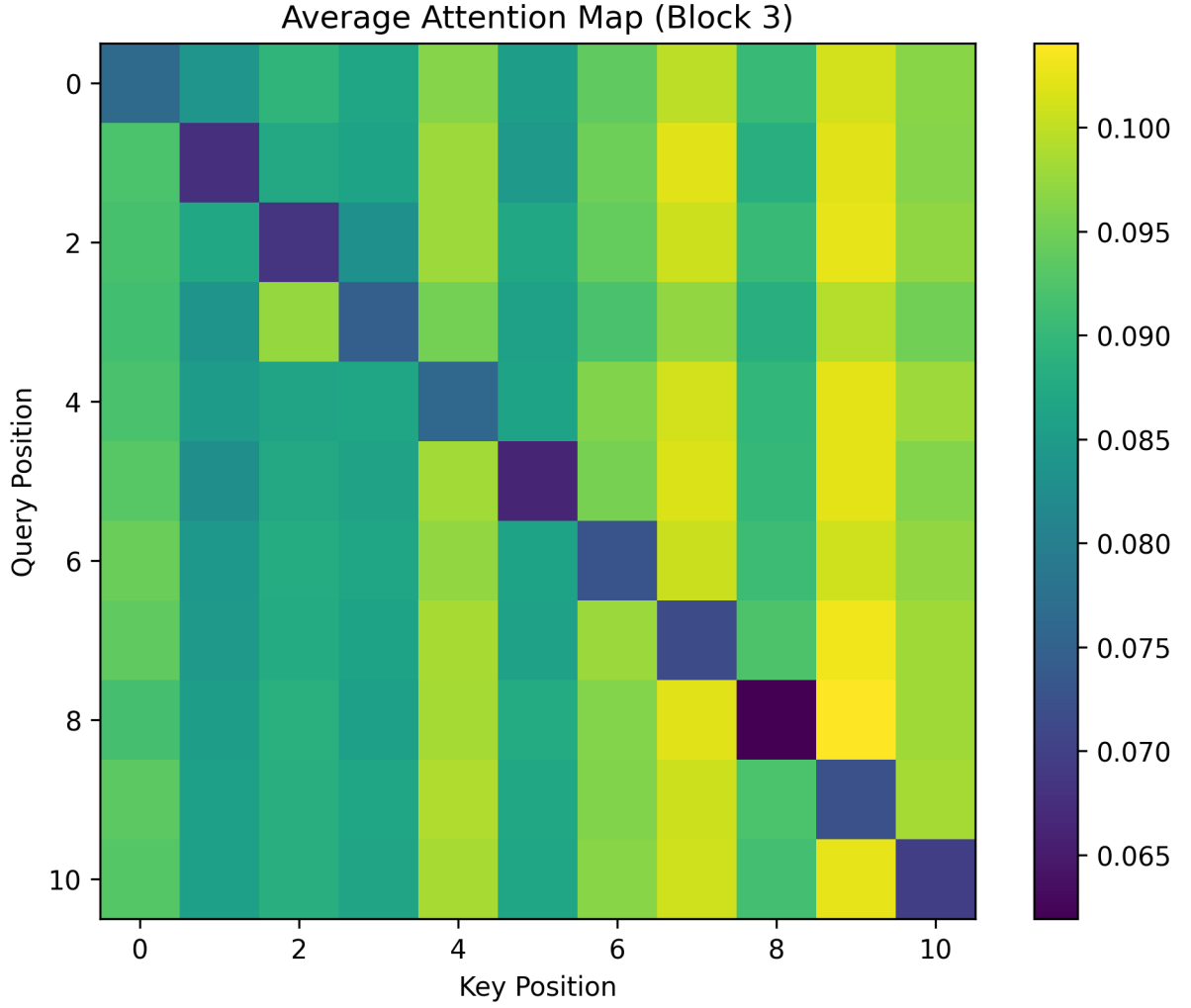


Figure 5.12: Average attention heatmap of the last encoder block after 50 epochs, baseline model, feature-wise scaled dataset.

These vertical bands in Figure 5.12 mean that, across almost every “query” feature, the model is repeatedly attending to the same few “key” features, specifically, in our case, features four, seven, and nine. The query and key features refer to their respective weight matrices in the multi-head attention formula defined in Subsection 1.2.6.

From all the evidence, we conclude feature importance is concentrated. The model is relying too much on a few features and is therefore not generalizing well to the patterns in the data.

Another set of model insights are the attribution maps as shown in Figures 5.13, 5.14, 5.15, and 5.16. They showcase feature importance in different scenarios, such as when the true class is signal and the model predicts signal, showing us what features nudge the model into making a correct prediction and vice versa. They also show us what creates confusion by observing an attribution map where the true class is background and the model predicts signal.

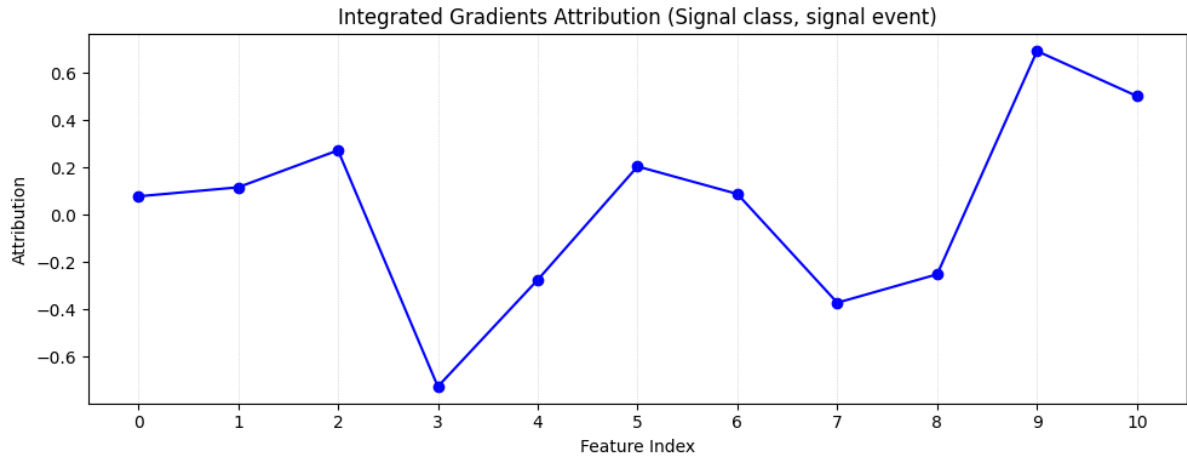


Figure 5.13: Attribution map for the signal class — signal event scenario, baseline model.

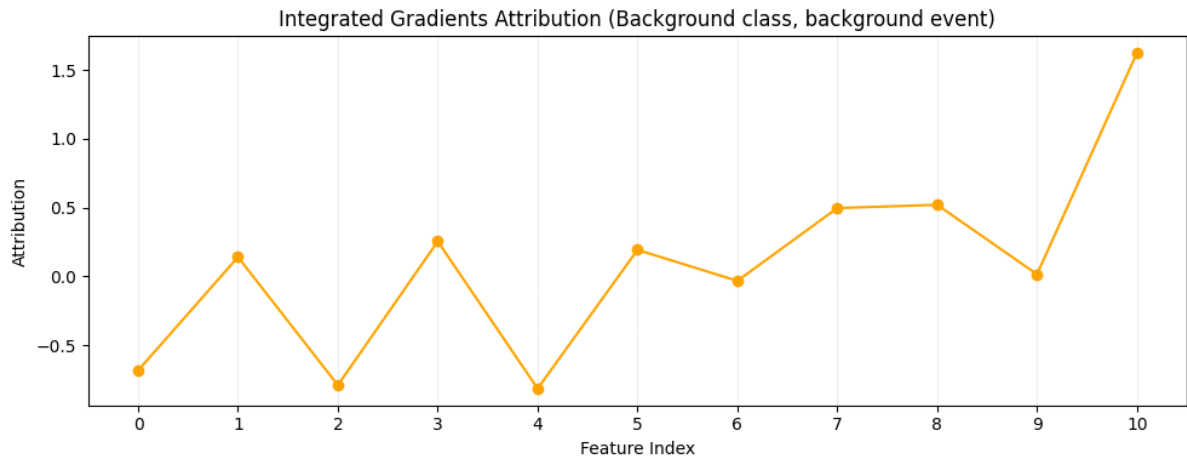


Figure 5.14: Attribution map for the background class — background event scenario, baseline model.

The attribution plots warrant attention for several reasons. A notable subset of features is at or near zero, indicating that the model considers them unimportant for their respective scenarios. To gain a complete understanding, we must analyze the four scenarios separately.



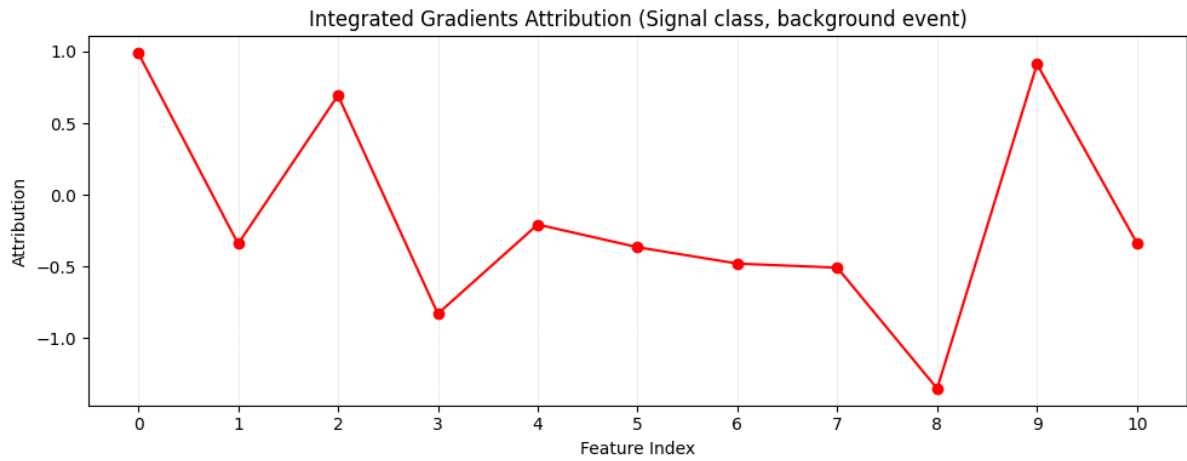


Figure 5.15: Attribution map for the signal class — background event scenario, baseline model.

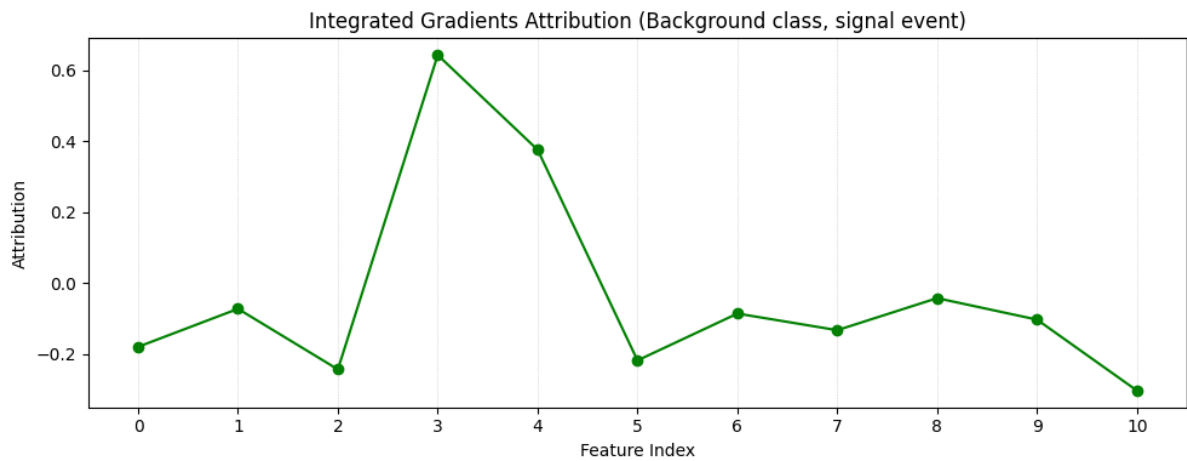


Figure 5.16: Attribution map for the background class — signal event scenario, baseline model.

From these plots, we observe a relatively strong reliance on a small amount of features for decision making. This reinforces the observations we made about the average attention map regarding three over-important features.

Knowing that the model’s activation values for the input projection and the last encoder block are sharply centered around zero, we can conclude the model is naturally ignoring some features completely in favor of others. This could be because of poor initialization, a low learning rate, or the lack of a regularization technique. We must construct a new model with an improved architecture to address these issues and retrain it, as detailed in Chapter 6.

## 5.6 Network Output

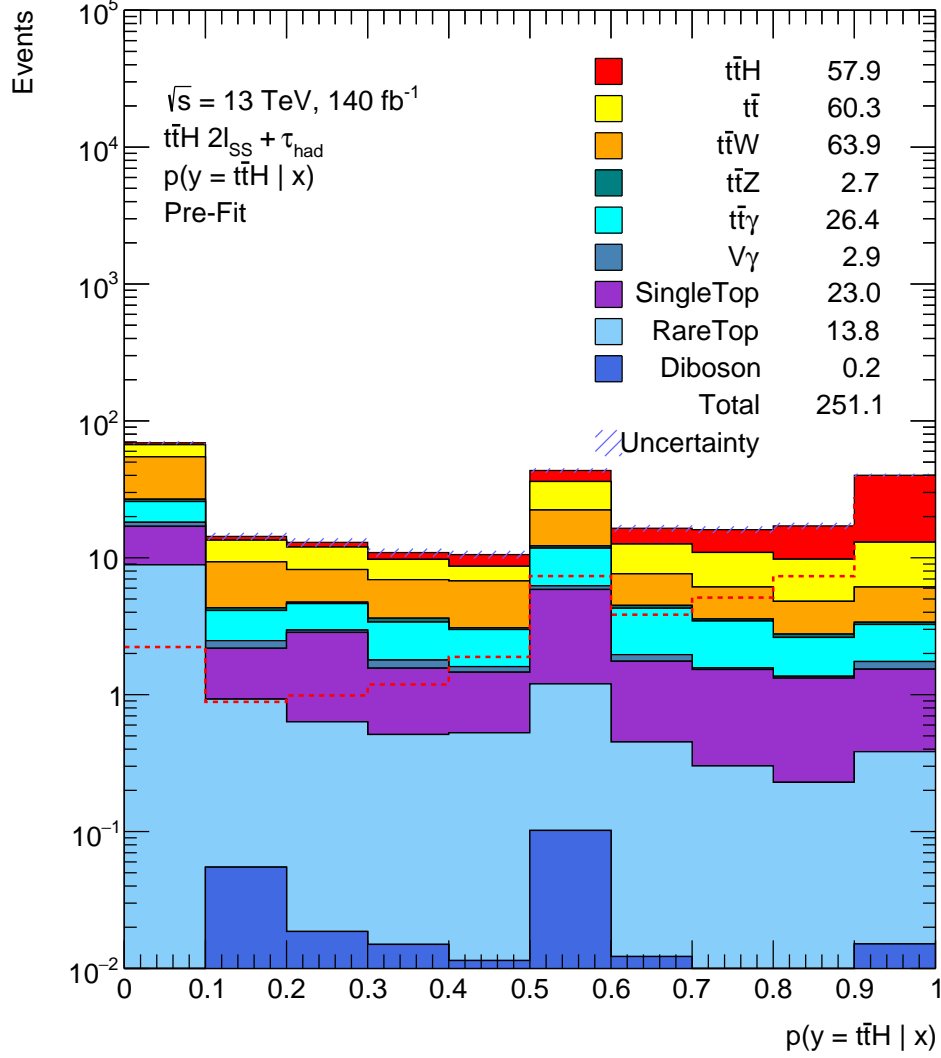


Figure 5.17: Distribution of the posterior probability  $P(y = t\bar{t}H | x)$  as estimated by the TRExFitter, baseline model trained on the feature-wise scaled dataset.

Figure 5.17, produced by the TRExFitter software, showcases the separation power of models. Here, the axes represent:

- **Horizontal Axis** ( $P(y = t\bar{t}H | x)$ ): the probability the model assigns to an event being  $t\bar{t}H$ .
- **Vertical Axis** (Events): the expected number of events in each probability bin, shown on a logarithmic scale.

Note the right edge of the plot, where the total expected yield from each process (in units of events) and the overall sum ( $\approx 251$  events) are denoted. The plot also displays

a dashed red line, this shows the pure  $t\bar{t}H$  distribution, (i.e. the distribution of the  $t\bar{t}H$  scaled up).

Figure 5.17 shows the models' predictions. The network output shows a couple of interesting aspects. The model is favoring our  $t\bar{t}H$  processes; we can see this from the slight mismatch of the dotted line shape over the background shape. However, background signal occupies a significant portion of the  $p > 0.9$  bins, so there is a lot of space for improvement.

## 5.7 Performance Comparison

The model trained and validated on z-score standardized data performed moderately better, while experiencing much of the same issues as the model trained on the feature-wise scaled data. It achieved a higher ROC score of 0.731 as well as producing a healthier average attention map (Figure 5.18).

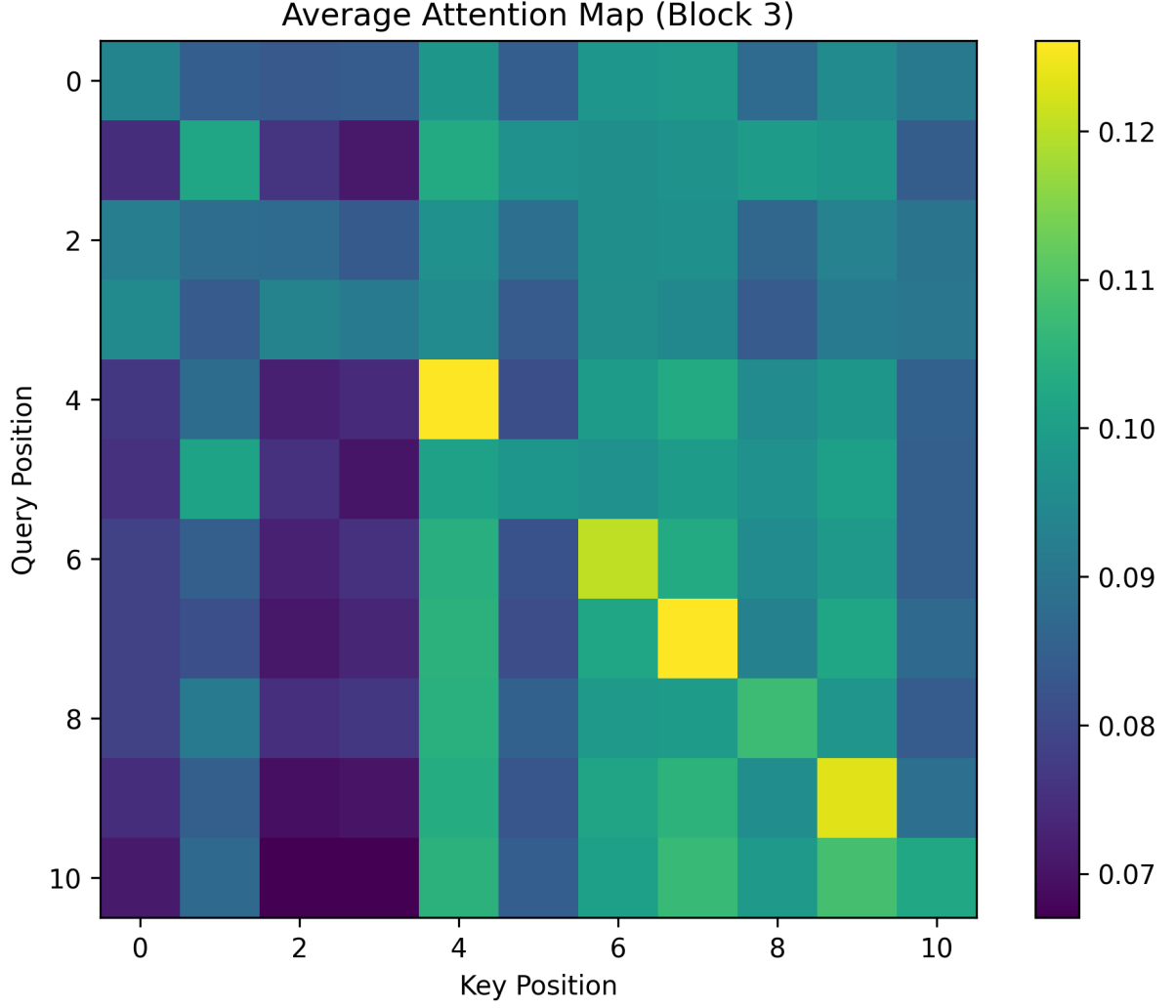


Figure 5.18: Average attention heatmap of the last encoder block after 50 epochs, baseline model, z-score standardized dataset.

The absence of clear horizontal or vertical bands in the heatmap tells us that, on average, no individual feature serves as a universal focal point (bright column), nor does any single token attend uniformly to all others (bright row). The model’s attention heads do not collectively converge on a specific special feature, nor do they distribute their focus equally across the entire sequence. Instead, the map shows two or more bright cells along the main diagonal (representing feature-to-itself pairs) that consistently receive elevated attention weights across multiple heads. These diagonal hotspots likely reflect strong self-attention pathways, preserving critical feature representations. Together, this suggests that the model is effectively detecting and exploiting inherent self-correlations within this data set. The activation histograms are quite similar between models, indicating that the choice of normalization scheme had minimal impact on the learned activation distributions. This robustness tells us that the model’s internal representations remain stable across different preprocessing strategies.

# Chapter 6

## Proposed Model

Creating a suitable attention-based ML model is inherently an iterative task. After multiple iterations, we have derived an optimal model architecture, from now on referred to as the proposed model. This architecture and training scheme employ advanced methods to prevent overfitting, reliably find the global minimum, and enhance the effectiveness of the attention mechanism.

Selecting the optimal architecture involves carefully balancing its advantages and drawbacks. In our case, we face pronounced overfitting, vanishing gradients, and suboptimal attention weight distributions. Figure 6.1 provides a high-level view of the overall model layout, illustrating the encoder and classification head structure.

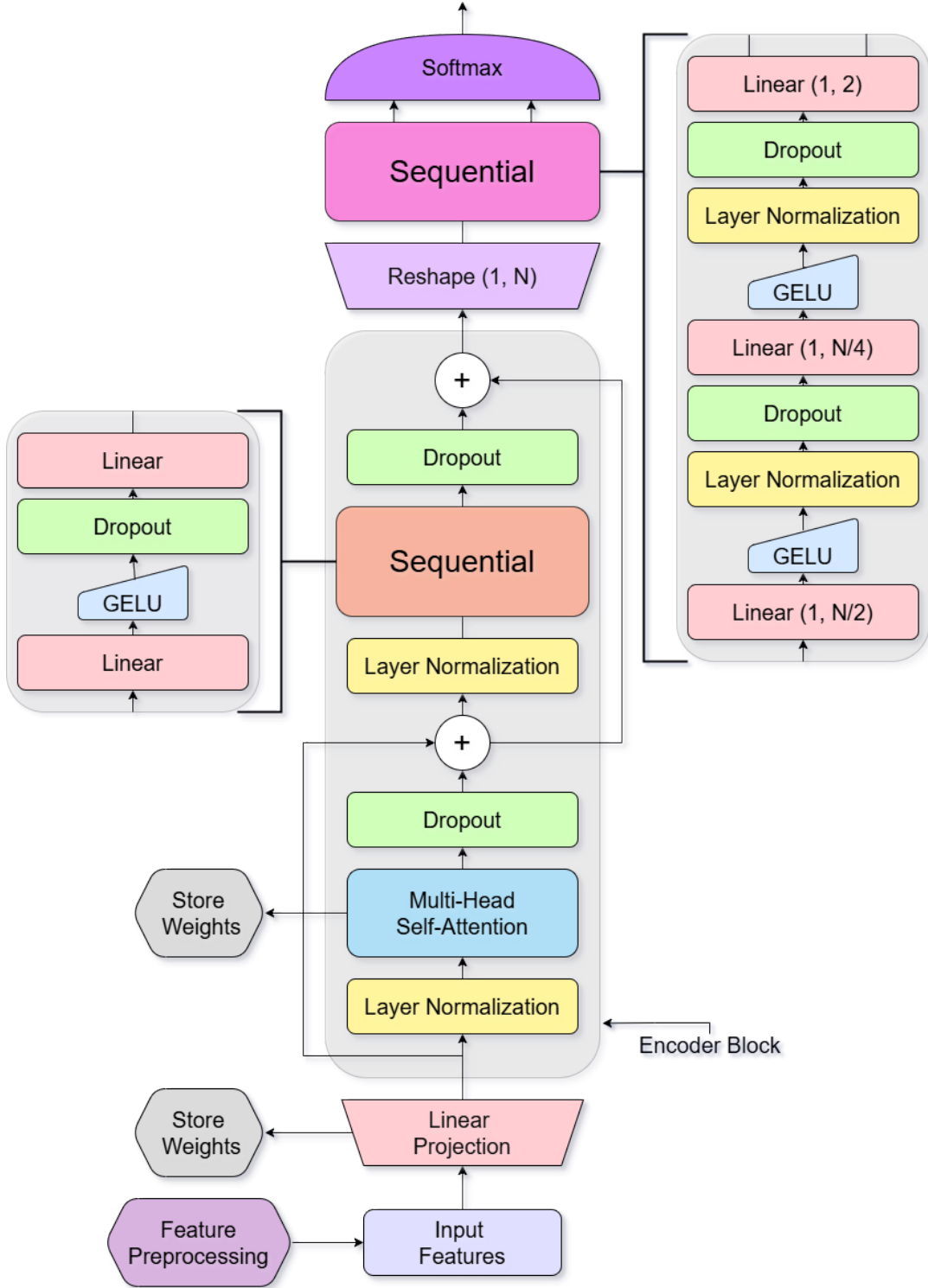


Figure 6.1: Diagram of the proposed model, where  $N = L \times d_{model}$ .

## 6.1 Overfitting

To prevent overfitting, dropout layers are placed inside the classification head and every encoder block. The dropout layers are strategically inserted after each multi-head atten-

tion layer and again after the feed-forward GELU activation in every encoder block, with the two extra dropout layers placed between the dense layers of the classification head. This dropout configuration regularizes both the encoder’s internal representations and the final projection before softmax.

To mitigate vanishing gradients, multiple residual connections are introduced immediately after each major computing block to preserve the original data and encourage healthier gradient flow. We also add a small MLP at the end of every encoder block. The attention mechanism is primarily linear in nature, so the MLP serves to add necessary non-linearity through a GELU activation and two linear transformations. The linear layers preserve the shape of the data to further stabilize gradient flow.

To address suboptimal attention distributions, we apply normalization just before each major computing block to stabilize the multi-head attention and feed-forward networks. This normalization allows the attention mechanism to better focus on subtle correlations in the data. The input projection is kept simple here. A single linear layer keeps gradient flow clean and allows the attention mechanism to perform the heavy lifting.

## 6.2 Weight Initialization

PyTorch naturally initializes all layers using sensible, effective, and battle-tested weight initialization techniques. For non-attention-based models, the default initialization schemes are often enough. However, transformer models are extra sensitive to weight initializations. In the proposed model, we utilize the Xavier initialization scheme for the attention and linear layer weights. The Xavier initialization has been chosen because it shows superior performance when applied to transformer models [29].

The Xavier initialization scheme sets the weights by drawing from a distribution with zero mean and a specific variance that depends on the number of inputs and outputs of a layer. For a layer with  $n_{in}$  input units and  $n_{out}$  output units, weights  $W$  are initialized as

$$W \sim \mathcal{U}\left(-\sqrt{\frac{6}{n_{in} + n_{out}}}, \sqrt{\frac{6}{n_{in} + n_{out}}}\right). \quad (6.1)$$

Where  $\mathcal{U}(a, b)$  denotes the continuous uniform distribution over the interval  $[a, b]$ . As done in the baseline model, normalization layers are initialized using PyTorch’s default settings.

### 6.3 Training & Hyperparameters

Training methodology and hyperparameter tuning are often as crucial as the architectural design of a model. In the proposed model, we utilize a range of advanced optimization techniques to ensure stable convergence and improved generalization performance. The key strategies used are as follows:

- **OneCycleLR Scheduler:** A three-phase learning rate schedule that first increases the learning rate, then decreases it, and finally maintains a low rate to stabilize convergence during the final training phase. Used in linear mode, the scheduler sets a maximum learning rate and scaling parameters to adjust the learning rate dynamically at each training step (per batch).
- **Batch Size Warm-up:** Gradually increasing the effective batch size during early epochs to stabilize training and reduce initial noise in gradient estimates. We start with an initial batch size and gradually increase it to the target batch size over a specified number of warm-up epochs.
- **AdamW Optimizer:** An improved variant of the Adam optimizer used in the baseline model that separates weight decay from the gradient update, leading to better regularization and generalization. Used with our dynamic learning rate scheduler and a weight decay hyperparameter.
- **Gradient Clipping:** Restricts the norm of gradients to prevent exploding gradients.
- **Class Weighting:** Balances the loss contribution of each class to address class imbalance. Since approximately 36.61% of the dataset consists of  $t\bar{t}H$  events, the model would otherwise be biased toward predicting background for a disproportionate number of events. We account for this by computing class weights and incorporating them into the loss function. Let  $n_{\text{total}}$  be the total number of training samples, with  $n_S$  and  $n_B$  representing the number of signal and background samples, respectively. The class weights are computed as:

$$w_S = \frac{n_{\text{total}}}{2n_S}, \quad w_B = \frac{n_{\text{total}}}{2n_B}. \quad (6.2)$$

- **Anti-Zero Weight Mechanism:** Prevents model weights from collapsing to zero by adding a custom penalty term to the loss function. During training, we compute a penalty proportional to the number of weights falling below a small threshold and scale it by a factor. This acts as a form of regularization, encouraging the model to maintain a minimum level of activation and discouraging dead or inactive units.



## 6.4 Hyperparameter Search

Optimizing hyperparameters for a complex model with numerous interacting components requires extensive experimentation. To address this, we utilize Optuna (Subsection 4.4.2), an efficient hyperparameter optimization framework, to explore and fine-tune the following key parameters:

- Model Dimension
- Number of Attention Heads
- Number of Encoder Blocks
- Dropout Rate
- Weight Decay
- Maximum Learning Rate
- Initial and Target Batch Size
- Number of Warm-up Epochs
- Zero-Weight Penalization Threshold and Factor

We rank iterations by the ROC AUC score and terminate sub-par trials early using Optuna’s pruning strategy, significantly accelerating convergence toward good configurations. We explore 100 trials per search session. The complete results, including parameter importances and optimization history, are saved for analysis. The same Optuna setup is used to optimize hyperparameters for both the feature-wise scaled dataset and the z-score standardized dataset. After analyzing both dataset schemes, we produce two hyperparameter sets:

Hyperparameter	Feature-wise Scaling	Z-score Standardization
<b>Model Architecture</b>		
D_MODEL	128	128
N_HEADS	4	8
N_LAYERS	6	6
DROP_OUT	0.22	0.19
WEIGHT_DECAY	$5.8 \times 10^{-4}$	$2.4 \times 10^{-5}$
<b>Training Setup</b>		
WARMUP_EPOCHS	9	9
INITIAL_BATCH_SIZE	256	768
TARGET_BATCH_SIZE	768	1792
MAX_LR	$3.8 \times 10^{-3}$	$2.1 \times 10^{-3}$
<b>Anti-zero Regularization</b>		
THRESHOLD	0.05	0.04
PENALTY_FACTOR	0.34	0.24

Table 6.1: Hyperparameters optimized by Optuna for the proposed model using both the feature-wise scaled dataset and the z-score standardized dataset.

## 6.5 Internal Dynamics Improvement

One of the goals of the proposed model is to improve the internal behavior of the network. We evaluate how the introduced regularization methods and architectural changes influence activation diversity and attention structure. We compare these internal signals against the baseline model and highlight improvements in representational capacity in the input projection layer and in the last encoder block.

### 6.5.1 Activations

One of the issues observed in the baseline model was a strong tendency to zero out features in the input projection layer and final encoder block. This issue was present equally in both datasets tested, and our modifications produced very similar plots for both models, so we will be analyzing the model trained on the feature-wise scaled dataset.

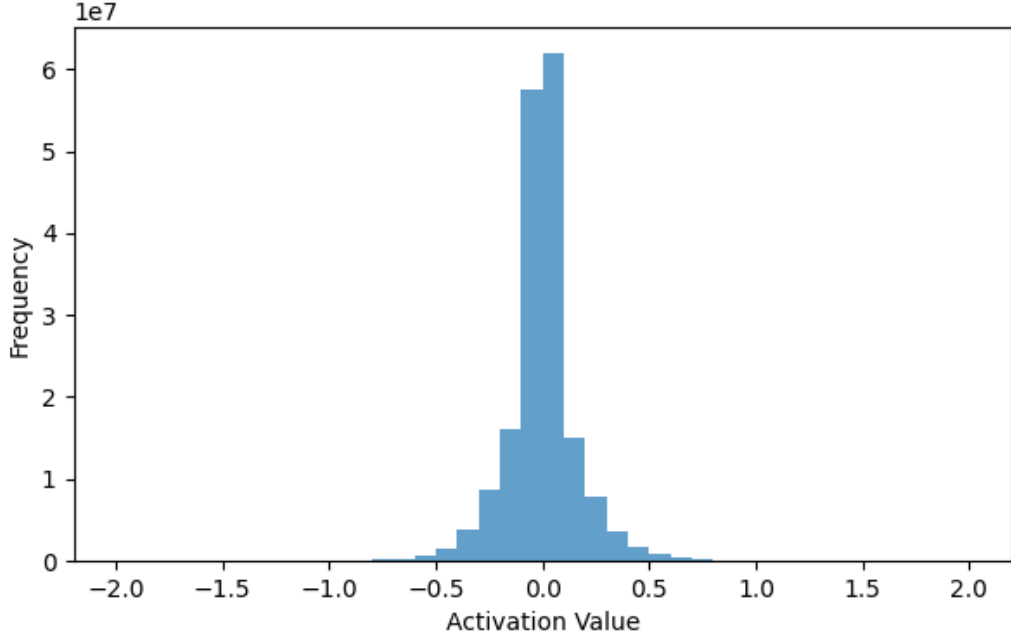


Figure 6.2: Activation values for the input projection layer after 40 epochs, proposed model.

We see a slight improvement in the input projection layer activation histogram, but a sharp distribution centered around zero is still present. This may be due to several factors, including the use of z-score normalized inputs for most features, even in the feature-wise scaled dataset, which naturally cluster around zero, and the Xavier initialization scheme, which produces weights with a symmetric distribution centered at zero. Activation functions like GELU can further suppress small inputs. The persistence of near-zero activations may also indicate that the model has learned to suppress certain input dimensions, treating them as uninformative to the classification task. It is important to note that we are penalizing weights within 0.05 of zero for the model, so many weights might still hover at or around the threshold. Where we really see our anti-zero regularization mechanic materialize is in the final encoder block.

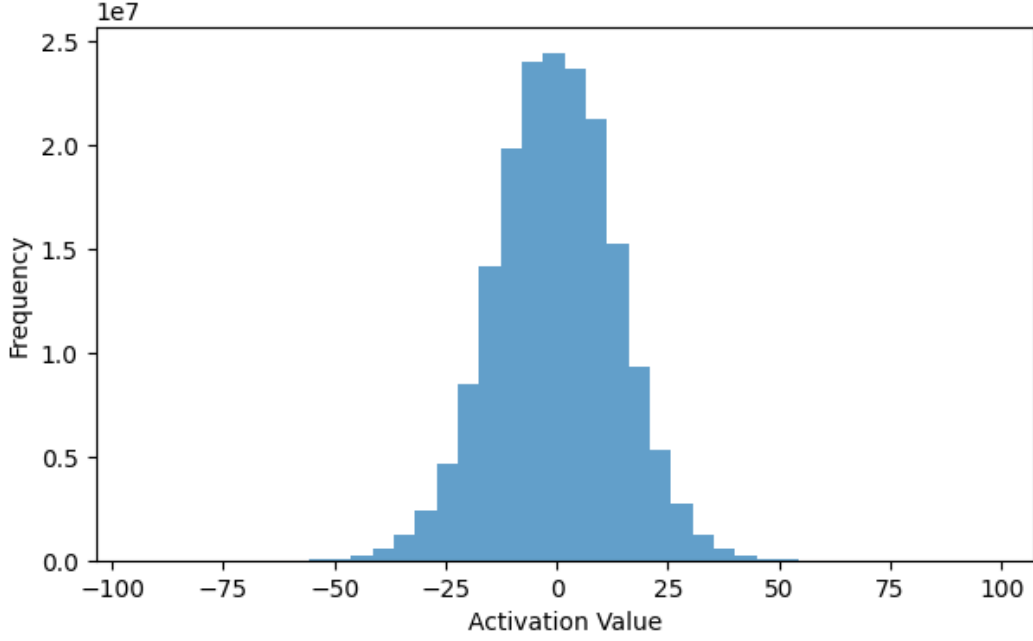


Figure 6.3: Activation values for the last encoder block after 40 epochs, proposed model.

The activations span a healthy range with no sharp clustering around zero. This indicates that the encoder blocks are active and using their capacity, rather than collapsing or saturating. We can conclude our models experienced healthy gradient flow and the network is not over-regularized or saturated.

### 6.5.2 Attention

In the baseline model, we observed attention layers dominated by strong feature-to-itself interactions and only a few bright columns, indicating limited cross-feature attention. Figure 6.4 represents the attention map of head 0 in our last encoder block.

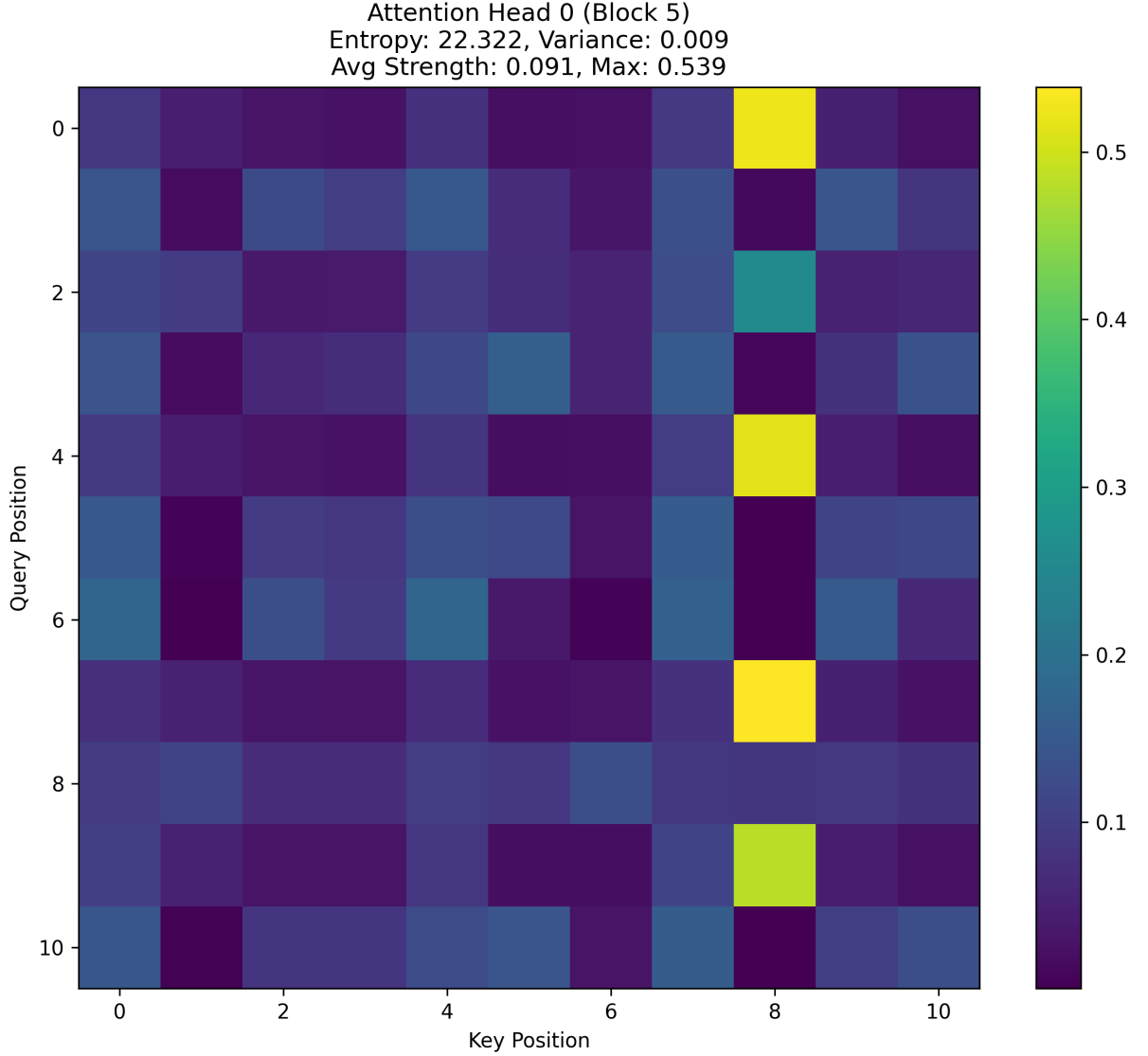


Figure 6.4: Attention heatmap of head 0 in the last encoder block after 40 epochs, proposed model, feature-wise scaled dataset.

The attention head in the final encoder block shows an entropy of 22.322, a low variance of 0.009, and an average attention strength of 0.091, with a maximum score of 0.539. These metrics indicate a broadly distributed attention pattern, suggesting that the head is aggregating information across many features rather than concentrating on a narrow subset, as observed in Figure 5.18.

However, the attention map also shows us four noticeably brighter cells that are aligned with feature index 8, telling us that multiple features consistently direct their attention toward this particular key. This suggests that while the head maintains a sparse attention profile overall, it still finds feature 8 as especially informative.

This pattern is consistent with the activation histogram of the final encoder block, which shows a healthy and diverse range of activations. Together, these observations

show us that the model not only preserves representational richness but also develops targeted sensitivity to specific high-importance features such as feature 8.

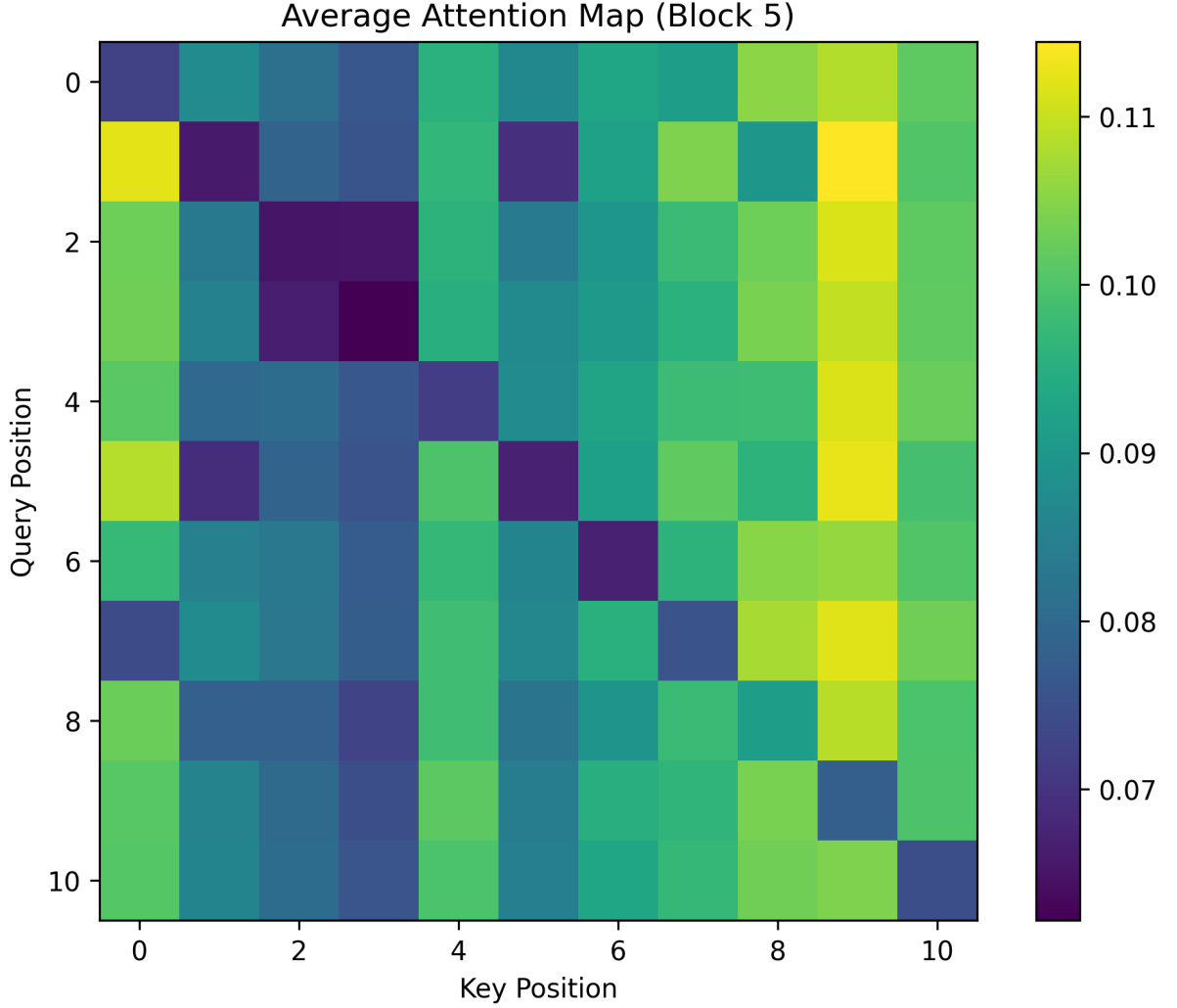


Figure 6.5: Average attention heatmap of the last encoder block after 40 epochs, proposed model, feature-wise scaled dataset.

The average attention map of the final encoder block (Figure 6.5) does not highlight feature 8 as a dominant focus as indicated in head zero (Figure 6.4). This tells us that the strong attention toward feature 8 is specific to a single attention head, rather than a block-wide pattern. The overall attention is broadly distributed, with no single feature universally prioritized across heads. This diversity in attention behavior suggests that different heads specialize in attending to different parts of the input, giving us richer and more varied feature interactions in the final representation.

## 6.6 Performance Metrics

We highlight the performance and separation power achieved by the proposed model on the feature-wise scaled dataset and the z-scored dataset. Using 5-fold cross-validation, we obtain  $0.801 \leq \text{ROC score} \leq 0.805$  and  $0.801 \leq \text{ROC score} \leq 0.804$  for the feature-wise scaled dataset and the z-scored dataset, respectively.

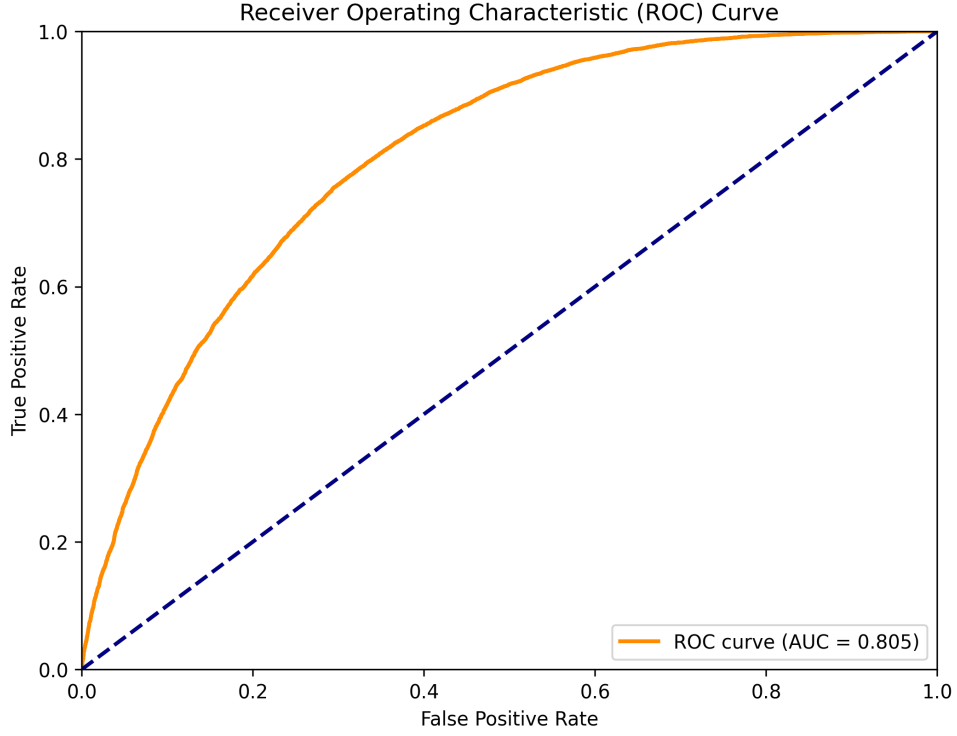


Figure 6.6: ROC curve of the proposed model trained on the feature-wise scaled dataset.

Compared to the baseline, our model's ROC score improves by 0.082 (from 0.723 to 0.805), representing a substantial gain in discriminative power between signal and background. But the improved ROC score does not tell us much about the statistical improvements. For that, we analyze the statistical performance of our two models.

### 6.6.1 Feature-wise Scaled Dataset

Class	Precision	Recall	F <sub>1</sub> -score	Support
background	0.8539	0.6492	0.7376	20 268
signal	0.5717	0.8082	0.6697	11 752
accuracy			0.7075	32 038
macro avg	0.7128	0.7287	0.7036	32 038
weighted avg	0.7503	0.7075	0.7127	32 038

Table 6.2: Classification report for background versus signal, proposed model trained on the feature-wise scaled dataset.

The proposed model shows gains across nearly every metric. The small dip in background recall is offset by improvements elsewhere. These statistics are compared to the baseline model trained on the feature-wise scaled dataset.

- **Background class:**

- Precision rose from 79.29% to 85.39% (+6.10 pp, +7.69%).
- Recall decreased from 65.89% to 64.92% (-0.97 pp, -1.47%).
- F<sub>1</sub>-score improved from 71.97% to 73.76% (+1.79 pp, +2.49%).

The background class contains the most modest of the improvements. However, a +7.69% improvement in precision is encouraging.

- **Signal class:**

- Precision rose from 53.97% to 57.17% (+3.20 pp, +5.93%).
- Recall jumped from 69.92% to 80.82% (+10.90 pp, +15.59%).
- F<sub>1</sub>-score increased from 60.92% to 66.97% (+6.05 pp, +9.94%).

The signal class is where we want to divert most of our attention. Although recall is at a very promising level, a precision score of 57.17% still implies about 42.83% false positives, which calls for further investigation in Chapter 7.

- **Overall performance:**

- Accuracy improved from 67.36% to 70.75% (+3.39 pp, +5.03%).
- Macro-average F<sub>1</sub>-score improved from 66.44% to 70.36% (+3.92 pp, +5.90%).
- Weighted-average F<sub>1</sub>-score improved from 67.95% to 71.27% (+3.32 pp, +4.89%).

From the balanced gains, we can conclude the proposed model offers a clear, overall lift in classification performance.



### 6.6.2 Z-score Standardized Dataset

The proposed model trained on the z-scored dataset performed very similarly to the model trained on the feature-wise scaled dataset.

Class	Precision	Recall	F <sub>1</sub> -score	Support
background	0.8562	0.6455	0.7361	20 241
signal	0.5723	0.8140	0.6721	11 797
accuracy			0.7075	32 038
macro avg	0.7143	0.7297	0.7041	32 038
weighted avg	0.7517	0.7075	0.7125	32 038

Table 6.3: Classification report for background versus signal, proposed model trained on the z-score standardized dataset.

Its improvements, or slight downgrades as seen in background recall, over the proposed model trained on the feature-wise scaled dataset for each class are represented below:

- **Background class:**

- Precision rose from 85.39% to 85.62% (+0.23 pp, +0.27%).
- Recall decreased from 64.92% to 64.55% (-0.37 pp, -0.57%).
- F<sub>1</sub>-score decreased from 73.76% to 73.61% (-0.15 pp, -0.20%).

- **Signal class:**

- Precision rose from 57.17% to 57.23% (+0.06 pp, +0.11%).
- Recall rose from 80.82% to 81.40% (+0.58 pp, +0.72%).
- F<sub>1</sub>-score rose from 66.97% to 67.21% (+0.24 pp, +0.36%).

- **Overall performance:**

- Accuracy remained unchanged at 70.75% ( $\pm 0.00$  pp,  $\pm 0.00\%$ ).
- Macro-average F<sub>1</sub>-score rose from 70.36% to 70.41% (+0.05 pp, +0.07%).
- Weighted-average F<sub>1</sub>-score decreased from 71.27% to 71.25% (-0.02 pp, -0.03%).

The proposed model trained on the z-score standardized dataset edges out the proposed model trained on the feature-wise scaled dataset on most metrics, especially signal recall (+0.0058 pp, +0.72%) and overall precision averages, though all gains are under 1%. We can reliably conclude that the non-uniform standardization had little, but non-negligible, impact on the performance of our attention model.

# Chapter 7

## Results and Discussion

### 7.1 Final Model Performance

The final model, based on a six-layer, multi-head self-attention architecture trained on z-score standardized input features, achieved an ROC AUC score of 0.804 and an  $F_1$ -score of 0.7041. This demonstrates respectable classification capability for distinguishing  $t\bar{t}H$  events from the background.

#### 7.1.1 Statistical Uncertainty

In order to quantify the overall uncertainty of our classifier, we use TRExFitter to evaluate the signal strength  $\mu$  and its  $\pm 1\sigma$  statistical uncertainty in two scenarios. First, with no classifier applied (single-bin setup), we obtain Figure 7.1.

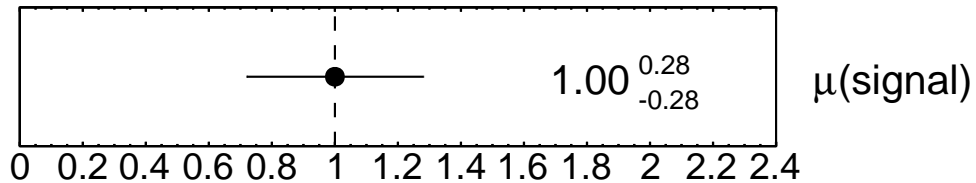


Figure 7.1: Normalization factor for the single-bin TRExFitter setup:  
 $\mu_1 = 1 \pm 0.28$ .

This represents the baseline statistical uncertainty when only the total event count is used. When we instead apply our model and split the discriminant into ten bins, the network output yields the uncertainty shown in Figure 7.2.

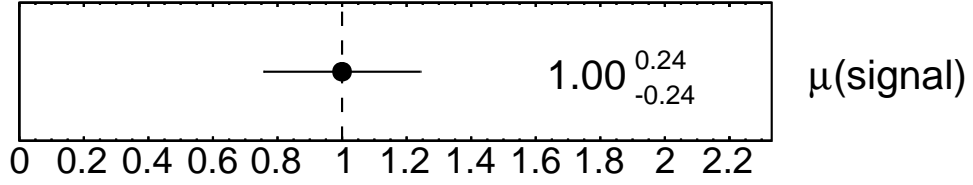
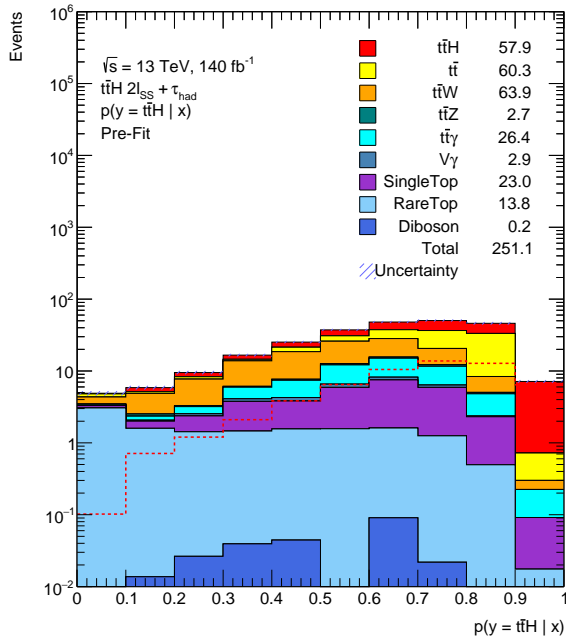


Figure 7.2: Normalization factor for the ten-bin TRExFitter setup with our proposed model, trained on the z-score standardized dataset, applied:

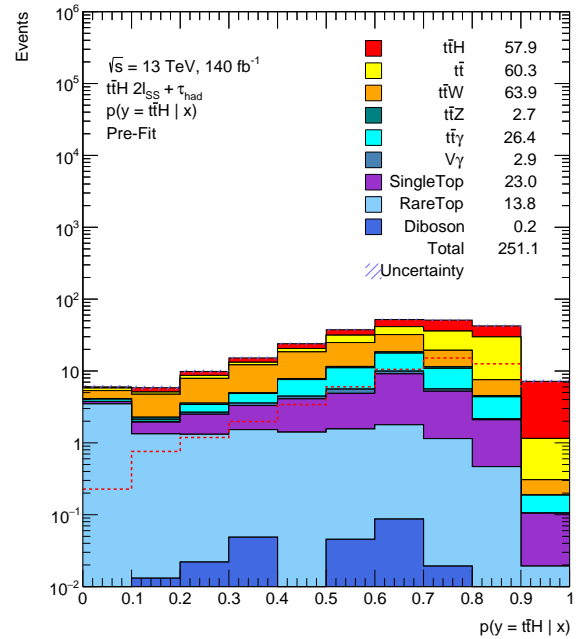
$$\mu_{10} = 1 \pm 0.24$$

This indicates a clear improvement in the signal-strength statistical uncertainty and demonstrates enhanced precision in the overall normalization. Performance comparison shows that the feature-wise scaled model attains slightly reduced precision. Its ten-bin network output yields  $\mu_{10} = 1 \pm 0.25$ , which is 0.01 less precise than the result obtained with z-score standardization.

### 7.1.2 Network Output



Distribution of the posterior probability  $P(y = t\bar{t}H | x)$  for the feature-wise scaled model.



Distribution of the posterior probability  $P(y = t\bar{t}H | x)$  for the z-score standardized model.

Figure 7.3a and Figure 7.3b showcase the posterior probability distributions estimated by TRExFitter for the feature-wise scaled and z-score standardized models, respectively.

Both distributions show that predictions of  $P(y = t\bar{t}H \mid x) = 1$  are relatively infrequent. These high-confidence assignments mostly capture true  $t\bar{t}H$  events, while a large concentration of events falls in the 0.6-0.9 probability range, reflecting the model’s moderate confidence in cases where signal and background features overlap. Considering the slight performance increase seen in the proposed model trained on the z-score standardized dataset, we assume that changes in feature correlations introduced by feature-wise scaling had a greater negative impact than the performance improvements gained by healthier feature distributions.

## 7.2 Dataset Considerations

Because we are the first to use Release 22, the contemporary selection cuts and object definitions were not yet fully optimized, resulting in a dataset that falls short of the clean, tightly tuned sample we could have produced under Release 21. In particular, the looser preselection has introduced an overabundance of background events such as  $t\bar{t}$  and  $t\bar{t}W$ , and a larger total event count than is typical in previous studies.

To verify that our model’s separation power is constrained by the data rather than the architecture or training methodology, we benchmarked the same tabular inputs with XGBoost [30], a gold-standard algorithm renowned for its performance on tabular data classification. After hyperparameter tuning and cross-validation, the XGBoost classifier achieved an ROC AUC of 0.8052 and an  $F_1$ -score of 0.71 on the feature-wise scaled dataset, an ROC AUC of 0.8053 and an  $F_1$ -score of 0.70 on the z-score standardized dataset, and an ROC AUC of 0.8053 and an  $F_1$ -score of 0.71 on the unnormalized dataset. The nearly identical performance of XGBoost, an algorithm many classify as state-of-the-art for tabular data, and our proposed model confirms limitations in the dataset.

## 7.3 Summary of Experimental Findings

After training both our proposed self-attention model and a benchmark XGBoost classifier on our simulated dataset corresponding to the  $2\ell_{\text{SS}} + \tau_{\text{had}}$  final state, we performed a side-by-side comparison of their discrimination power and performance metrics. While the XGBoost model achieved an ROC AUC of 0.8053 and an  $F_1$ -score of 0.71, our self-attention-based architecture closely matched these metrics.

We can reliably conclude that the self-attention mechanism, accompanied by suitable architecture choices and training and regularization schemes, demonstrated several advantages in the context of HEP data:

- **Adaptive Feature Learning:** Individual attention heads capture key feature relationships, incorporating the pairs into a larger representation of the event.
- **Enhanced Interpretability:** Head- and block-level heatmaps and attribution maps highlight physically meaningful inputs, providing a view of model focus we can check with intuition.
- **Preprocessing Robustness:** Consistent performance under both z-score and feature-wise scaling indicates attention naturally adapts to input scales.
- **Balanced Capacity:** Extensive regularization and dynamic scheduling harness the model’s parameter count without overfitting.

These results demonstrate that self-attention not only rivals tree-based methods on tabular HEP data but also brings built-in interpretability and flexibility.

## 7.4 Implications & Future Directions

This thesis demonstrates that attention layers effectively capture inter-feature correlations. Through visualizations of attention weights and attribution maps, we show how the model prioritizes meaningful inputs, offering both interpretability and strong performance.

Rivalling established algorithms such as XGBoost indicates that this model could serve as a powerful signal extractor in future  $t\bar{t}H$  analyses. As Releases 22 matures and produces more robust and attractive signal region cuts and datasets, we can expect this model to further improve. In the future, we will:

- Improve signal region cuts and include more features in the analysis.
- Implement a similar model to look for hypothesized particles such as charged Higgs bosons or leptoquarks.

## Conclusion

This thesis demonstrates that self-attention-based architectures are not only viable but effective for tabular classification tasks in high-energy physics. By designing a robust architecture, utilizing state-of-the-art learning and regularization techniques, and tuning hyperparameters with Optuna, we obtained an expected precision of 24% on the signal-strength parameter  $\mu$  using the z-score standardized dataset. Although systematic uncertainties and real-data effects remain a subject for future work, our results demonstrate that attention mechanisms can robustly capture subtle correlations in  $2\ell_{\text{SS}} + \tau_{\text{had}}$  events. With further refinement and broader integration, these models have strong potential to contribute to future  $t\bar{t}H$  sensitivity studies and related measurements.

# Appendix A

## Appendix

### A.1 Event Weighting and TRExFitter NormFactor

All Monte Carlo events in this analysis are weighted by the branch `weight_total_NOSYS`, which encapsulates the full per-event normalization:

$$w_{\text{event}} = \text{weight\_total\_NOSYS},$$

including the generator-level normalization, cross-section scaling, pileup reweighting, lepton and  $b$ -tagging scale factors, and trigger efficiencies. No additional per-event weights or hand-tuned scale factors are applied at the analysis level; all uncertainty propagation is handled in the statistical fit.

In the TRExFitter configuration, the signal strength parameter  $\mu_{\text{signal}}$  is defined via a `NormFactor` block that allows the  $t\bar{t}H$  yield to float in the likelihood fit:

```
NormFactor: "mu_signal"  
    Max:      100  
    Min:     -100  
    Nominal:    1  
    Samples: ttH  
    Region: "probs-ttH"  
    Title:   "#mu(signal)"
```

Here:

- **Max**, **Min** define the allowed range of  $\mu_{\text{signal}}$ .
- **Nominal** is the pre-fit value ( $1 \times$  the SM prediction).
- **Samples** specifies that this factor applies only to the  $t\bar{t}H$  sample.

- **Region** specifies the fit region in which the normalization factor is applied.
- **Title** gives the label used in fit summaries and plots.

This approach ensures that the overall normalization of the signal is the only free normalization parameter, with all effects already folded into `weight_total_NOSYS`.

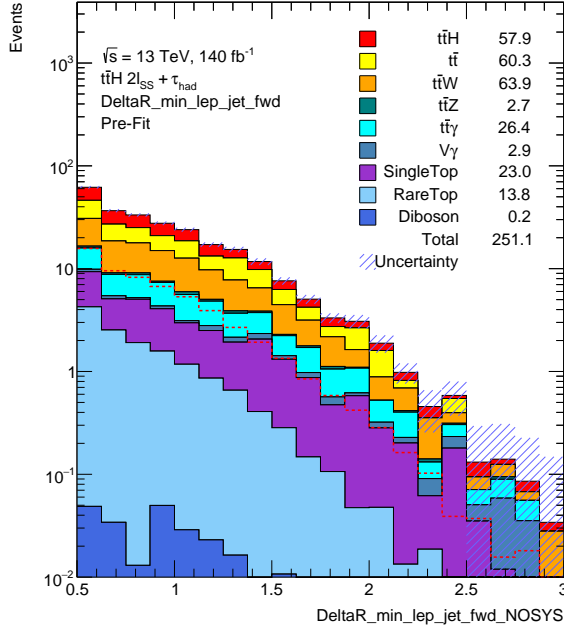


## A.2 Hyperparameters

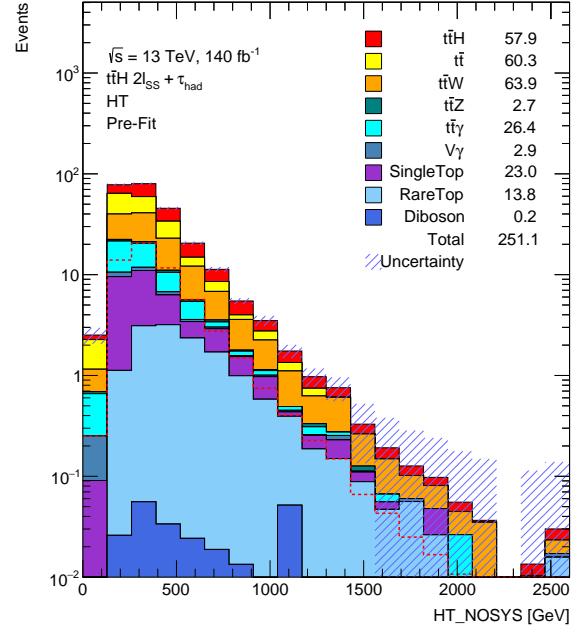
Hyperparameter	Feature-wise Scaling	Z-score Standardization
<b>Model Architecture</b>		
D_MODEL	128	128
N_HEADS	4	8
N_LAYERS	6	6
DROP_OUT	0.22	0.19
WEIGHT_DECAY	$5.8 \times 10^{-4}$	$2.4 \times 10^{-5}$
<b>Training Hyperparameters</b>		
EPOCHS	100	100
WARMUP_EPOCHS	9	9
PCT_START	0.3	0.3
<b>Learning Rate Hyperparameters</b>		
MAX_LR	$3.8 \times 10^{-3}$	$2.1 \times 10^{-3}$
DIV_FACTOR	25.0	25.0
FINAL_DIV_FACTOR	$1 \times 10^4$	$1 \times 10^4$
<b>Batch Size Hyperparameters</b>		
INITIAL_BATCH_SIZE	256	768
TARGET_BATCH_SIZE	768	1792
GRADIENT_CLIP	1	1
<b>Anti-zero Regularization</b>		
THRESHOLD	0.05	0.04
PENALTY_FACTOR	0.34	0.24

Table A.1: Full list of all hyperparameters for the proposed model using both the feature-wise optimally scaled dataset and the z-score standardized dataset.

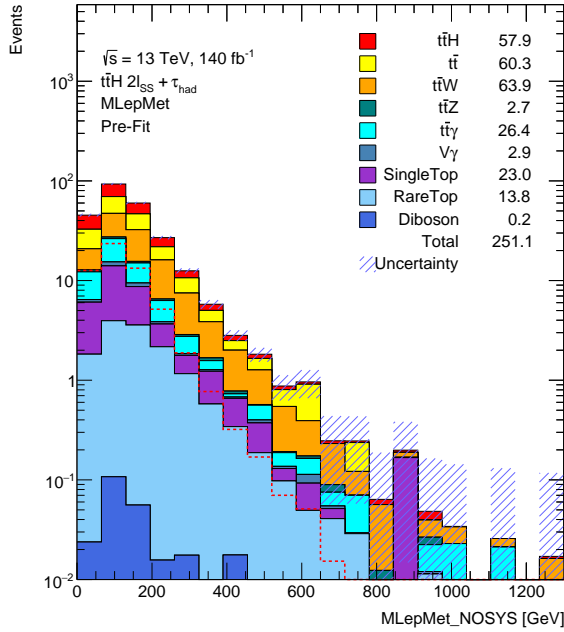
### A.3 Normalized Feature Distributions



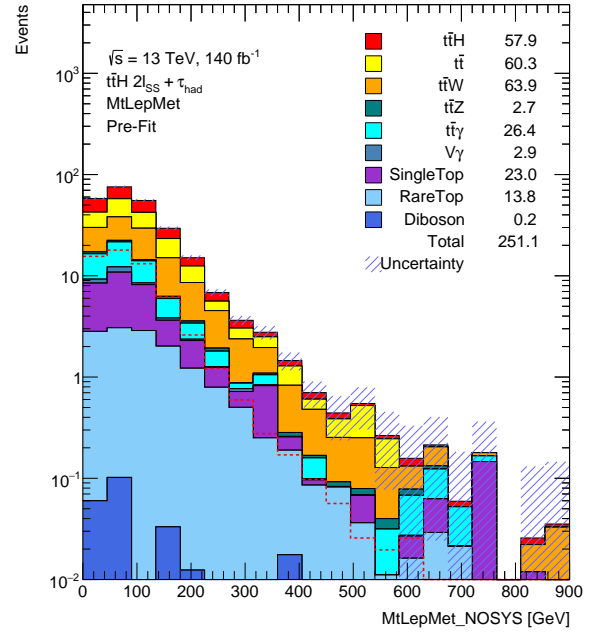
DeltaR\_min\_lep\_jet\_fwd\_NOSYS



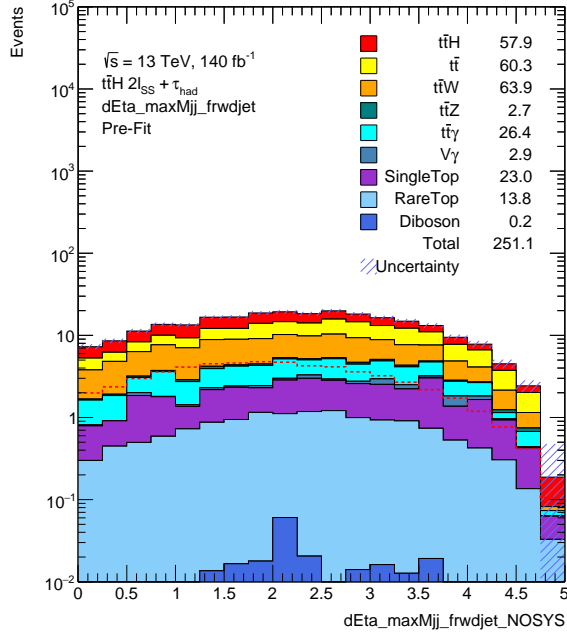
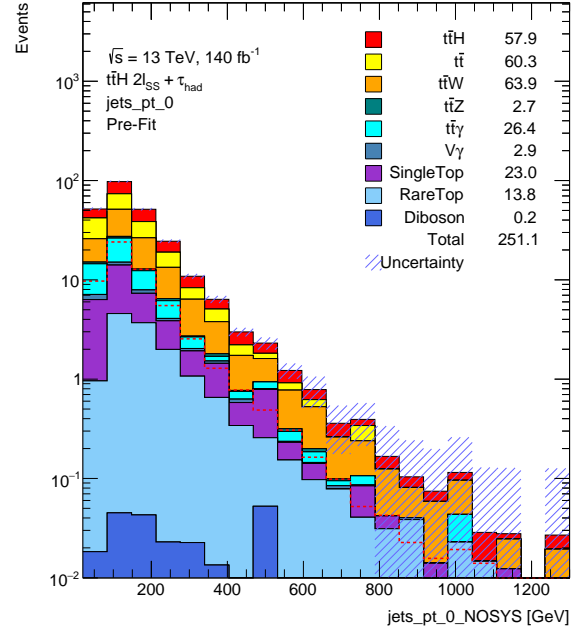
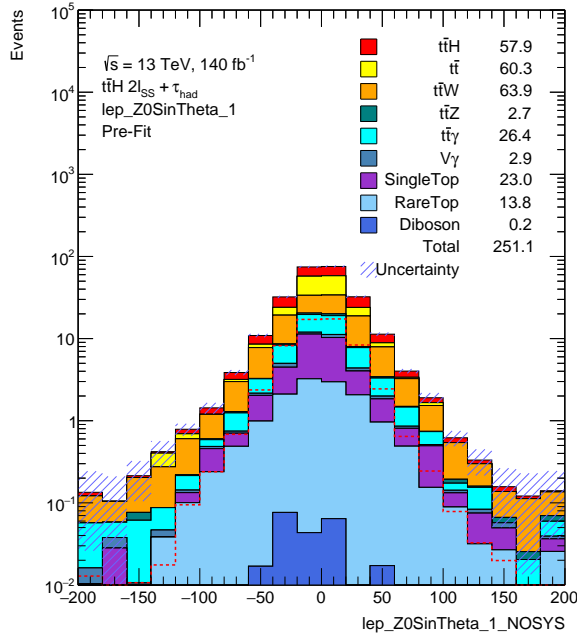
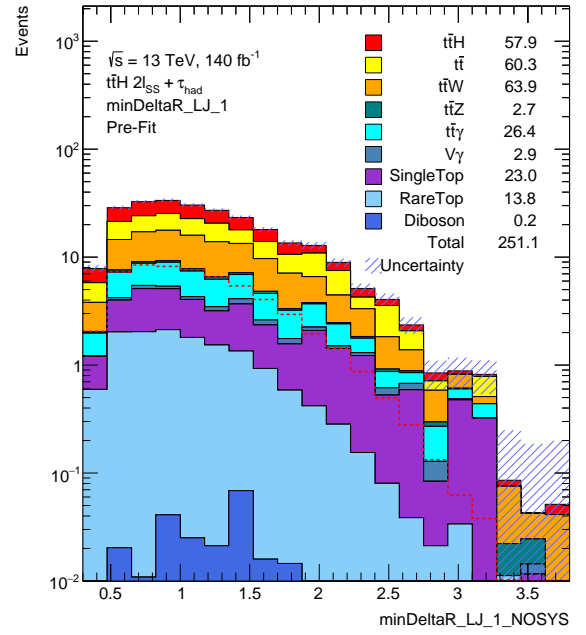
HT\_NOSYS

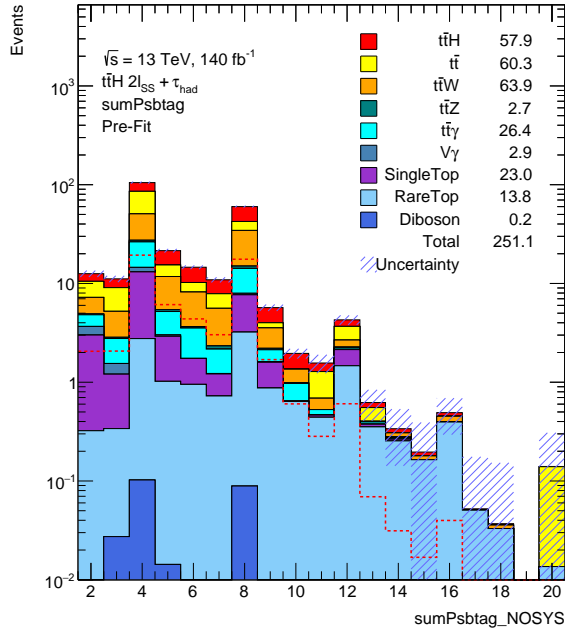


MLepMet\_NOSYS

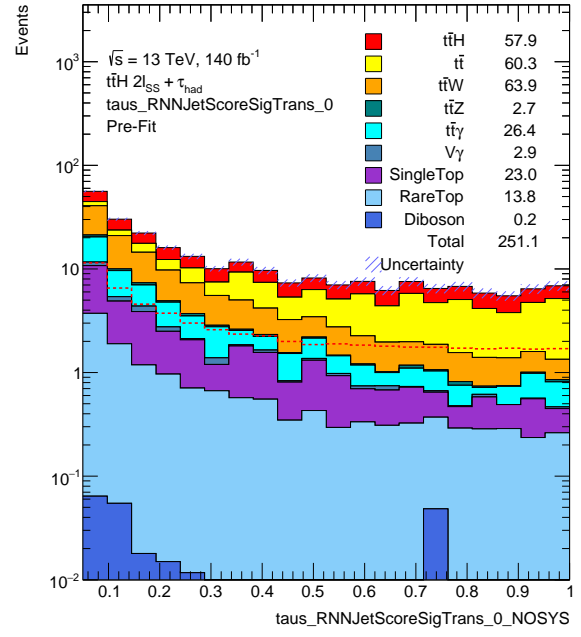


MtLepMet\_NOSYS

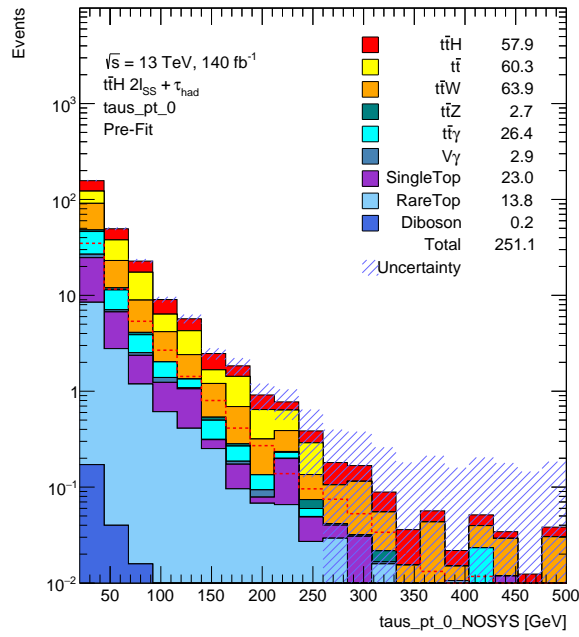
 $d\text{Eta\_maxMjj\_frwdjet\_NOSYS}$  $\text{jets\_pt\_0\_NOSYS}$  $\text{lep\_Z0SinTheta\_1\_NOSYS}$  $\text{minDeltaR\_LJ\_1\_NOSYS}$



sumPsbttag\_NOSYS



taus\_RNNJetScoreSigTrans\_0\_NOSYS



taus\_pt\_0\_NOSYS

## A.4 Software Dependencies

Package == Version	Package == Version
absl-py==2.2.2	aiohappyeyeballs==2.6.1
alembic==1.15.2	async-timeout==5.0.1
attrs==25.3.0	awkward_cpp==45
awkward==2.8.1	bottle==0.13.3
certifi==2025.1.31	captum==0.8.0
charset-normalizer==3.4.1	click==8.1.8
cycler==0.12.1	filelock==3.13.1
fonttools==4.57.0	fsspec==2025.3.2
grpcio==1.71.0	hist==2.8.1
histoprint==2.6.0	idna==3.10
Jinja2==3.1.4	joblib==1.4.2
Markdown==3.7	MarkupSafe==2.1.5
matplotlib==3.9.4	mpmath==1.3.0
multidict==6.3.2	networkx==3.2.1
numpy==1.26.4	nvidia-cublas-cu11==11.11.3.6
nvidia-cuda-cupti-cu11==11.8.87	nvidia-curand-cu11==10.3.0.86
nvidia-cuda-nvrtc-cu11==11.8.89	nvidia-cuda-runtime-cu11==11.8.89
nvidia-cusolver-cu11==11.4.1.48	nvidia-cusparse-cu11==11.7.5.86
nvidia-cudnn-cu11==9.1.0.70	nvidia-cufft-cu11==10.9.0.58
nvidia-nccl-cu11==2.21.5	nvidia-nccl-cu12==2.26.2
nvidia-nvtx-cu11==11.8.86	optuna==4.3.0
optuna-dashboard==0.18.0	packaging==24.2
pandas==2.2.3	pillow==11.0.0
psutil==7.0.0	pyparsing==3.2.3
python-dateutil==2.9.0.post0	pytorch-tabnet==4.1.0
pytz==2025.2	PyYAML==6.0.2
requests==2.32.3	scikit-learn==1.6.1
scipy==1.13.1	seaborn==0.13.2
threadpoolctl==3.6.0	torch==2.6.0+cu118
torch-geometric==2.6.1	torch-tb-profiler==0.4.3
torchaudio==2.6.0+cu118	torchvision==0.21.0+cu118
tqdm==4.67.1	triton==3.2.0
typing_extensions==4.13.1	tzdata==2025.2
uhi==0.5.0	uproot==5.6.0
urllib3==2.3.0	Werkzeug==3.1.3
xgboost==2.1.4	xxhash==3.5.0

Table A.2: Project dependencies.

# Bibliography

- [1] *The Higgs boson*, Apr. 2025. [Online]. Available: <https://home.cern/science/physics/higgs-boson>.
- [2] *Antiparticles*. [Online]. Available: <http://hyperphysics.phy-astr.gsu.edu/hbase/Particles/antimatter.html>.
- [3] *Spin*. [Online]. Available: <https://atlas.cern/glossary/spin>.
- [4] S. J. Ling, J. Sanny, and W. Moebs, *The Standard Model*. [Online]. Available: <https://pressbooks.online.ucf.edu/osuniversityphysics3/chapter/the-standard-model/>.
- [5] ATLAS Collaboration, *What's so special about the Higgs boson?*, Apr. 2025. [Online]. Available: <https://home.cern/science/physics/higgs-boson/what>.
- [6] *TRExFitter*. [Online]. Available: <https://trexfitter-docs.web.cern.ch/trexfitter-docs/>.
- [7] *Pytorch Activation Functions*. [Online]. Available: [https://colab.research.google.com/github/aaubs/ds-master/blob/main/notebooks/M3\\_1\\_Activation\\_Functions\\_Tutorial.ipynb](https://colab.research.google.com/github/aaubs/ds-master/blob/main/notebooks/M3_1_Activation_Functions_Tutorial.ipynb).
- [8] DeepLearning.AI, *Natural Language Processing (NLP) [a complete guide]*, Jan. 2023. [Online]. Available: <https://www.deeplearning.ai/resources/natural-language-processing/>.
- [9] Srikantha, *Understanding gradient descent in AI/ML - go gradient descent*, Jan. 2025. [Online]. Available: <https://gogradientdescent.com/gradient-descent-in-ai-ml/>.
- [10] *Cross Entropy Loss*. [Online]. Available: <https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html>.
- [11] *Positional encoding*. [Online]. Available: [https://d2l.ai/chapter\\_attention-mechanisms-and-transformers/self-attention-and-positional-encoding.html](https://d2l.ai/chapter_attention-mechanisms-and-transformers/self-attention-and-positional-encoding.html).
- [12] M. Saeed, “A gentle introduction to positional encoding in transformer models, part 1”, *MachineLearningMastery*, Jan. 2023. [Online]. Available: <https://www.cs.bu.edu/fac/snyder/cs505/PositionalEncodings.pdf>.
- [13] *Softmax*. [Online]. Available: <https://pytorch.org/docs/stable/generated/torch.nn.Softmax.html>.
- [14] 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, 173–191, Jul. 2018. DOI: 10.1016/j.physletb.2018.07.035. [Online]. Available: <https://arxiv.org/abs/1806.00425>.

- [15] W. Cui, “Quantization in quantum mechanics”, in *Relativistic Quantum Field Theory I*. 2023. [Online]. Available: [https://ocw.mit.edu/courses/8-323-relativistic-quantum-field-theory-i-spring-2023/mit8\\_323\\_s23\\_rec02.pdf](https://ocw.mit.edu/courses/8-323-relativistic-quantum-field-theory-i-spring-2023/mit8_323_s23_rec02.pdf).
- [16] ATLAS Collaboration, “Review of top quark mass measurements in CMS”, *Physics Reports*, Jan. 2025. DOI: 10.1016/j.physrep.2024.12.002. [Online]. Available: <https://arxiv.org/abs/2403.01313>.
- [17] *LHC Higgs Cross Section*. [Online]. Available: <https://twiki.cern.ch/twiki/bin/view/LHCPhysics/CrossSections2011>.
- [18] S. Frixione, V. Hirschi, D. Pagani, H.-s. Shao, and M. Zaro, “Electroweak and QCD corrections to top-pair hadroproduction in association with heavy bosons”, *Journal of High Energy Physics*, vol. 2015, no. 6, Jun. 2015. DOI: 10.1007/jhep06(2015)184. [Online]. Available: <https://www.osti.gov/pages/biblio/1182439>.
- [19] *Life of the Higgs Boson*. [Online]. Available: <https://cms.cern/news/life-higgs-boson>.
- [20] S. Y. Choi, J. S. Lee, and J. Park, “Decays of Higgs bosons in the Standard Model and beyond”, *Progress in Particle and Nuclear Physics*, vol. 120, p. 103880, May 2021. DOI: 10.1016/j.ppnp.2021.103880. [Online]. Available: <https://arxiv.org/abs/2101.12435>.
- [21] H. Ono, “Higgs branching ratio measurements at the ILC”, *Proceedings of the 12th Asia Pacific Physics Conference (APPC12)*, Mar. 2014. DOI: 10.7566/jpscp.1.013005. [Online]. Available: <https://doi.org/10.7566/jpscp.1.013005>.
- [22] *About CERN*, Apr. 2025. [Online]. Available: <https://home.cern/about>.
- [23] V. Kain, J. Wenninger, R. Schmidt, CERN, A. Siemko, and R. Jones, *SAFE MACHINE PARAMETERS*, E. Carlier, B. Goddard, J. Serrano, *et al.*, Eds. Oct. 2009. [Online]. Available: <https://ab-div-bdi-bl-blm.web.cern.ch/Specification/LHC-CI-ES-0004-10-00.pdf>.
- [24] *A transformative leap in physics: ATLAS results from LHC Run 2*, Apr. 2025. [Online]. Available: <https://atlas.cern/Updates/Feature/Run-2-Physics>.
- [25] ATLAS Collaboration, *The ATLAS Detector*. [Online]. Available: <https://atlas.cern/Discover/Detector>.
- [26] EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH, *Physics of Tau Leptons*. Sep. 1995. [Online]. Available: <https://cds.cern.ch/record/290480/files/ppe-95-147.pdf>.
- [27] *About Geant4*. [Online]. Available: <https://geant4.web.cern.ch/about/>.
- [28] ATLAS Collaboration, “The ATLAS simulation infrastructure”, *The European Physical Journal C*, vol. 70, no. 3, 823–874, Sep. 2010. DOI: 10.1140/epjc/s10052-010-1429-9. [Online]. Available: <https://doi.org/10.1140/epjc/s10052-010-1429-9>.
- [29] *Unlocking Grokking: a Comparative study of Weight Initialization Strategies in Transformer Models*. 2022. [Online]. Available: [https://sakana.ai/assets/ai-scientist/weight\\_initialization\\_grokking.pdf](https://sakana.ai/assets/ai-scientist/weight_initialization_grokking.pdf).
- [30] *XGBoost Documentation*. [Online]. Available: [https://xgboost.readthedocs.io/en/release\\_3.0.0/](https://xgboost.readthedocs.io/en/release_3.0.0/).