Karlsruher Institut für Technologie

ETP-KA/2025-3

Applications of modern machine learning methods in high-energy physics at the example of the CMS experiment

Zur Erlangung des akademischen Grades eines

DOKTORS DER NATURWISSENSCHAFTEN
(Dr. rer. nat.)

von der KIT-Fakultät für Physik des
Karlsruher Instituts für Technologie (KIT)

genehmigte

DISSERTATION

von

M.Sc. Lars Daniel Sowa

aus Karlsruhe

Tag der mündlichen Prüfung:	30. Mai 2025
Referent:	Prof. Dr. Markus Klute
Korreferent:	Priv.-Doz. Dr. Roger Wolf

Erklärung der selbstständigen Anfertigung der Dissertationsschrift

Hiermit erkläre ich, dass ich diese Dissertation selbstständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht habe, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde. Ich versichere außerdem, dass ich diese Dissertation nur in diesem und keinem anderen Promotionsverfahren eingereicht habe und dass diesem Promotionsverfahren keine endgültig gescheiterten Promotionsverfahren vorausgegangen sind.

Im Rahmen dieser Dissertation wurden die von künstlichen Intelligenzen gestützten Werkzeuge *Grammarly*[1], *ChatGPT*[2] und *GitHub Copilot*[3] eingesetzt, um Texte hinsichtlich Grammatik und Stil zu optimieren sowie bei der Entwicklung von Programmcode zu unterstützen. Sämtliche vorgeschlagenen Änderungen wurden von mir überprüft.

Karlsruhe, 23.04.2025

. . . .
(Lars Daniel Sowa)

[1]`https://www.grammarly.com`
[2]`https://chat.openai.com`
[3]`https://github.com`

Contents

Introduction

The Large Hadron Collider (LHC) located at CERN, Geneva, produces proton-proton (pp) collisions at a center-of-mass energy of up to 13.6 TeV. With a record-breaking integrated luminosity of 163 fb^{-1} provided from 2015 to 2018 [1], the LHC marks the forefront of global data production in high-energy physics (HEP). The Compact Muon Solenoid (CMS) Collaboration uses a cutting-edge particle detector to record this exceptional amount of information. These collisions are nearly unbiased, and the resulting data can be viewed as a realization of genuinely independent and identically distributed data. In consequence, HEP is, more than any other field of science, a statistics-dominated field in which each collision represents a statistical event, the outcome of a random experiment. In this environment, the laws of statistics are fully applicable and dozens of discoveries have proven this with relative precision levels up to 10^{-3} [2]. This has led to the standard model (SM) of particle physics, one of the most tested and robust theories in history. Comparisons of the SM with experiments are usually pursued in the form of likelihood-based hypothesis tests, and parameters of the SM are measured based on the maximum likelihood principle. The CMS experiment utilizes state-of-the-art statistical methods to study the most fundamental properties of particles at the highest precision possible.

Over the past decades, data analysis techniques have evolved from traditional cut-based methods to classical machine learning (ML) algorithms such as Boosted Decision Trees. These methods have proven instrumental in landmark discoveries, such as in the discovery of the Higgs boson in 2012 [3, 4].

With the exponential increase in computing power in recent years, the deployment of significantly more capable ML models has become available. Today, ML is about to become deeply embedded in HEP workflows and is employed in analyses, simulations, and calibration tasks. The recent Nobel Prize in Physics, awarded for contributions related to ML, underscores the growing significance of ML in shaping the outcomes and progress of modern scientific research [5].

As the utilization and the complexity of ML models grow, so does the need for their control: high performance alone does not guarantee reliability. Model reasoning and validation are crucial steps in conducting sustainable and robust scientific progress. Particle physics may play an essential role in this respect, providing a unique and rigorous benchmark for all kinds of ML algorithms combined with a long tradition in the application of statistical methods and a highly evolved and precise underlying theoretical foundation.

In the course of this thesis, the Taylor Coefficient Analysis (TCA), an analytical and gradient-based approach for neural network (NN) reasoning introduced at Karlsruhe Institute of Technology [6], has been further developed. Investigating the impact of input features on the NN output, the TCA provides interpretable insights into how the NN derives its prediction. Its application to a simple multilayer perceptron (MLP) and a fundamentally different graph neural network (GNN) architecture is tested for consistency in how both models respond to physics-motivated key features. Furthermore, this is connected to a full-scale momentum correction of particle clusters, called jets, based on a generative normalizing flow (NF) model to access the implied probability density function of the per-jet momentum. This is used to give a corrected momentum estimate in combination with a per-jet momentum resolution. The TCA is applied to the NF model as a proof of the versatility of the TCA method.

In the following, Part I explains fundamental concepts of ML to understand the utilized methods in this thesis. In Part II, theoretical foundations of the SM as well as the experimental setup for data collection with the CMS experiment are explained. Part III gives an introduction to the TCA including demonstrations on a toy dataset and simulated pp events. Finally, the momentum regression using NFs is discussed in Part IV.

Part I.

Machine learning

1. Neural networks

While neural networks (NNs) are playing an essential role in modern artificial intelligence (AI) technologies, their roots predate the 1950s, when Frank Rosenblatt introduced the concept of an artificial neuron [7] (perceptron), a pioneering idea in AI. The perceptron was designed to generate an output from a given set of input signals, mimicking the functionality of biological neurons.

During the 1960s, AIs gained momentum through attempts using them for pattern recognition. However, the models of that time could not achieve significant breakthroughs due to the limited computing power at hand. This led to criticism of the field, as progress seemed to stagnate [8], and the research field entered what became known as the "AI winter" of the 1980s.

With the advantages of semiconductor manufacturing, which unlocked new computational power, the interest in AI rewoke. Although AI researchers were at the forefront of pushing these new limits, the rise of graphics processing units (GPUs) in the late 2000s marked the beginning of a deep learning revolution, which led to widespread applications across many research fields and industries, enabling the analysis of data patterns that traditional methods could not uncover.

A pivotal moment was reached in 2017 when the paper "Attention is All You Need" [9] by Google drew global awareness. Its implementation in large language models (LLMs) demonstrated the extraordinary capabilities of AI, sparking widespread excitement outside the field.

Today, AI is deeply embedded in everyday life, from tools like AI chatbots and autonomous driving to further pushing or conceptually enabling cutting-edge research in fields of natural science [10]. The dramatic acceleration of breakthroughs in research enabled by AI underscores its importance to modern science. This is further highlighted by the Nobel Prize in physics in 2024 [5], which honored the application of AI in scientific discovery.

This chapter is designed to give a general introduction to ML and NNs, providing a foundation for advanced ML algorithms such as graph neural networks (GNNs) and normalizing flows (NFs). Throughout this thesis, ML concepts are explained using likelihood-based approaches, as detailed in Section 1.1. multilayer perceptron (MLP)s are explained in Section 1.3 and their ability for universal approximations in Section 1.4. After that, the most fundamental training concepts, i.e., loss functions, backpropagation, and regularization methods, are discussed in Section 1.5, 1.6, and 1.7 respectively.

1.1. Likelihood

The likelihood is a central quantity in statistical modeling and thus also in machine learning. It is used to quantify how well a model $\Omega_{\hat{y}}$ aligns with a given dataset $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$ of size N. Possible applications of Likelihoods are parameter estimation or hypothesis testing. If X is independent and identically distributed (i.i.d.), the likelihood reads as

$$\mathcal{L}(\boldsymbol{\theta}) = \prod_{i}^{N} P(\mathbf{x}_i \mid \boldsymbol{\theta}). \tag{1.1}$$

where $P(\mathbf{x} \mid \boldsymbol{\theta})$ describes the probability to observe $\mathbf{x}$ for given model parametrized by $\boldsymbol{\theta}$. Maximizing $\mathcal{L}(\boldsymbol{\theta})$ w.r.t. $\boldsymbol{\theta}$, results in a parameter set $\hat{\boldsymbol{\theta}}$ that provides the best description for X. Therefore, the probability density function (PDF) of a given event $\mathbf{x}$ must be analytically defined to describe the likelihood.

For numerical stability and general applicability, a common transformation of $P(\mathbf{x} \mid \boldsymbol{\theta})$ is to take the negative logarithmic likelihood

$$-\log \mathcal{L}(\boldsymbol{\theta}) = -\sum_{i}^{N} \log P(\mathbf{x} \mid \boldsymbol{\theta}). \tag{1.2}$$

This is motivated by practical considerations: the product in Eq. (1.1) may lead to numerical issues due to very small values. In addition, most optimization algorithms are implemented for minimization instead of maximization. Given the monotonicity of $\log(\cdot)$, maximizing Eq. (1.1) is equivalent to minimizing Eq. (1.2). This concept is called maximum likelihood estimation (MLE)

$$\hat{\boldsymbol{\theta}} = \arg\max_{\theta} \mathcal{L}(\boldsymbol{\theta}), \tag{1.3}$$

$$\hat{\boldsymbol{\theta}} = \arg\min_{\theta}(-\log \mathcal{L}(\boldsymbol{\theta})). \tag{1.4}$$

Particularly for the asymptotic limit of $N \to \infty$, the MLE has advantages over other methods of parameter estimation. MLE is a consistent estimator for $\boldsymbol{\theta}$ and becomes asymptotically unbiased for $N \to \infty$. Moreover, in this asymptotic regime, the distribution of the estimated parameters $\hat{\boldsymbol{\theta}}$ approaches a normal distribution centered around the true parameter $\boldsymbol{\theta}$. However, MLE does not guarantee non-convex problem-solving in the presence of local minima.

1.2. Likelihood ratio

The likelihood $\mathcal{L}$ expresses how well a model $\Omega_{\hat{y}}$ $(\boldsymbol{\theta})$ within a given set of parameters $\boldsymbol{\theta}$ fits a given dataset X. Introducing the likelihood ratio

$$\lambda_{\mathrm{LR}} = \frac{\mathcal{L}(\boldsymbol{\theta}_1)}{\mathcal{L}(\boldsymbol{\theta}_0)}, \tag{1.5}$$

two different models are compared to decide whether X fits more likely under model $\Omega_0(\boldsymbol{\theta}_0)$ (null hypothesis) or $\Omega_1(\boldsymbol{\theta}_1)$ (alternative hypothesis). It holds that

- $\lambda_{\mathrm{LR}} \gg 1$ favors $\Omega_1(\boldsymbol{\theta}_1)$
- $\lambda_{\mathrm{LR}} \ll 1$ favors $\Omega_0(\boldsymbol{\theta}_0)$
- $\lambda_{\mathrm{LR}} \approx 1$ is considered inconclusive.

For implementations, λ_{LR} is commonly extended to $\log \lambda_{\mathrm{LR}}$ for the same reasons discussed in Section 1.1.

The Neyman-Pearson Lemma [11] underlines the discrimination power of λ_{LR} as the most powerful test to distinguish two hypotheses. Therefore, λ_{LR} is a commonly used test statistic for hypothesis testing and model comparisons in ML.

1.3. Neurons and multilayer perceptrons

A neuron is the fundamental building block of all NNs. Its core idea is to obtain a set of n input parameters $\mathbf{x}$ and to transform them to an output y. This is done by a linear transformation using $\boldsymbol{\omega}, b \in \boldsymbol{\theta}$

$$y = f\left(\sum_{i=1}^{n} \omega_i x_i + b\right) = f(\boldsymbol{\omega}^{\mathrm{t}}\mathbf{x} + b), \tag{1.6}$$

with trainable parameters $\boldsymbol{\omega}$. The activation function f introduces non-linearity to the neuron, allowing the approximation of more complex functions. This non-linearity is essential, as it allows the modeling of arbitrary, non-linear relationships by arranging multiple neurons to an NN. Common choices for f are the ReLu function [12]

$$\mathrm{ReLU}(x) = \max(0, x), \tag{1.7}$$

its modified version Leaky ReLu [13],

$$\text{Leaky ReLU}(x) = \begin{cases} x, & \text{if } x > 0 \\ \beta x, & \text{if } x \leq 0, \end{cases} \tag{1.8}$$

or the hyperbolic tangent

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \tag{1.9}$$

where β is a selectable parameter for the Leaky ReLU. The tanh is continuously differentiable, which enables the computation of higher-order derivatives. However, it exhibits asymptotic behavior: large values of the linear transformation in Eq. (1.6) can result in vanishing gradients during the optimization process. This is especially prominent if many neurons are arranged in a sequence.

The multilayer perceptron (MLP) is one of the most commonly used NN architectures to arrange neurons. Here, the neurons are stacked in L layers, each layer l consisting of n_l neurons. The input values are given by the previous layer $l-1$, accordingly, the layer output reads as

$$\mathbf{h}^{(l)} = f^{(l)}\left(\boldsymbol{\omega}^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}\right), \quad \forall l = 1, \dots, L \tag{1.10}$$

where $\boldsymbol{\omega}^{(l)} \in \mathbb{R}^{n_l \times m}$ represents a matrix of trainable weights, $\mathbf{h}^{(l)} \in \mathbb{R}^{n_l}$ the output, and $\mathbf{b} \in \mathbb{R}^{n_l}$ the bias of the l^{th} layer. The activation function of layer l is written as $f^{(l)}$. There are two layers of special meaning:

- $\mathbf{h}^{(0)}$ is the input layer and consists of the input features $\mathbf{x}$
- $\mathbf{h}^{(L)}$ is the output layer and corresponds to the model output, denoted as $\hat{\mathbf{y}}$.

Each layer $\mathbf{h}^{(l)}$ in between is referred to as a hidden layer. MLPs with one or more hidden layers are referred to as deep neural network (DNN). For regression tasks, no function $f^{(L)}$ is applied avoiding a constrained model output

$$\Omega_{\hat{y}} : \mathbb{R}^{n_L} \longrightarrow \mathbb{R}^{n_L}, \tag{1.11}$$

assuming n_L output values. For classification tasks, the output is restricted to values between zero and one

$$\Omega_{\hat{y}} : \mathbb{R}^{n_L} \longrightarrow [0,1]^{n_L}, \tag{1.12}$$

allowing for a probability-like interpretation. For $n_L = 1$, the sigmoid function

$$\varphi(x) = \frac{1}{1+e^{-x}}, \tag{1.13}$$

is a common choice. For $n_L > 1$, the softmax function

$$\text{Softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n_L} e^{x_j}}, \quad i = 1, \dots, n_L, \tag{1.14}$$

can be utilized to introduce a normalization for all output values.

For an MLP architecture based on Eq. (1.10), three hyperparameters have to be decided upon:

- the choice of activation functions
- the number of layers L
- and the number of neurons n_l per layer l.

For tasks corresponding to the complexity of typical high-energy physics problems, the number of layers is usually in the range $2 < L < 5$, with the number of neurons typically ranging from $50 < n_l < 500$.

1.4. Universal approximation theorem

The universal approximation theorem (UAT) [14, 15] is one of the most fundamental results in NN theory. The formal statement of the UAT is that $\Omega_{\hat{y}}$ can approximate any continuous function $g : [a, b] \mapsto \mathbb{R}^n$, where $[a, b] \subseteq \mathbb{R}$, with arbitrary precision. Various formulations of the UAT exist, some of which impose restrictions on the number of hidden layers, allowing an arbitrary number of neurons for $\Omega_{\hat{y}}$, while others constrain the number of neurons, permitting the hidden layers to vary in size. However, there are also UAT proofs for more complex architectures like for convolutional neural networks (CNNs) [16, 17] or GNNs [18, 19].

In the following, an illustration is provided to understand why MLPs are universal approximators. For this, $\Omega_{\hat{y}}$ is selected to be a simple MLP with scalar input $h^{(0)}$ and scalar output $\Omega_{\hat{y}}(\cdot)$. The architecture is chosen to have one hidden layer $L = \{0, 1, 2\}$ with two neurons

$$\mathbf{h}^{(1)} = \left(h_1^{(1)}, h_2^{(1)}\right)^T . \tag{1.15}$$

The full equation for $\Omega_{\hat{y}}$ (x) reads as weighted sum of the components of $\mathbf{h}^{(1)}$

$$\Omega_{\hat{y}}(x) = \omega_1^{(2)} \underbrace{\varphi\left(\omega_1^{(1)} x + b_1^{(1)}\right)}_{h_1^{(1)}} + \omega_2^{(2)} \underbrace{\varphi\left(\omega_2^{(1)} x + b_2^{(1)}\right)}_{h_2^{(1)}} + b^{(2)}. \tag{1.16}$$

The activation function is chosen to be a sigmoid function $f^{(1)} = \varphi$, and the identity function for the final layer $f^{(2)}$. The function $g(\cdot)$ to approximate is a piecewise step function with discontinuities at $x = 0$ and $x = 1$

$$g(x) = \begin{cases} 0, & \text{if } x < 0 \\ 1, & \text{if } 0 \leq x < 1 \\ 2, & \text{if } x \geq 1 \end{cases} . \tag{1.17}$$

To approximate the step function in Eq. (1.17), the properties of Eq. (1.13) are used in the following way: due to the exponential function in the denominator of Eq. (1.13), the limits for φ are

$$\lim_{x \to \infty} \varphi(x) = 1 \quad \text{and} \tag{1.18}$$

$$\lim_{x \to -\infty} \varphi(x) = 0. \tag{1.19}$$

This behaviour can be used to model step functions based on φ. With Equation (1.16), this can be achieved by choosing the outputs of $h_1^{(1)}$ and $h_2^{(1)}$ sufficiently large to approximate one step in Eq. (1.17), respectively. A sufficiently large factor of 50 is chosen to account for inputs assumed to be of order one. For instance, the parameters in $h_1^{(1)}$ and $h_2^{(1)}$ can be set to

$$\omega_1^{(1)} = 50, \quad b_1^{(1)} = 0, \quad \omega_2^{(1)} = 50, \quad \text{and} \quad b_2^{(1)} = 50. \tag{1.20}$$

The output layer $\mathbf{h}^{(2)}$ determines the height of the step functions. To match Eq. (1.17), the parameters are set to

$$\omega_1^{(2)} = 2, \quad \omega_2^{(2)} = 1, \quad \text{and} \quad b^{(2)} = 0. \tag{1.21}$$

The final form of Eq. (1.16) is

$$\Omega_{\hat{y}}(x) = 2\varphi(50x) + \varphi(50(x-1)), \tag{1.22}$$

which approximates Eq. (1.17). By adding more neurons in the hidden layer, one can also extend this to arbitrary complex compositions of step functions. By using the Stone–Weierstrass theorem [20, 21], stating that arbitrarily many step functions can approximate each continuous function, the final conclusion can be made, that an MLP can approximate any arbitrary complex function $g(\cdot)$ assuming the required model complexity.

Although the UAT holds, it does not ensure that an NN training will find an optimal approximation for $g(\cdot)$. This can be due to computational limits regarding the NN size, a limited dataset, or a nonoptimal training setup.

1.5. Loss functions and optimization

The fundamental problem of all ML tasks is to approximate a function

$$\Omega(\mathbf{x}) : X \subseteq \mathbb{R}^n \mapsto Y \subseteq \mathbb{R}^m \quad \text{with} \quad X = \{\mathbf{x}_i \mid \mathbf{x}_i \in X, i = 1, \ldots, N\} \tag{1.23}$$

$$\text{and} \quad Y = \{y_i \mid y_i \in Y, i = 1, \ldots, N\}. \tag{1.24}$$

Further, it is distinguished between regression tasks with $Y \subseteq \mathbb{R}$ and classification tasks with $Y = \{1, \ldots, C\}$ for C discrete classes. For regressions, a calibrated mapping from X to Y is identified, while for classifications, a decision boundary to discriminate classes in Y is identified. Both tasks are fundamental for addressing HEP problems. To approximate Ω, a model $\Omega_{\hat{y}}(\boldsymbol{\theta}, \mathbf{x})$ with adjustable parameters $\boldsymbol{\theta}$ is selected. A mathematical function, characterizing the underlying task, has to be defined to measure the performance of $\Omega_{\hat{y}}$ in approximating Ω. Such an objective is called loss function L. Since the likelihood provides consistent and Gaussian estimates for $\hat{\boldsymbol{\theta}}$, it is an optimal candidate for L, which can be optimized by applying the MLE method described in Eq. (1.4).

For a regression task, the difference between $\Omega_{\hat{y}}(\cdot)$ and y (residual) is minimized. Following the central limit theorem, the residual distribution, i.e., the probability of y_i being approximated correctly by $\Omega_{\hat{y}}(\cdot) = \hat{y}$, can be described as a Gaussian distribution, leading to

$$P(y_i|\hat{y}_i, \sigma_N^2) = \frac{1}{\sqrt{2\pi\sigma_N^2}} \exp\left(-\frac{(y_i - \hat{y}_i)^2}{2\sigma_N^2}\right) \tag{1.25}$$

where $\sigma_{\mathcal{N}}$ corresponds to the standard deviation. This probability distribution can be used to construct $-\log \mathcal{L}$ for a set of N samples, leading to

$$-\log \mathcal{L}(\boldsymbol{\theta}|X) = -\sum_{i=1}^{N} \log P(y_i|\hat{y}_i, \sigma^2) \tag{1.26}$$

$$= \sum_{i=1}^{N} \left(\underbrace{\frac{1}{2}\log(2\pi\sigma_{\mathcal{N}}^2)}_{\text{const.}} + \frac{(y_i - \hat{y}_i)^2}{2\sigma_{\mathcal{N}}^2} \right). \tag{1.27}$$

Since $\sigma_{\mathcal{N}}$ is a fixed parameter, $-\log \mathcal{L}$ can be optimized by minimizing $(y_i - \hat{y}_i)^2$ or, being independent of the sample size, the mean squared error (MSE)

$$\text{MSE} = \frac{1}{N}\sum_{i}^{N}(y_i - \hat{y}_i)^2. \tag{1.28}$$

To obtain L for a classification task, the discrimination between two classes is considered. The probability P that a prediction $\hat{y}$ is correct, given its true class y, is given by a Bernoulli distribution

$$P(y_i \mid \hat{y}_i) = \hat{y}_i^{y_i}(1 - \hat{y}_i)^{1-y_i}. \tag{1.29}$$

Using the Bernoulli probability for a set of N samples leads to

$$-\log \mathcal{L}(\boldsymbol{\theta}|X) = -\sum_{i}^{N} \log P(y_i \mid \hat{y}_i) \tag{1.30}$$

$$= -\sum_{i=1}^{N} \left(y_i \log \hat{y}_i + (1 - y_i)\log(1 - \hat{y}_i)\right). \tag{1.31}$$

For being independent of the sample size, $-\log \mathcal{L}$ is divided by N, leading to the cross entropy (CE) H in its binary case

$$H(\hat{y}, y) = -\frac{1}{N}\sum_{i=1}^{N} \left(y_i \log \hat{y}_i + (1 - y_i)\log(1 - \hat{y}_i)\right). \tag{1.32}$$

Since this thesis does not apply the CE to problems with more than two classes, all references to CE refer to the binary case. By following the likelihood approach, as demonstrated for regression and classification tasks, the loss function L can be constructed for advanced NN architectures, such as for generative models. By applying the MLE principle on L, $\hat{\boldsymbol{\theta}}$ can be identified by

$$\hat{\theta} = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{N} L(\Omega_{\hat{y}}(\boldsymbol{\theta}, \mathbf{x}_i), y). \tag{1.33}$$

This minimization can be performed using the stochastic gradient descent (SGD) algorithm, in which the parameters for $\boldsymbol{\theta}$ are initially guessed and iteratively updated by calculating the gradient of L and updating $\boldsymbol{\theta}$ in the direction that minimizes it iteratively. The k^{th} update step of this procedure reads

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta \nabla_{\boldsymbol{\theta}} L(\Omega_{\hat{y}}(\boldsymbol{\theta}_k, \mathbf{x}_i), y_i), \tag{1.34}$$

where η is the learning rate and determines the step size to be used to update $\boldsymbol{\theta}_k$ and $\nabla_{\boldsymbol{\theta}} L$ is the gradient of L w.r.t. $\boldsymbol{\theta}$.

While Eq. (1.34) is written for one event i, SGD uses a stochastic subsample of the full dataset, referred to as a minibatch. The training state is evaluated after a certain number of iterations, typically when all data samples have been processed once. This iteration is then referred to as one training epoch.

Since computing Eq. (1.34) separately for each weight in $\Omega_{\hat{y}}$ is computationally very expensive, a more efficient approach is to determine the gradients layer-wise. The corresponding derivation is referred to as the error term

$$\boldsymbol{\delta}^{(l)} = \frac{\mathrm{d}L}{\mathrm{d}\mathbf{z}^{(l)}} \quad \text{with } \mathbf{z}^{(l)} = \boldsymbol{\omega}^{(l)} f(\mathbf{z}^{(l-1)}) + \mathbf{b}^{(l)}, \tag{1.35}$$

for updates in layer l. Since

$$\frac{\mathrm{d}L}{\mathrm{d}\mathbf{z}^{(l)}} = \frac{\mathrm{d}L}{\mathrm{d}\mathbf{z}^{(l+1)}} \frac{\mathrm{d}\mathbf{z}^{(l+1)}}{\mathrm{d}\mathbf{z}^{(l)}} = \frac{\mathrm{d}L}{\mathrm{d}\mathbf{z}^{(l+1)}} \boldsymbol{\omega}^{(l+1)} f'(\mathbf{z}^{(l)}), \tag{1.36}$$

the error propagation through the network can be performed iteratively

$$\boldsymbol{\delta}^{(l)} = \frac{\partial L}{\partial \mathbf{z}^{(l)}} = \boldsymbol{\delta}^{(l+1)} \boldsymbol{\omega}^{(l+1)} f'(\mathbf{z}^{(l)}). \tag{1.37}$$

This concept is known as backpropagation [22] and is a standard procedure in modern frameworks for gradient calculations, such as `PyTorch` [23] or `TensorFlow` [24].

Since SGD is a basic approach of implementing an optimizer, more advanced optimizers extend the principle of SGD. One option is to use momentum terms, which include previous gradient updates $\boldsymbol{\theta}_{k-1}$, improving robustness against statistical fluctuations. Another extension is to adapt η during the training, which allows for larger (slower) minimization steps in steep (flat) areas of L. Examples of such implementations are listed below. Some optimizers utilize ϵ, which corresponds to a small number, ensuring computational stability.

MSGD [25] extends SGD by adding a momentum term weighted by μ, which helps to smooth the update by incorporating information of past gradients

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta \nabla_{\boldsymbol{\theta}} L + \mu \Delta \boldsymbol{\theta}.$$

AdaGrad [26] adapts η for each parameter θ^j individually by scaling it inversely with the square root of summed past gradients h^j

$$\theta^j_{k+1} = \theta^j_k - \frac{\eta}{\sqrt{h^j} + \epsilon} \nabla_{\theta^j} L.$$

AdaDelta [27] further removes the need for a manually chosen start value for η by using a moving average over past updates. Using $E[a] = \rho a + (1-\rho)a$ with the decay rate ρ leads to:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \frac{\sqrt{E[\Delta\boldsymbol{\theta}_k^2] + \epsilon}}{\sqrt{E[\nabla_{\boldsymbol{\theta}_k} L^2] + \epsilon}} \nabla_{\boldsymbol{\theta}_k} L.$$

Adam [28] combines momentum and adaptive learning rates by using both the first moment $\hat{m}_j$ (mean) of the gradients for the direction of the update and second moment $\hat{v}_j$ (variance) to reduce the step size if $\nabla_{\boldsymbol{\theta}_k}$ differs from $\nabla_{\boldsymbol{\theta}_{k-1}}$:

$$\begin{aligned} \boldsymbol{\theta}_{k+1} &= \boldsymbol{\theta}_k - \frac{\eta}{\sqrt{\hat{v}_k} + \epsilon}\hat{m}_k \\ \text{with } \hat{m}_k &= \beta_1 \hat{m}_{k-1} + (1-\beta_1)\nabla_{\boldsymbol{\theta}_k} L \\ \text{and } \hat{v}_k &= \beta_2 \hat{v}_{k-1} + (1-\beta_2)\nabla_{\boldsymbol{\theta}_k} L^2. \end{aligned}$$

and coefficients β_i, which are set to$\beta_1 = 0.9$ and $\beta_2 = 0.999$ in this work.

A comparison of the minimization behavior of SGD and the above-listed algorithms, provided by Ref. [29], is shown in Fig. 1.1. The Himmelblau function [30] is used as a loss function

$$L(x_0, x_1) = (x_0^2 + x_1 - 11)^2 + (x_0 + x_1^2 - 7)^2, \tag{1.38}$$

resulting in the plane visualized in 3D (upper) and 2D (lower) and revealing four minima. In this setup, the parameters x_0 and x_1 correspond to model parameters $\boldsymbol{\theta} = (x_0, x_1)^{\mathrm{T}}$ which are minimized within L. Starting at a common point in L, $\boldsymbol{\theta}$ is updated for 100 steps using the described optimizers. The figure presents the resulting trajectories, illustrating the characteristic behavior of each algorithm in terms of step size and direction.

Comparing the minimization trajectories in Fig. 1.1, it is noticeable that not all setups converge into the same minimum, a commonly faced problem for NN weight optimization. Especially in high-dimensional and complex parameter spaces, there may be several non-global minima or, in general, non-convex forms of L. Therefore, an optimizer should, on the one hand, be able to escape local minima by significant update steps of $\boldsymbol{\theta}$ and, on the other hand, determine the (global) minimum exactly enough using small update steps. Several optimizers solve this issue by using adaptive learning rates. Lastly, an optimizer needs to be performant in its convergence time, which significantly impacts the training of large NNs.

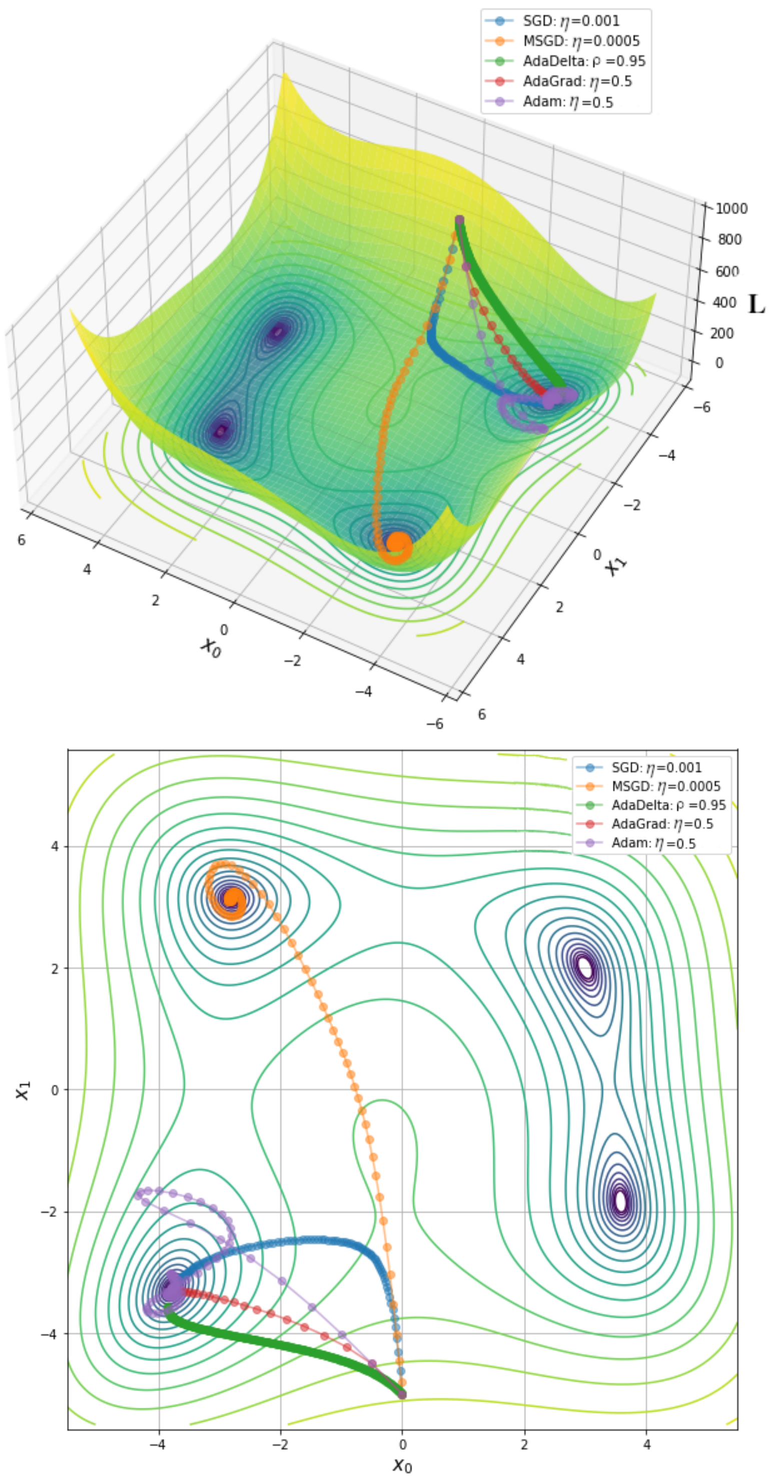


Figure 1.1.: Comparison of different optimizers applied on the Himmelblau function [30]. Shown are the SGD (blue), MSGD (orange), AdaDelta (green), AdaGrad (red), and ADAM (purple) optimizers. Each optimizer is applied with an individual η respectively. Adapted from Ref. [29].

Figure 1.1 illustrates the difference of extending SGD with momentum to form MSGD. The study shows that MSGD takes larger update steps in steep regions of L and smaller steps in flatter areas, ultimately converging to a different minimum that is further away and is not the most immediate solution. Other optimizers with a mechanism to scale η exhibit similar behavior in step size. While SGD, AdaDelta, and AdaGrad converge directly towards the minimum, adaptive moment estimation (ADAM) is the only algorithm among them that tends to overshoot the minimum. This behavior may allow ADAM to escape shallow local minima, if present. However, ADAM requires more update steps to converge to a minimum, particularly when compared to AdaGrad, which also effectively adapts its step size in steep regions of L.

In conclusion, the ADAM algorithm shows a reasonable tradeoff between escaping minima in case of non-convex problems and fast convergence to a given minimum. Due to its robustness, ADAM is among the most used optimizers for estimating $\hat{\boldsymbol{\theta}}$ and is employed throughout this work.

1.6. Overfitting and early stopping

During the training process of an NN, it is possible for the model to memorize the training dataset X^{train} to such an extent that its learned mapping fails to generalize to the dataset it is tested on X^{test}. After the training of $\Omega_{\hat{y}}$, this can be stated as:

$$\Omega_{\hat{y}} : X^{\text{train}} \mapsto Y^{\text{train}} \quad \text{but} \quad \Omega_{\hat{y}} : X^{\text{test}} \not\mapsto Y^{\text{test}}. \tag{1.39}$$

When the model fails to generalize across both datasets, it is referred to as overfitting. In machine learning tasks, overfitting is generally caused by one or more of the following factors:

- insufficient data
- number of model parameters exceeds the complexity of the problem
- inherently noisy or biased data.

These factors typically result in numerous overfitted solutions to the optimization problem, rendering it ill-posed [31]. To detect overfitting, the dataset is divided into three subsets: a training set used to optimize the parameters $\hat{\boldsymbol{\theta}}$, a validation set employed during training to monitor overfitting, and a test set for independent performance evaluation of $\Omega_{\hat{y}}$. A characteristic sign of overfitting is a continuous decrease in training loss accompanied by a stagnation or increase in validation loss, indicating that the model overfits to noise in the training data rather than capturing generalizable patterns.

Therefore, the training process of a NN should be stopped when L determined on the validation dataset starts increasing. A common implementation of this approach is to save the model parameters, validation, and training losses throughout training. If the

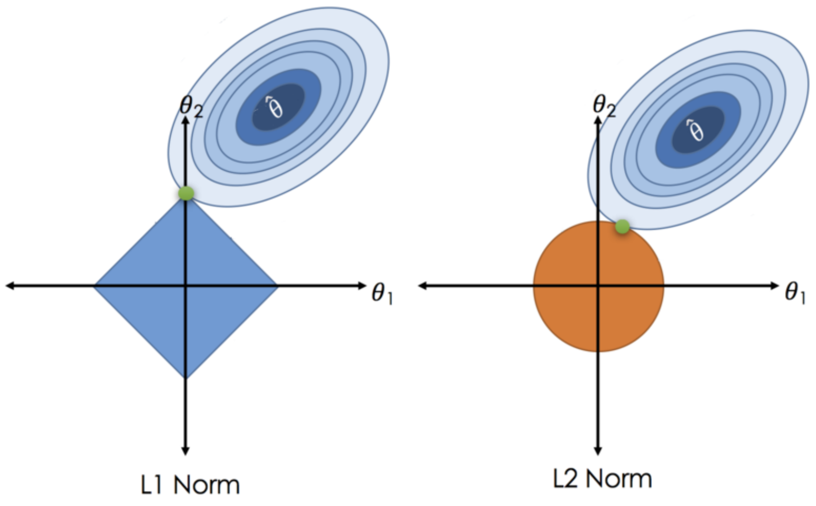


Figure 1.2.: Illustrative effects of applying the L_1 norm (left) and L_2 norm (right) as a weight regularization method in a two-dimensional parameter space. While the L_1 norm prefers solutions where single parameters are set to zero (blue diamond), the L_2 norm acts isotropically in the parameter space (orange circle). Both applications enforce small values for $\hat{\boldsymbol{\theta}}$. Adapted from Ref. [33, 34]

validation loss does not improve for a given number of epochs, the training is stopped and the best-performing parameters are chosen for the final model. This method is called Early Stopping [32] and is typically employed with 8 to 12 training epochs of patience.

In addition to Early Stopping, regularization techniques are specifically designed to prevent overfitting. Several regularization methods have been developed to address overfitting, the most important methods are described in the next section.

1.7. Regularization

This section explains a collection of methods to address the phenomenon of overfitting and general methods with a stabilizing effect on the training. Weight regularization methods suppress large elements in $\boldsymbol{\theta}$ to mitigate the overreliance on single features θ_i. As a result, $\Omega_{\hat{y}}$ is hindered in learning too detailed properties of the training dataset and is forced to generalize so that $\Omega_{\hat{y}}$ applies not only to the training but also to the validation dataset.

A straightforward method is to penalize large values of $\boldsymbol{\theta}$ directly. One method is L_2 regularization [35], also known as weight decay. Here, the L_2 norm of θ_i is incorporated as a constraint to the optimization problem

$$L' = L + \lambda \sum_i \|\theta_i\|_2, \tag{1.40}$$

where λ introduces a Lagrange multiplier, determining the impact on L'. In the parameter space of possible values of $\boldsymbol{\theta}$, the L_2 norm forces $\hat{\boldsymbol{\theta}}$ on a spherical plane around the origin. This is illustrated in Fig. 1.2 (right), where the choice of $\boldsymbol{\theta}$ without regularisation is denoted as $\hat{\boldsymbol{\theta}}$ and the L_2 regularization is displayed as an orange circle in an exemplary two-dimensional parameter space.

analogue to the L_2 regularization, L_1 regularization [36] enforces small $\boldsymbol{\theta}$ by constraining their L_1 norm, reading as

$$L' = L + \lambda \sum_i \|\theta_i\|_1. \tag{1.41}$$

In a two-dimensional example, applying the L_1 norm corresponds to minimizing $\boldsymbol{\theta}$ to a diamond shape around the origin, while the edges of the diamond lie on the axes of the coordinate system, as illustrated in Fig. 1.2 (left).

The L_2 norm enforces small values for $\boldsymbol{\theta}$ while being isotropic in the parameter space, the L_1 norm prefers to enforce $\boldsymbol{\theta}$ to be small and can set elements in $\boldsymbol{\theta}$ to zero. Therefore, the L_1 norm effectively reduces the parameter space. In some applications, L_1 is also used to shrink the input space, as in Ref. [36].

Dropout [37] is a third regularization method that deactivates a certain percentage of neurons $\mathrm{h}_i^{(l)}$ of an NN model during the training. This prohibits $\Omega_{\hat{y}}$ from relying on only a few weights with high impact on $\Omega_{\hat{y}}(\cdot)$. Instead, the weights are enforced to be more equally distributed, thus relying on the full range of parameters. Since the neuron deactivations are random in each update step, this procedure can be interpreted as equivalent to an ensemble training of models with different architectures.

Weight regularization techniques operate directly on the model parameters $\boldsymbol{\theta}$. An alternative is to aim for a normalization of the layer output $h_i^{(l)}$ to provide inputs at a suitable scale for the following layer $h_i^{(l+1)}$. This indirectly reduces the need for large weights in $h_i^{(l+1)}$. The minimization of L is performed in minibatches. Hence, the utilized data can be represented as matrices, where row i indicates the i^{th} feature in the layer and column j corresponds to the j^{th} event of the minibatch. To explain batch norm (BN) and layer norm (LN), the output of a layer $h_i^{(l)}$ will in this section, and only in this section, be extended by the batch dimension j to $h_{ij}^{(l)}$.

To scale the entries in $h_{ij}^{(l)}$ to a suitable size for the next layer, it can be operated either along the batch dimension j or along the layer dimension i.

These two techniques are known as batch norm (BN) [39] and layer norm (LN) [40] and both are illustrated in Fig. 1.3. BN utilizes mean μ_J and variance σ_J^2 for each feature to standardize $h_{ij}^{(l)}$, as illustrated in Fig. 1.3. After transforming $h_{ij}^{(l)}$ to a standardized parameter space, the inputs are scaled and shifted by learnable parameters γ_J and β_J, respectively, allowing for fine-tuning of the transformed layer output $\hat{h}_{ij}^{(l)}$ at training time.

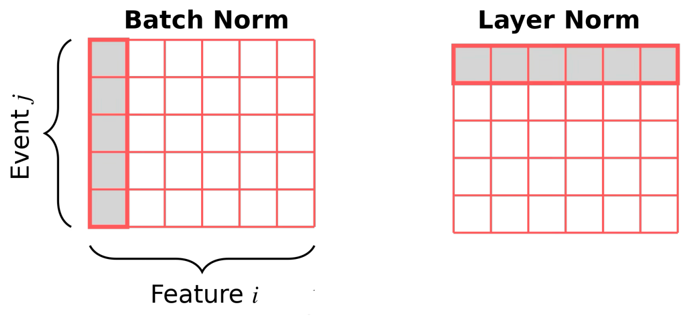


Figure 1.3.: Illustration of BN in the left section and LN in the right. The two-dimensional layer output $h_{ij}^{(l)}$ is shown as a red grid with batch dimension j and feature dimension i. For each method, μ and σ are determined for the grey elements. Adapted from Ref. [38]

The BN reads as

$$\hat{h}_{ij} = \frac{h_{ij} - \mu_J}{\sqrt{\sigma_J^2 + \epsilon}} \gamma_J + \beta_J, \tag{1.42}$$

where a small number ϵ is utilized to avoid zero divisions. The procedure of LN is analogue to the BN but applying it on the layer dimension i:

$$\hat{h}_{ij} = \frac{h_{ij} - \mu_I}{\sqrt{\sigma_I^2 + \epsilon}} \gamma_I + \beta_I. \tag{1.43}$$

For both methods, the mean and standard deviation are computed batch-wise during the training process. In the inference step, they are determined using the weighted mean of the whole training phase. This is particularly important for BN, as computing μ_J and σ_J is not feasible when evaluating individual events.

2. Graph neural networks

Since MLPs are limited to grid-like or vectorial data, they lack in flexibility to represent more complex data structures. Graph theory, dating back to the work of Leonhard Euler [41] in the 18$^{\text{th}}$ century, provides a natural framework for modeling relationships between objects. In HEP, analyses rely on objects, such as reconstructed jets in the detector. The number of these objects may vary between events, and they do not follow an inherent grid-like structure. GNNs are well-suited to address such abstract structures within a mathematical graph, as this reflects their natural form of representation. In Section 7.2.3, the advantages of a graph architecture are employed for an event classification based on reconstructed jets, missing transverse momentum, and a lepton in events from pp collisions. Today, GNN architectures such as exemplary highlighted in Ref. [42, 43], are considered state-of-the-art and are actively applied in HEP [44].

This chapter provides an introduction to graph theory in Section 2.1. Mechanisms for updating GNN properties are discussed in Section 2.2. These sections form the foundation for the studies presented in Part III.

2.1. Introduction to graph theory

Objects are described by a GNN as a set of $|V|$ vertices $V = \{\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_{|V|}\}$ which are connected with edges $E = \{\mathbf{e}_{ij}, \mathbf{e}_{ik}, ...\}$ where $\mathbf{e}_{ij}$ indicates the connection between $\mathbf{v}_i$ and $\mathbf{v}_j$. While vertex $\mathbf{v}_i$ holds object-specific information, the edge vector $\mathbf{e}_{ij}$ keeps relational information between $\mathbf{v}_i$ and $\mathbf{v}_j$. The combination of V and E is called a graph $G = (V, E)$. Information processing is realized by iteratively updating representations of $\mathbf{v}_i$ and $\mathbf{e}_{ij}$ based on the features of their connected components.

Various arrangements of $\mathbf{v}_i$ and $\mathbf{e}_{ij}$ allow for complex graphs, but in order to be part of G, $\mathbf{v}_i$ is required to be directly or indirectly connected to each other $\mathbf{v}_j$. An exemplary graph structure G is illustrated in Fig. 2.1, indicating vertices $\mathbf{v}_i$ as blue circles and edges $\mathbf{e}_{ij}$ as red lines. All $\mathbf{v}_j$ are at least indirectly connected with each other. For the edges, three types can be defined:

- Directed edges point from $\mathbf{v}_i$ to $\mathbf{v}_j$, which means that information from $\mathbf{v}_i$ is used to update $\mathbf{v}_j$. Directed edges are indicated with arrows in Fig. 2.1.
- Undirected edges transport information in both directions: from $\mathbf{v}_i$ to $\mathbf{v}_j$ and vice versa. This type of edge is illustrated without arrows in Fig. 2.1.

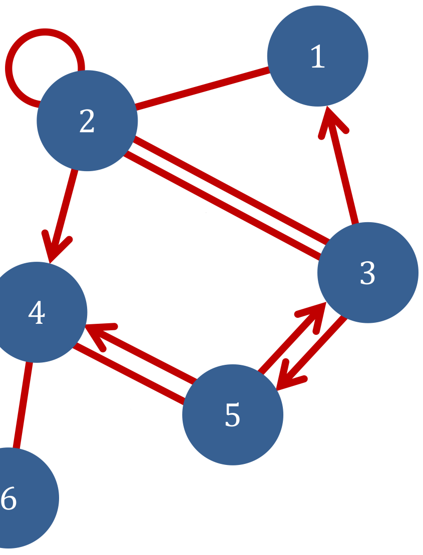


Figure 2.1.: Illustration of a GNN architecture. The blue circles indicate the vertex $\mathbf{v}_i$ while the red connections indicate the edges $\mathbf{e}_{ij}$. Latter ones can be distinguished in directed (arrows), undirected (no arrows), and self-loops, connecting $\mathbf{v}_i$ with itself, as shown in $\mathbf{v}_2$. Each edge and vertex is described as a feature vector. Adapted from Ref. [45].

- Self-loops provide information from a previous state of a vertex to its following state by introducing edges that connect the vertex to itself.

Edge connections are represented by the adjacency matrix $\mathbf{A} \in \mathbb{R}^{|V|\times|V|}$, indicating connections from $\mathbf{v}_i$ (row) to $\mathbf{v}_j$ (column). The example shown in Fig. 2.1 translates to an adjacency matrix of the form

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}, \tag{2.1}$$

where all edges are marked by non-zero elements. In special cases, entries in A_{ij} determine dedicated edge weights, allowing for $A_{ij} \neq 1$. G is called directed when the adjacency is not symmetric $A_{ij} = A_{ji}$. Further, G is considered to be fully connected when $A_{ij} = A_{ji}$ for $i \neq j$. The representation of A_{ij} does not distinguish between an undirected edge and two directed edges. This is because, in practice, undirected edges are usually constructed by implementing two direct edges with opposite directions and combining their features.

A vertex update for $\mathbf{v}_i$, which is explained in the following section, is based on all connected objects. Therefore, A_{ij} depicts the neighborhood $\mathcal{N}(i)$ of $\mathbf{v}_i$, defined as the set of all $\mathbf{v}_j$ with an undirected or directed edge connection to $\mathbf{v}_i$.

2.2. Update functions

The output of a GNN is calculated based on its edge and vertex features, these features must be updated through a series of transformations. This is done by iteratively updating each vertex and edge feature based on the connected edges or vertices. During this update transformation, the information flow depends on $\mathbf{A}$ and the number of update steps. These update steps are elaborated on in the following paragraphs. For this, it is assumed that all $\mathbf{v}_i$ and $\mathbf{e}_{ij}$ are provided as input parameters for $\Omega_{\hat{y}}(V, E)$.

To update $\mathbf{v}_i$, all $\mathbf{e}_{ij}$ pointing towards $\mathbf{v}_i$ are accumulated and combined. This is achieved using the i^{th} row in $\mathbf{A}$ to identify connected edges to combine their information using an aggregation function $\mathcal{F}_{\text{Agg}}$. Common choices are maximum, weighted mean, or mean operations. Using the mean operation results in

$$\mathcal{F}_{\text{Agg}}(\mathbf{v}_i) = \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \mathbf{e}_{ij}, \tag{2.2}$$

for vertex $\mathbf{v}_i$ and $|\mathcal{N}(i)|$ determining the number of neighboured vertices. In the next step, $\mathcal{F}_{\text{Agg}}(\mathbf{v}_i)$ is used to update $\mathbf{v}_i$ by applying an update function ϕ

$$\mathbf{v}'_i = \phi\left(\mathbf{v}_i, \mathcal{F}_{\text{Agg}}(\mathbf{v}_i)\right). \tag{2.3}$$

The function ϕ can, for example, be implemented by an MLP transforming the concatenated vectors of $\mathbf{v}_i$ and $\mathcal{F}_{\text{Agg}}(\mathbf{v}_i)$. In addition, ϕ is not required to be dependent on $\mathbf{v}_i$. After updating $\mathbf{v}_i$, the edges $\mathbf{e}_{ij}$ are updated in a similar way by an independent update function ψ

$$\mathbf{e}'_{ij} = \psi\left(\mathbf{v}_i, \mathbf{v}_j, \mathbf{e}_{ij}\right). \tag{2.4}$$

Where ψ does not necessarily have to depend on $\mathbf{e}_{ij}$.

A fundamental advantage of GNNs over other NN architectures is the local conception of ϕ and ψ relying only on directly connected edges and vertices, respectively. This concept implies that the update procedure is independent of the overall structure, allowing arbitrary complexity of G in terms of connectivity, and it also introduces invariance for the input order of $\mathbf{v}_i$. Analogue to the number of neurons in layer n_l in Eq. (1.10), the output dimension of ψ_l and ϕ_l can differ from their input dimension, facilitating changes in the feature dimensions after each update step.

Since many GNN applications rely dominantly on vertex rather than edge properties, edge features are not always necessarily required, and the edges can solely be used to determine $\mathcal{N}(i)$. The vertex updates can then be made by directly aggregating the vertex features of the neighborhood.

After updating $\mathbf{v}_i$ and $\mathbf{e}_{ij}$ for a fixed number of iterations, a final classification or regression prediction can be conducted based on G. This can be made at the vertex, edge, or graph level:

- A vertex-level prediction can be interpreted as a task of classifying one or more jets in an pp event.
- An edge classification can be interpreted as a task to predict relative information between two objects.
- A graph-level prediction requires a pooling operation on V or E to obtain global information and to conduct a final prediction. This can be seen as an event classification using features from reconstructed objects.

This can be achieved through a mean operation on all vertex features to obtain a final graph-level prediction using an MLP

$$\Omega_{\text{GNN}} = \Omega^{G}_{\text{MLP}} \left(\frac{1}{|V|} \sum_{\mathbf{v}_i \in V} \mathbf{v}_i \right). \tag{2.5}$$

The general GNN properties and the flexibility of information processing between vertices and edges discussed in this section give an impression of how flexible GNNs are and how versatile their applications can be. A full example of such an implementation using a graph-level prediction is demonstrated in Part III.

3. Normalizing flows

While the previous chapter focused on models for classification and regression tasks, such as MLPs and GNNs, this chapter turns to generative models, in particular NFs [46], that are capable of modeling complex, continuous probability distributions p_Y. NFs are utilized to map a known latent space Z to complex target space Y

$$\Omega_{\mathrm{NF}}^{-1} : Z \mapsto Y \quad \text{with} \quad Y, Z \subseteq \mathbb{R}^n. \tag{3.1}$$

In contrast to other generative models, NFs are required to be bijective

$$\Omega_{\mathrm{NF}}^{-1}(\Omega_{\mathrm{NF}}(\mathbf{z})) = \mathbf{z}, \quad \forall \mathbf{z} \in Z. \tag{3.2}$$

If $\mathbf{z}$ follows a known distribution, NFs can be used for efficient sampling from Y

$$\Omega_{\mathrm{NF}}^{-1}(z) \sim Y, \tag{3.3}$$

when the distribution of Y is complex or analytically not known. Due to this bijective property, NFs are also referred to as invertible neural networks (INNs). NFs are compositions of multiple bijective flow functions $\mathcal{T}$, similar to layers $l = \{1, 2, ..., L\}$ of an MLP:

$$\Omega_{\mathrm{NF}} = \mathcal{T}^{(L)} \circ \mathcal{T}_{L-1} \circ \cdots \circ \mathcal{T}_0 \quad \text{and} \quad \Omega_{\mathrm{NF}}^{-1} = \mathcal{T}_0^{-1} \circ \ldots \mathcal{T}_{L-1}^{-1} \circ \mathcal{T}_L^{-1}, \tag{3.4}$$

where more flows $\mathcal{T}$ increase the expressiveness of Ω_{NF}. To apply the sampling procedure in Eq. (3.2), Ω_{NF} is trained to map the unknown distribution of Y to the latent distribution of Z.

There are several architectures for $\mathcal{T}$ following different paradigms. The most important flow architectures are summarized and described in Ref. [47]. In this chapter, three prominent flow architectures are explained:

- the Real-value Non-Volume Preserving (RNVP) flow [48] in Section 3.3.1
- the masked autoregressive flow (MAF) [49] in Section 3.3.2
- and the cubic spline flow (CSF) [50] in Section 3.3.3.

The analyses performed in Ref. [51] deal with the architectures discussed in this chapter. Since these analyses were designed to pinpoint a suitable candidate for the study conducted in Part IV, this chapter generally relies on Ref. [51].

3.1. Change of variables

In probability theory, problems may be facilitated by transforming a complex PDF p_Y to a simpler one p_Z. Such a transformation can be expressed by a flow $\mathcal{T}$ or a combination of flows Ω_{NF}.

To obtain the probability of p_Y in a phase space A, it has to be integrated over A

$$P\left(\mathrm{y} \in A\right) = \int_A p_Y(\mathrm{y})\mathrm{dy}. \tag{3.5}$$

When transforming Y to Z using the transformation Ω_{NF}, the integral reads

$$\int_A p_Y(\mathrm{y})\mathrm{dy} = \int_{\mathcal{T}(A)} p_Z(\mathrm{z})\mathrm{dz} \tag{3.6}$$

$$= \int_A p_Z(\mathcal{T}(\mathrm{y})) \cdot |\mathrm{det} J_{\mathcal{T}}(\mathrm{y})|\mathrm{dy} \tag{3.7}$$

with the Jacobian matrix

$$J_{\mathcal{T}}(\mathrm{y}) = \frac{\partial \mathcal{T}}{\partial \mathrm{y}} = \begin{pmatrix} \frac{\partial \mathcal{T}_1}{\partial y_1} & \cdots & \frac{\partial \mathcal{T}_1}{\partial y_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial \mathcal{T}_n}{\partial y_1} & \cdots & \frac{\partial \mathcal{T}_n}{\partial y_n} \end{pmatrix}. \tag{3.8}$$

$J_{\mathcal{T}}(\mathrm{y})$ occurs in the functional determinant $|\mathrm{det} J_{\mathcal{T}}(\mathrm{y})|$, which operates as a scaling factor to guarantee volume preservation or probability conservation for the transformation $\mathcal{T}$. With this assumption, the transformation $\mathcal{T}$ can be defined to map Y to Z. Using the change of variable formula in Eq. (3.7), the density of y can be inferred. This is possible, even if p_Y is analytically not available.

Applying Eq. (3.4) leads to the variable transformation for Ω_{NF} consisting of L flows $\mathcal{T}$

$$p_Y(y) = p_Z(y) \prod_{l=1}^{L} \left|\mathrm{det}\, J_{\mathcal{T}^{(l)}}\left(\mathcal{T}^{(l-1)}\right)\right|, \tag{3.9}$$

where $\mathcal{T}^{(l-1)}$ indicates the output of the previous flow layer. $J_{\mathcal{T}}$ should have a form to keep the costs of computing its determinant low. Therefore, $\mathcal{T}$ is constructed in a way that $\mathbf{J}_{\mathcal{T}}(\mathrm{y})$ is of triangular or diagonal form, reducing the need to compute all elements in $\mathbf{J}_{\mathcal{T}}(\mathrm{y})$.

3.2. Kullback-Leibler divergence

The Kullback-Leibler (KL) divergence [52] is a quantity to measure the similarity between a target distribution $p_Y(\mathrm{y})$ and a distribution of a model output $\hat{p}_Y(\mathrm{y}, \boldsymbol{\theta})$. This can be utilized as a loss function L for the training of Ω_{NF}.

To do so, $p_Y(\mathbf{y})$ and $\hat{p}_Y(\mathbf{y},\boldsymbol{\theta})$ are considered as likelihoods and $\log\lambda_{\mathrm{LR}}$ quantifies their similarity

$$\log\lambda_{\mathrm{LR}} = \log\left(\frac{p_Y(\mathbf{y})}{\hat{p}_Y(\mathbf{y},\boldsymbol{\theta})}\right) \tag{3.10}$$

$$\mathbb{E}[\log\lambda_{\mathrm{LR}}] = \int \mathrm{d}\mathbf{y}\, p_Y(\mathbf{y}) \cdot \log\left(\frac{p_Y(\mathbf{y})}{\hat{p}_Y(\mathbf{y},\boldsymbol{\theta})}\right), \tag{3.11}$$

where $\mathbb{E}[\log\lambda_{\mathrm{LR}}]$ corresponds the expectation value of $\log\lambda_{\mathrm{LR}}$ for a continuous variable $\mathbf{y}$. This refers directly to the KL divergence

$$\mathbb{E}[\log\lambda_{\mathrm{LR}}] = \mathrm{KL}[p_Y(\mathbf{y}) \parallel \hat{p}_Y(\mathbf{y},\boldsymbol{\theta})]. \tag{3.12}$$

Based on Eq. (3.11), the MLE principle is applied to find $\hat{\boldsymbol{\theta}}$. According to the definition of λ_{LR} in Eq. (1.5), this is achieved by minimizing Eq. (3.12). When optimizing $\boldsymbol{\theta}$, Eq. (3.12) can be simplified

$$\mathrm{KL}[p_Y(\mathbf{y}) \parallel \hat{p}_Y(\mathbf{y},\boldsymbol{\theta})] = \int \underbrace{\mathrm{d}\mathbf{y}\, p_Y(\mathbf{y}) \cdot \log(p_Y(\mathbf{y}))}_{\text{const.}} - \underbrace{\int \mathrm{d}\mathbf{y}\, p_Y(\mathbf{y}) \cdot \log(\hat{p}_Y(\mathbf{y},\boldsymbol{\theta}))}_{=\mathbb{E}_{p_Y}[\log(\hat{p}_Y(\mathbf{y},\boldsymbol{\theta}))]} \tag{3.13}$$

$$\propto -\mathbb{E}_{p_Y}\left[\log(\hat{p}_Y(\mathbf{y},\boldsymbol{\theta}))\right], \tag{3.14}$$

where the first term in Eq. (3.13) is independent of $\boldsymbol{\theta}$ and therefore irrelevant for the minimization. Accordingly, the minimization is reduced to Eq. (3.14).

The estimation $\hat{p}_Y(\mathbf{y},\boldsymbol{\theta})$ still follows an intractable PDF, while obeying Eq. (3.9), Ω_{NF} can be used for transforming p_Y to p_Z. For a finite dataset X of size N, the expectation value is approximated by the sample mean:

$$-\mathbb{E}_{p_Y}\left[\log(\hat{p}_Y(\mathbf{y},\boldsymbol{\theta}))\right] = -\frac{1}{N}\sum_i^N \log(\hat{p}_Y(\mathbf{y},\boldsymbol{\theta})) \tag{3.15}$$

$$= -\frac{1}{N}\sum_i^N \left(\underbrace{\log(p_Z(\Omega_{\mathrm{NF}}(\mathbf{y}_i,\boldsymbol{\theta})))}_{\propto \frac{-\Omega_{\mathrm{NF}}(\mathbf{y}_i,\boldsymbol{\theta})^2}{2} \text{ for } p_Z \sim \mathcal{N}} + \sum_l^L \log\left|\det J_{\mathcal{T}_i^{(l)}}\left(\mathcal{T}_i^{(l-1)}\right)\right|\right). \tag{3.16}$$

Utilizing the logarithm and choosing a Normal distribution $\mathcal{N}$ as latent space simplifies the training objective further. Consequently, Eq. (3.16) can be minimized by adjusting $\boldsymbol{\theta}$ in Ω_{NF}, such that Ω_{NF} maps the base distribution p_Z while minimizing the sum of the logarithmic determinants of the Jacobian components of each transformation.

3.3. Flow transformations

A flow $\mathcal{T}$ is the fundamental building block of NFs. Typically, multiple successive flows $\mathcal{T}^{(l)}$ are combined to construct a complete NF model Ω_{NF}. Various architectures for flows, such as planar or linear flows [53], use simple, bijective transformations. While each is considered computationally efficient, these come short in terms of expressiveness for approximating complex functions p_Y. Therefore, more advanced architectures are considered for this work. Coupling flows [54, 48, 55] rely on a splitting of the feature set. Bijectivity is introduced by applying coupling functions on the split set. Autoregressive flows [56, 57, 49] introduce a per-feature coupling in a recursive way to improve expressiveness. Spline flows [58, 59, 50] are the third class of flows under discussion, which models $\mathcal{T}$ by piecewise-defined spline functions.

The following section gives a detailed explanation of the RNVP, MAF, and CSF. In each section, the architectures and approaches for efficiently calculating the determinant are discussed.

3.3.1. Real-valued Non-volume Preserving flow

Coupling flows rely on splitting the inputs into two parts $\mathbf{y} \rightarrow (\mathbf{y}_1, \mathbf{y}_2)^{\mathrm{T}}$ with

$$\mathbf{y} \in \mathbb{R}^n \quad \text{and} \quad \mathbf{y}_1 \in \mathbb{R}^{n_1}, \mathbf{y}_2 \in \mathbb{R}^{n_2} \quad \text{with} \quad n = n_1 + n_2. \tag{3.17}$$

Ω_{NF} uses an invertable coupling function $\mathbf{c}(\mathbf{y}_1; \mathbf{y}_2, \boldsymbol{\theta}) = \mathbf{z}_2$ to transform one half of $\mathbf{y}$ while leaving the other one unchanged. This coupling function uses $\mathbf{y}_1$ to conduct simple operations on $\mathbf{y}_2$. All inputs are transformed by applying the same mechanism on the other half $\mathbf{c}(\mathbf{y}_2; \mathbf{y}_1, \boldsymbol{\theta}) = \mathbf{z}_2$. The RNVP utilizes an affine coupling function, reading

$$\mathcal{T}_{\mathrm{RNVP}} : \begin{pmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{pmatrix} \mapsto \begin{pmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{y}_1 \\ \underbrace{\mathbf{s}(\mathbf{y}_1, \boldsymbol{\theta}) \odot \mathbf{y}_2 + \mathbf{t}(\mathbf{y}_1, \boldsymbol{\theta})}_{\mathbf{c}(\mathbf{y}_2; \mathbf{y}_1, \boldsymbol{\theta})} \end{pmatrix}, \tag{3.18}$$

where $\odot$ indicates element-wise multiplication. The functions $\mathbf{s}$ and $\mathbf{t}$ are used as scale and shift operations for $\mathbf{y}_2$ and are obtained from an

$$\mathrm{NN}(\mathbf{y}_1, \boldsymbol{\theta}) : \mathbf{y}_1 \mapsto (\mathbf{s}, \mathbf{t})^{\mathrm{T}}, \tag{3.19}$$

e.g. an MLP. Since $\mathbf{c}(\mathbf{y}_2; \mathbf{y}_1, \boldsymbol{\theta})$ is easy to invert, $\mathcal{T}_{\mathrm{RNVP}}$ is also invertible. The core effect of using the coupling mechanism is to avoid inversion of the NN for $\mathcal{T}^{-1}_{\mathrm{RNVP}}$. This central feature is illustrated in Fig. 3.1 where the affine coupling function for $\mathcal{T}_{\mathrm{RNVP}}$ is illustrated on the left and the inverse case is shown on the right. A transformation of $\mathbf{y}_1$ is done by switching $\mathbf{y}_1$ and $\mathbf{y}_2$ and applying a second instance of $\mathcal{T}_{\mathrm{RNVP}}$.

Following Eq. (3.8), the Jacobian can be determined for Eq. (3.18) as

$$J_{\mathcal{T}_{\mathrm{RNVP}}}(\mathbf{y}) = \begin{pmatrix} 1 & 0 \\ \frac{\mathrm{d}\mathbf{z}_2}{\mathrm{d}\mathbf{y}_1} & \mathbf{s}(\mathbf{y}_1, \boldsymbol{\theta}) \end{pmatrix}, \tag{3.20}$$

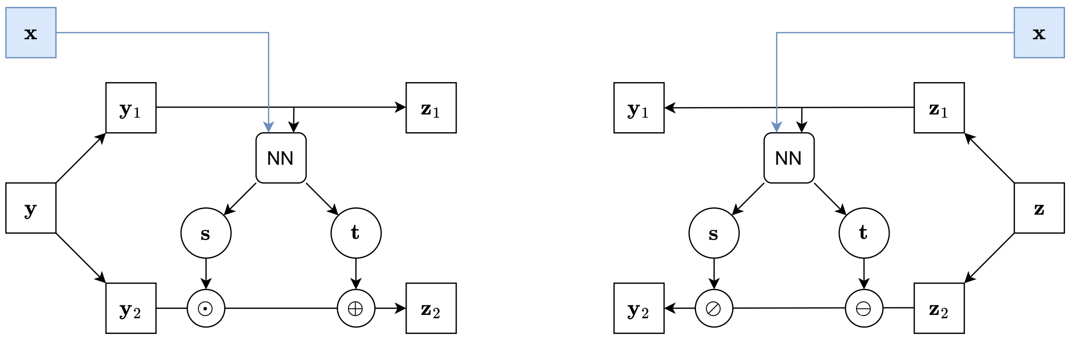


Figure 3.1.: Illustration of the affine coupling transformation used by the RNVP flow. An NN is used for scaling and shifting $\mathbf{y}_2$ (left). For the inverse operation (right), the NN itself does not need to be inverted. All operations are done element-wise, indicated by $\odot$, $\oplus$, $\oslash$, and $\ominus$. The blue marked input vector $\mathbf{x}$ indicates an optional input, evaluating $\mathcal{T}_{\text{RNVP}}$ conditionally. Adapted from Ref. [51].

which is a triangular matrix. The costly-to-compute off-diagonal elements cancel out when calculating the determinant, simplifying Eq. (3.20) to

$$\det J_{\mathcal{T}_i^{(l)}}\left(\mathcal{T}_i^{(l-1)}\right) = \prod_i^{n_2} \mathbf{s}(\mathbf{y}_1, \boldsymbol{\theta})_i. \tag{3.21}$$

3.3.2. Masked autoregressive flows

Instead of splitting the input, the maf [49] operates on each element of $\mathbf{y}$. Analogous to the coupling functions, an affine transformation of $\mathbf{y}_i$ is pursued by calculating $s_i(\mathbf{y}_{j<i}, \boldsymbol{\theta})$ and $t_i(\mathbf{y}_{j<i}, \boldsymbol{\theta})$ based on all $\mathbf{y}_i$ with $j < i$. For the MAF, the mapping $\mathcal{T}_{\text{MAF}} : \mathbf{y} \mapsto \mathbf{z}$ reads:

$$\mathcal{T}_{\text{MAF}} : \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{pmatrix} \mapsto \begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_n \end{pmatrix} = \begin{pmatrix} s_1(\theta) \cdot y_1 + t_1(\theta) \\ s_2(y_1, \theta) \cdot y_2 + t_2(y_1, \theta) \\ s_3(y_1, y_2, \theta) \cdot y_3 + t_3(y_1, y_2, \theta) \\ \vdots \\ s_n(y_1, y_2, \dots, y_{j-1}, \theta) \cdot y_n + t_n(y_1, y_2, \dots, y_{j-1}, \theta) \end{pmatrix}. \tag{3.22}$$

Computing $\mathbf{z}_i$ autoregressively based on $\mathbf{y}_{j<i}$ leads to a triangular form of $J_{\mathcal{T}_{\text{MAF}}}(\mathbf{y})$. Therefore, the lower triangle is again irrelevant when computing the determinant, only the trace elements contribute to the Jacobian

$$\det J_{\mathcal{T}_{\text{MAF}}}(\mathbf{y}) = \prod_i^n s_i(\mathbf{y}_{j<i}, \boldsymbol{\theta}), \quad \text{for} \quad i < j. \tag{3.23}$$

Since the affine couplings in Eq. (3.22) fulfill the same purpose as in Eq. (3.18), an inverse of element y_i of $\mathcal{T}_{\text{MAF}}$ is given as

$$y_i = \frac{z_i - t_i(\mathbf{y}_{j<i}, \boldsymbol{\theta})}{s_i(\mathbf{y}_{j<i}, \boldsymbol{\theta})}. \tag{3.24}$$

In contrast to Section 3.3.2, which is computed in one vectorized step, y_i in Eq. (3.24) relies on the results for $\mathbf{y}_{j<i}$, requiring an iterative computation. Computing s_i and t_i using a dedicated NN for each dimension i scales like $O^{(n)}$, making it more costly for applications in higher dimensions.

The Masked Autoencoder for Distribution Estimation (MADE) [60] resolves this issue by utilizing a single MLP to compute all $s_i(\mathbf{y}_{j<i}, \boldsymbol{\theta})$ and $t_i(\mathbf{y}_{j<i}, \boldsymbol{\theta})$. The dependency problem is solved by introducing masking layers between the hidden layers of the MLP. Following the algorithm described in Ref. [60], this masking deactivates neurons systematically to ensure s_i and t_i are computed exclusively on $\mathbf{y}_{j<i}$. This approach has similarities to the dropout mechanism explained in Section 1.7 and can be viewed as an example for its interpretation as an ensemble training.

Compared to the RNVP, which applies an affine transformation to one half of the feature set $\mathbf{y}$, the MAF performs such transformations sequentially for each individual feature. As a consequence, the MAF is considered particularly effective in modeling complex distributions Y.

3.3.3. Cubic spline flow

Cubic spline flows (CSFs) [50] follow the coupling scheme of two feature sets $\mathbf{y} \longrightarrow (\mathbf{y}_1, \mathbf{y}_2)^{\mathrm{T}}$ following Eq. (3.17). But instead of applying an affine transformation for the coupling step $\mathbf{c}(\mathbf{y}_1; \mathbf{y}_2, \boldsymbol{\theta}) = \mathbf{z}_2$, CSFs model $\mathbf{c}(\mathbf{y}_1; \mathbf{y}_2, \boldsymbol{\theta})$ by third-order polynomials in three bins of $\mathbf{y}_2$.

These are exemplary illustrated in Fig. 3.2 as red, green, and orange lines that are constrained by four knots K (blue). While the first knot is a fixed boundary condition $(k_y^{(j,0)}, k_z^{(j,0)}) = (-12, -12)$, the coordinates $(k_y^{(j,\alpha_i)}, k_z^{(j,\alpha_i)})$ with $\alpha_i \in \{1, 2, 3\}$ of the remaining knots are learnable parameters of polynom α_i. After determining the coordinates for all knots, the interval of the spline gets rescaled to $[-12, 12]$ to maintain symmetry. The choice of this interval follows Ref. [51] and is a hyperparameter of the model. This implies that the last knot is always rescaled to $(k_y^{(j,3)}, k_z^{(j,3)}) = (12, 12)$. Outside of the polynomial range, an identity function is applied. The coupling function is written as

$$\mathbf{c}(\mathbf{y}_1; \mathbf{y}_2, \boldsymbol{\theta}) = \begin{cases} \mathbf{a}^{(1)}\xi^3 + \mathbf{b}^{(1)}\xi^2 + \mathbf{c}^{(1)}\xi^1 + \mathbf{d}^{(1)}, & \mathbf{k}_y^{(0)} \leq \mathbf{y}_2 < \mathbf{k}_y^{(1)} \\ \mathbf{a}^{(2)}\xi^3 + \mathbf{b}^{(2)}\xi^2 + \mathbf{c}^{(2)}\xi^1 + \mathbf{d}^{(2)}, & \mathbf{k}_y^{(1)} \leq \mathbf{y}_2 < \mathbf{k}_y^{(2)} \\ \mathbf{a}^{(3)}\xi^3 + \mathbf{b}^{(3)}\xi^2 + \mathbf{c}^{(3)}\xi^1 + \mathbf{d}^{(3)}, & \mathbf{k}_y^{(2)} \leq \mathbf{y}_2 < \mathbf{k}_y^{(3)} \end{cases} . \tag{3.25}$$

The bin boundaries $\mathbf{k}_y^{(\alpha_i)}$ decide which polynom α_i is applied for the mapping, while $\xi = \mathbf{y}_2 - \mathbf{k}_y^{(\alpha_i)}$ describes the relative position of $\mathbf{y}_2$ in the corresponding bin. In order to determine the coefficients $\mathbf{a}^{(\alpha_i)}, \mathbf{b}^{(\alpha_i)}, \mathbf{c}^{(\alpha_i)}, \mathbf{d}^{(\alpha_i)} \in \mathbb{R}^{n_2}$ of the polynomial, the position of each knot $(k_y^{(j,\alpha_i)}, k_z^{(j,\alpha_i)})$ is requiered to be known, with the derivatives $\{d_k^{(j)}\}_{k=0}^{K=3}$ to construct a smooth function. Based on these learned parameters, the Steffen method [61] is utilized to conclude on $\mathbf{a}^{(\alpha_i)}$, $\mathbf{b}^{(\alpha_i)}$, $\mathbf{c}^{(\alpha_i)}$, and $\mathbf{d}^{(\alpha_i)}$. Since these coefficients are determined based on an NN output, they are inherently dependent on $\mathbf{y}_1$ and $\boldsymbol{\theta}$. This dependence is

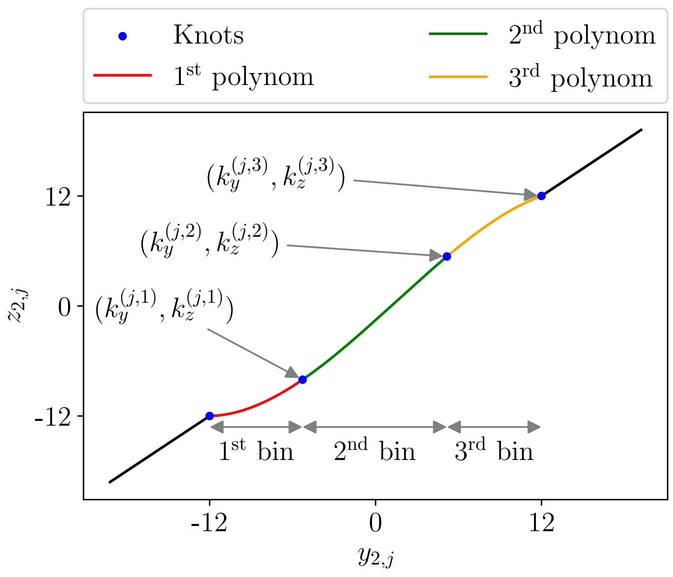


Figure 3.2.: Example for a third order polynomial spline mapping of the j^{th} component of $\mathbf{y}_2$ to the corresponding component in $\mathbf{z}_2$. Four knots K divide the spline into three bins (red, green, orange). While the first knot is fixed, the coordinates $(k_y^{(j,\alpha_i)}, k_z^{(j,\alpha_i)})$ with $\alpha_i \in \{1, 2, 3\}$ of the remaining ones are learnable parameters. Taken from Ref. [51].

omitted in Eq. (3.25) for clarity. The inverse function of Eq. (3.25) corresponds to mirroring the polynomials along unity. An analytical formulation can be found in Ref. [62]. Sharing the same coupling mechanism as the RNVP, the CSF results in a Jacobian matrix with identical structure. Following Eq. (3.20), the trace elements of the Jacobian matrix for the CSF are given by

$$\frac{\mathrm{d}\mathbf{z}_2}{\mathrm{d}\mathbf{y}_2} = 3\mathbf{a}^{(\alpha_i)}\xi^2 + 2\mathbf{b}^{(\alpha_i)}\xi + \mathbf{c}^{(\alpha_i)}. \tag{3.26}$$

The Jacobian determinant can be written as the product of all elements

$$\det J_{\mathcal{T}_{\text{CSF}}}(\mathbf{y}) = \prod_{j}^{n_2} \left(3a_j^{(\alpha_i)}\xi_j^2 + 2b_j^{(\alpha_i)}\xi_j + c_j^{(\alpha_i)}\right). \tag{3.27}$$

To transform both $\mathbf{y}_1$ and $\mathbf{y}_2$, permutation layers shuffling the feature dimension have to be implemented.

In conclusion, the CSF uses the coupling mechanism with cubic polynomials to model $\mathcal{T}_{\text{CSF}} : Z \mapsto Y$. Since the construction of these presents a more expressive coupling function than the affine one in Eq. (3.18), the CSF can be expected to be slightly better in performance than the RNVP.

3.3.4. Conditional normalising flows

The transforms for different flows are mappings $\Omega_{\text{NF}}(\mathbf{z}) : \mathbf{z} \mapsto \mathbf{y}$. In many practical applications, the target distribution is not fixed but depends on additional conditioning information. One important application in HEP is the unfolding of a distribution with respect to a specific phase space configuration $\mathbf{x}$. This conditional input is $\mathbf{x}$ and is interpreted as a parameterization of $\Omega_{\text{NF}}(\mathbf{z}, \mathbf{x}) : \mathbf{z} \mapsto \mathbf{y}$. For the $\mathcal{T}_{\text{RNVP}}$, this is illustrated in the left section of Fig. 3.1 for the NN, where $\mathbf{x}$ is concatenated with $\mathbf{y}$ and passed to the NN. Since the additional input x does not change the underlying concept of the inversion for coupling flows, the same procedure holds for $\mathcal{T}_{\text{RNVP}}^{-1}$, as shown in the right section of Fig. 3.1.

Since the CSF uses an NN to determine $(k_y^{(j,\alpha_i)}, k_z^{(j,\alpha_i)})$ and $\{d_k^{(j)}\}_{k=0}^{K=3}$, as well as the MAF for t_i and s_i, the same extension to a conditional model can be made by adding x as input to the NN in these cases. This integration makes NFs well-suited tools for addressing applications requiring context-dependent sampling or density estimation.

3.4. Transformation comparison

For the comparison of $\mathcal{T}_{\text{RNVP}}$, $\mathcal{T}_{\text{MAF}}$, and $\mathcal{T}_{\text{CSF}}$, several studies have been pursued in Ref. [51]. The selection of a suitable candidate for the studies in Part IV is based on the evaluation time and the approximation capability. For the latter, the Wasserstein distance [63] has been employed as a metric to quantify residual differences between the generated $\hat{\mathbf{y}}$ and target distributions $\mathbf{y}$.

The tests have been conducted for datasets with dimensions $d \in \{2, 4, 15, 25\}$. Here, the two-dimensional dataset consisted of a half-moon-shaped sample, while in the four-dimensional case, a Gaussian distribution and a linear function were added. For the extension to $d = 4$, a three-dimensional shape of the letter "S" and a dimension representing two interlocking circles were selected. Additionally, the Gaussian dimension was replaced by a correlated seven-dimensional Gaussian distribution for $d = 15$ and by a correlated 17-dimensional Gaussian distribution for the dataset with $d = 25$.

In Fig. 3.3, the different flow models are compared in terms of their approximation capabilities. For low-dimensional cases, especially for $d = 2$, all transformations exhibit comparable performance. However, as the dimensionality increases, the results show that $\mathcal{T}_{\text{MAF}}$ demonstrates more approximation ability compared to $\mathcal{T}_{\text{CSF}}$ and $\mathcal{T}_{\text{RNVP}}$, by achieving the highest $\hat{W}_1$ for d=15 and d=25. Compared to the $\mathcal{T}_{\text{RNVP}}$, the $\mathcal{T}_{\text{CSF}}$ performs slightly better in higher dimensions. Nevertheless, since the analysis in Part IV is conducted in two dimensions, these results show that all three flow models can be considered suitable candidates regarding their capability to model complex two-dimensional structures.

Figure 3.4 presents a timing comparison for the evaluation of the three transformations. The left plot shows the evaluation direction used during inference, from the latent space

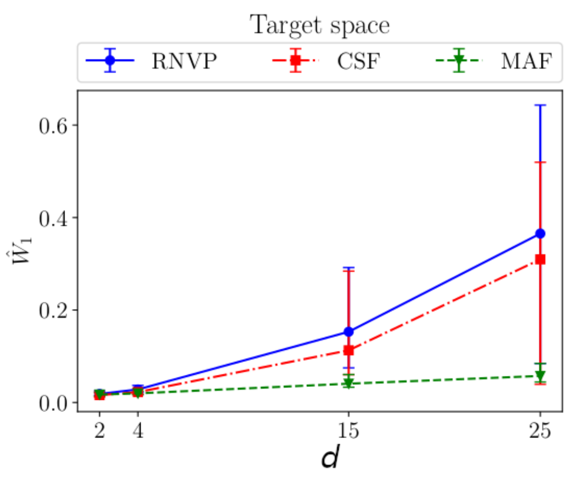


Figure 3.3.: Wasserstein distance $\hat{W}_1$ for the $\mathcal{T}_{\text{RNVP}}$, $\mathcal{T}_{\text{MAF}}$, and $\mathcal{T}_{\text{CSF}}$ for different dimensions. Each result is given as the median value and the corresponding 95 % confidence interval (CI). Adapted from Ref. [51].

Z to the target space Y. Across all dimensions, the same ranking is observed: $\mathcal{T}_{\text{RNVP}}$ is the fastest among the three models, followed by $\mathcal{T}_{\text{CSF}}$, and $\mathcal{T}_{\text{MAF}}$. In the opposite direction, from the target space Y to the latent space Z, which corresponds to the training direction of the NFs, $\mathcal{T}_{\text{RNVP}}$ performs slightly better than $\mathcal{T}_{\text{MAF}}$ for the two-dimensional case but slightly worse in higher dimensions. The significant timing differences observed for $\mathcal{T}_{\text{MAF}}$ arise from the fact that its inverse function, shown in Eq. (3.24), must be computed in an iterative way. Among the three models, $\mathcal{T}_{\text{CSF}}$ exhibits the slowest overall performance. Additionally, all models show better performance when evaluated in the training direction compared to the inference direction.

In conclusion, the performance differences between the transformations are marginal in low-dimensional settings. Since the two-dimensional case is of primary interest in the study conducted in Part IV, the $\mathcal{T}_{\text{RNVP}}$ appears as a natural choise due to its advantageous evaluation times in both the inference and training directions.

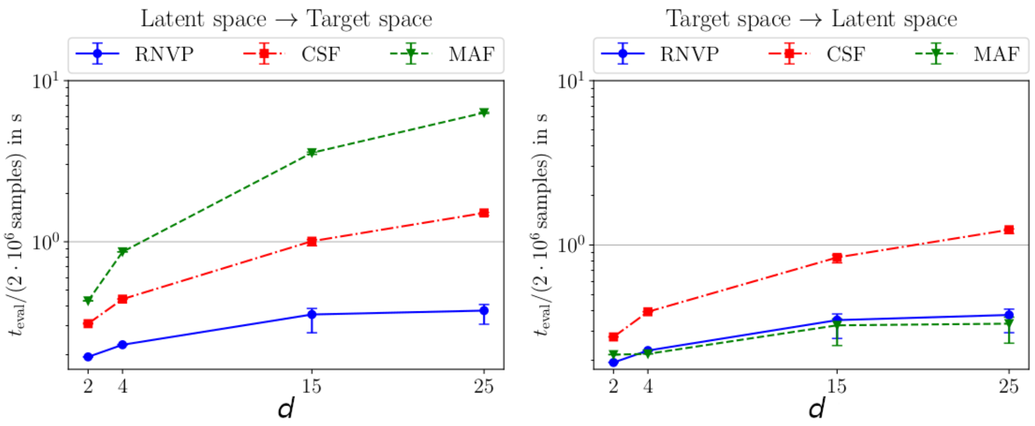


Figure 3.4.: Evaluation test for $\mathcal{T}_{\mathrm{RNVP}}$, $\mathcal{T}_{\mathrm{MAF}}$, and $\mathcal{T}_{\mathrm{CSF}}$. The left plot depicts the evaluation direction from the latent space Z to the target space Y, and the right plot shows the direction from the target space Y to the latent space Z. Adapted from Ref. [51].

Part II.

High-Energy Physics as a Frontier for Machine Learning

4. The Standard Model

The so-called SM is one of the most robust and tested models in physics. It was developed in the second half of the twentieth century and describes properties and interactions between the known fundamental particles. These can be split into two classes: Fermions, which are fundamental building blocks of matter, and bosons, which are exchanged between fermions and convey one of the three forces described by the SM.

Fermions are particles with spin $\frac{1}{2}$ and can further be divided into three generations of quarks and leptons. Up-type quarks, namely up, charm, and top, are of positive electrical charge, while down-type quarks, down, strange, and bottom, convey a negative electrical charge. Electrons, muons, and taus convey negative electrical charge, while their neutrino partners govern no charge at all: together, they form the leptonic sector in the SM. Bosons allow for interactions between fermions. The electromagnetic force couples to all electrically charged particles with an infinite interaction range and is propagated by photons γ. Weak interactions are mediated by $\mathrm{W}^{\pm}$ and Z bosons, which carry weak isospin on very short ranges ($\approx 10^{-18}$). This force allows for the β-decay of nuclei and is the only one to access the neutrino sector. Lastly, the strong force acts on a nuclear scale, binding quarks to stable particles (hadrons). Its mediators are gluons g, which govern color charge $C \in \{\text{red}, \text{green}, \text{blue}\}$ leading to color refinement. An overview of the fundamental force is provided in Table 4.1. An overview of properties for the explained leptons, quarks, and bosons is provided in Fig. 4.1.

The SM is a quantum field theory (QFT) that combines quantum mechanics and special relativity to describe particle interactions. In QFT, particles are not described as classical points but as fields existing at every point in time and space.

To learn about the properties of these fields, one has to define the action S

$$S = \int \mathscr{L} \mathrm{d}^4 x, \tag{4.1}$$

where the Lagrangian density $\mathscr{L}$ encodes the dynamics of a physical system and is integrated over three spatial and one time dimensions. By applying the Hamilton principle

Force	Mediator	Range	Charge	Coupling Strength
Electromagnetic	Photon (γ)	Infinite	Electric charge	$\alpha \approx \frac{1}{137}$
Weak	$\mathrm{W}^{\pm}, Z$	$\sim 10^{-18}$ m	Weak isospin	$\alpha_\mathrm{W} \approx 10^{-6}$
Strong	g	10^{-15} m	Color C	$\alpha_s \approx 1$

Table 4.1.: Overview of the fundamental forces in the Standard Model.

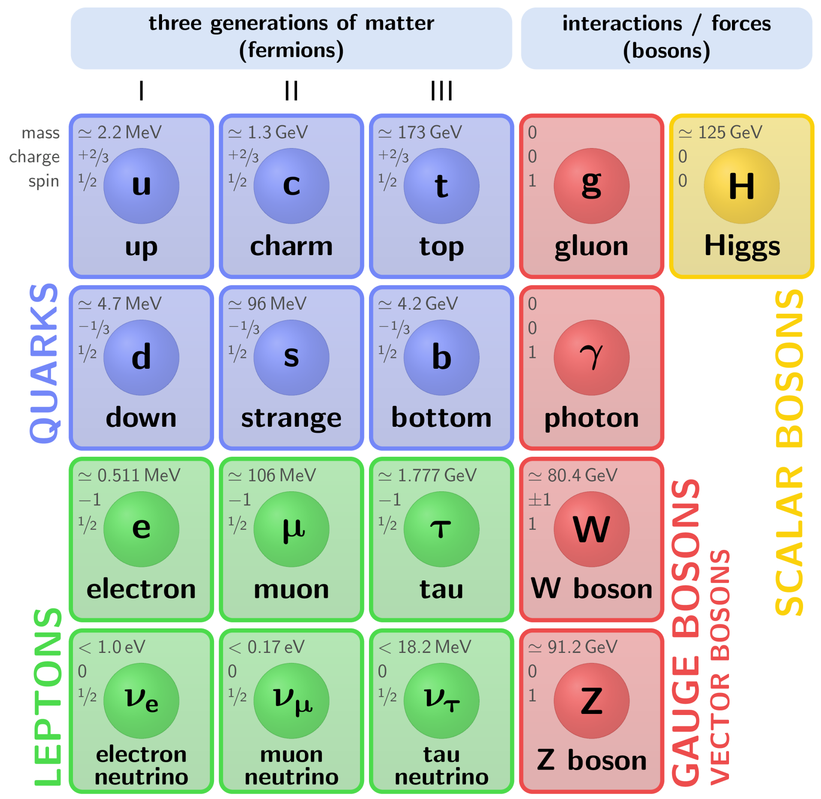


Figure 4.1.: Illustration of the Standard Model of particle physics. Fermions are spin $\frac{1}{2}$ particles that build matter. Bosons have spin 1 and are force mediators, allowing particles to interact. Taken from Ref. [67].

[64], i.e., by minimizing S, the equation of motion can be derived between two points of the physical system. To express $\mathscr{L}_{\text{SM}}$, it must fulfill certain symmetries to stay invariant under local transformations of gauge fields [65]. These are described by the Lie group

$$\text{SU}(3)_C \times \text{SU}(2)_L \times \text{U}(1)_Y, \tag{4.2}$$

where $\text{SU}(3)_C$ describes the strong force and $\text{SU}(2)_L \times \text{U}(1)_Y$ the electroweak force. The indices C (color) and Y (hypercharge) indicate the conserved quantities (charges) of the corresponding forces, which result from the Noether theorem [66]. The $\text{SU}(2)_L$ acts only on particles with left-handed chirality (L).

4.1. Quantum electrodynamics

Quantum electrodynamics (QED) [68, 69, 70] describes the electromagnetic interaction between electrically charged fermions, described by the field ψ, and photons γ. The scalar field ψ has to stay invariant under local U(1) gauge transformations. Such a transformation reads as

$$\psi(x) \longrightarrow \psi'(x) = e^{iq\alpha^a(x)}\psi(x), \tag{4.3}$$

with the q indicating the charge of $\psi(x)$ and $\alpha(x)$ as a scalar function of space-time coordinates. But when applying Eq. (4.3) to the Dirac equation of a free spin-$\frac{1}{2}$ particle

$$(i\gamma^\mu \partial_\mu - m)\psi = 0, \tag{4.4}$$

using the gamma matrices γ^μ fulfilling $\gamma^\mu, \gamma^\nu\} = 2\eta^{\mu\nu} I$, invariance does not hold by default. To maintain this symmetry, the covariant derivative D_μ has to be introduced

$$D_\mu = \partial_\mu - iqA_\mu \tag{4.5}$$

where A_μ can be identified as the electromagnetic vector potential. By using the Lagrangian for a massive free spin-$\frac{1}{2}$ field $\mathscr{L}_{\text{fermion}}$ based on Eq. (4.4) and adding it with the Lagrangian for the vector potential $\mathscr{L}_\gamma$, the Lagrangian for the QED can be obtained

$$\mathscr{L}_{\text{QED}} = \underbrace{\bar{\psi}(x)(i\gamma^\mu D_\mu - m)\psi(x)}_{\mathscr{L}_{\text{fermion}}} - \underbrace{\frac{1}{4}F_{\mu\nu}(x)F^{\mu\nu}(x)}_{\mathscr{L}_\gamma} \tag{4.6}$$

$$= \bar{\psi} i\gamma^\mu \partial_\mu \psi - m\bar{\psi}\psi - q\bar{\psi}\gamma^\mu A_\mu \psi - \frac{1}{4}F^{\mu\nu}F_{\mu\nu}. \tag{4.7}$$

The first term of this equation is the kinetic term of a free fermion, the second can be identified as a mass term for the fermion while the third one describes interactions of the fermion field with the electromagnetic vector potential. The last term uses the electromagnetic field tensor $F^{\mu\nu} = \partial^\mu A^\nu - \partial^\nu A^\mu$ to describe the dynamics of a free photon field. This term ensures that photons obey the Maxwell equations [71, 72].

4.2. Quantum chromodynamics

Quantum chromodynamics (QCD) [73, 74, 75] describes the interactions of strong nuclear force between fermions. These fermions are described by spinor triplets ψ and are representations in the special unitarian group $\text{SU}(3)_C$ with three dimensions corresponding to the color charges red, green, and blue.

To maintain underlying physics, the spinor ψ is required to be invariant under local gauge transformations of $\text{SU}(3)_C$. Such a transformation reads as

$$\psi(x) \longrightarrow \psi'(x) = e^{ig_s \frac{\lambda^a}{2} \alpha^a(x)} \psi(x), \tag{4.8}$$

where λ_i can be identified as the Gell-Mann matrices [76], $T^a = \frac{\lambda^a}{2}$ as the group generators T^a with $a = \{1, 2, ...8\}$, g_s as strong coupling constant and $\alpha^a(x)$ as a space-time-dependent gauge parameters of the ath color. However, the way to obtain invariance is analogous to the one for $\mathscr{L}_{\text{QED}}$.

To ensure invariance under Eq. (4.8), the covariant derivative is introduced as

$$D_\mu \psi(x) = \left(\partial_\mu - ig_s \frac{\lambda^a}{2} G^a_\mu(x)\right) \psi(x). \tag{4.9}$$

Hereby, a gauge field for the $\mathrm{SU}(3)_C$ has to be introduced: the gluon field $G^a_\mu(x)$ component-wise corresponding to T^a. The gluon strength tensor is defined as

$$G^a_{\mu\nu}(x) = \partial_\mu G^a_\nu(x) - \partial_\nu G^a_\mu(x) + g_s f^{abc} G^b_\mu(x) G^c_\nu(x), \tag{4.10}$$

with the structure constants f^{abc} of $\mathrm{SU}(3)_C$. The final QCD Lagrangian $\mathscr{L}_{\mathrm{QCD}}$ is obtained by adding the Lagrangian for a massless, free spin-$\frac{1}{2}$ field while using D_μ, together with the Lagrangian for the gluon field

$$\mathscr{L}_{\mathrm{QCD}} = \underbrace{\bar{\psi}(x) i\gamma^\mu D_\mu \psi(x)}_{\mathscr{L}_{\mathrm{quark}}} - \underbrace{\frac{1}{4} G^a_{\mu\nu}(x) G^{a\mu\nu}(x)}_{\mathscr{L}_{\mathrm{gluon}}}. \tag{4.11}$$

The first term, $\mathscr{L}_{\mathrm{quark}}$, describes the properties of the fermion field if the covariant derivative is multiplied out, a kinetic term for a free quark, and a term describing antiquark-gluon-quark interaction. If $\mathscr{L}_{\mathrm{gluon}}$ is expanded, gluon coupling terms with three and four gluons are obtained. These self-interaction terms are in contrast to QED, where the terms in $\mathscr{L}_\gamma$ cancel out, leading to no γ self-interactions. By using the covariant derivative D_μ in Eq. (4.11), the equation stays invariant under $\mathrm{SU}(3)_C$ transformations, given in Eq. (4.8).

Since Eq. (4.11) allows for gluon self-interactions, this leads to asymptotic freedom. Self-interaction causes the strong coupling α_s to decrease at short distances, allowing the quarks to move approximately asymptotically free. This is in contrast to QED, where the coupling constant decreases with more considerable distances. For extensive lengths, α_s increases with the distance, leading to growing forces between two color-charged particles. At some point, creating a new quark-antiquark pair becomes energetically more favorable, leading to the effect that color-charged particles do not occur in isolated states but in bound states, which are called hadrons. This property of QCD is known as confinement [77] and is a key difference from QED.

Color-charged objects produced in high-energy interactions, like during pp collisions at the LHC, may be separated from their initial states. Due to their high energies, they may briefly propagate and interact with other particles in isolation and begin the hadronization process, leading to confinement by creating new quark-antiquark pairs [78]. This process initiates a cascade, continuing until the system becomes color-neutral. At this point, the created objects combine into final-state particles such as neutrons, protons, kaons, and pions, so that in the end, only color-neutral hadrons remain. Hadronization is a crucial process in high-energy physics, as it allows for the reconstruction of the initial-state particles and provides insight into the dynamics of strong interactions.

4.3. Higgs mechanism

So far, gauge symmetries have not allowed for massive bosons. While this also holds for the electroweak sector in the SM, it was experimentally proven that the $\mathrm{W}^\pm$ and Z

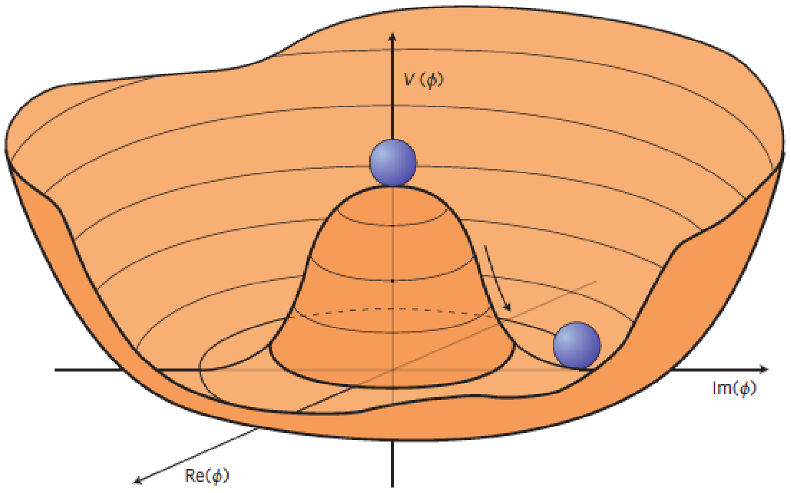


Figure 4.2.: Illustration of spontaneous symmetry breaking (SSB). The potential is symmetric for systems at their center, but the symmetry can be spontaneously broken by the transition to the energetically favorable state. Taken from Ref. [81].

bosons are massive particles. This gave motivation for a missing part of the SM: The Higgs sector. Brout, Englert, and Higgs introduced the Higgs theory in 1963 [79, 80], and with the discovery of the Higgs particle [3, 4], the theory was proven to be correct. The Higgs mechanism relies on the introduction of a weak isospin doublet, the so-called Higgs field

$$\Phi = \begin{pmatrix} \phi^+ \\ \phi^0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} \phi_1 + i\phi_2 \\ \phi_3 + i\phi_4 \end{pmatrix}, \tag{4.12}$$

with an electromagnetic-charged ϕ^+ and neutral component ϕ^0, each with a real (ϕ_1, ϕ_3) and an imaginary (ϕ_2, ϕ_4) component, adding up to four degrees of freedom. The Lagrangian $\mathscr{L}_{\mathrm{Higgs}}$ for the Higgs field reads

$$\mathscr{L}_{\mathrm{Higgs}} = \underbrace{(D_\mu \Phi)^\dagger (D_\mu \Phi)}_{\mathscr{L}^{\mathrm{kin}}_{\mathrm{Higgs}}} \underbrace{- \mu^2 \Phi^\dagger \Phi + \lambda (\Phi^\dagger \Phi)^2}_{V(\Phi)}, \tag{4.13}$$

where μ and λ are real, positive parameters and D corresponds to the covariant derivative. The latter part of the Lagrangian describes the Higgs potential, which is shown in Fig. 4.2. Due to the shape of the potential, the ground state of the system differs from its initial symmetry point. Following Hamilton's principle, the system can spontaneously break its initial symmetry to move to an energetically lower point of the potential. The Higgs potential and the process of spontaneous symmetry breaking (SSB) are illustrated in Fig. 4.2. The circular minimum around the symmetry point in the Higgs potential is defined as vacuum expectation value (VEV)

$$v = \sqrt{-\frac{\mu^2}{2\lambda}}. \tag{4.14}$$

Since the minimum is radially symmetric, the system can be described by one degree of freedom. After symmetry breaking, Φ cannow be rewritten as radial expansion

$$\Phi = \frac{1}{\sqrt{2}} \begin{pmatrix} 0 \\ v + h \end{pmatrix}, \tag{4.15}$$

with v as VEV and h as the excitation of the Higgs field around v, which is interpreted as Higgs boson H. By comparing Φ before symmetry breaking (Eq. (4.12)) and after (Eq. (4.15)), it can be noticed that symmetry breaking reduces the degrees of freedom by three. By the Goldstone theorem, this reduction leads to three massive Goldstone bosons [82].

To inspect these Goldstone bosons, Eq. (4.13) has to be retaken into account, and analogous to the sections before, the covariant derivative D_μ has to be defined to keep the invariance under $\mathrm{SU}(2)_L \times \mathrm{U}(1)_Y$ transformations of Ψ

$$D_\mu = \partial_\mu - i\frac{\sigma}{2} \cdot \mathbf{W}_\mu - ig'\frac{Y}{2}B_\mu, \tag{4.16}$$

introducing the gauge fields W_μ and B_μ, and using the pauli matrices σ, and the coupling constants of g and g' of gauge groups $\mathrm{SU}(2)_L$ and $\mathrm{U}(1)_Y$ respectively. By using Eq. (4.16) on Eq. (4.13), the Goldstone bosons $\mathrm{W}^\pm$ and Z can be identified as superposition of W_μ and B_μ with mass terms:

$$m_\mathrm{W} = \frac{1}{2}vg \quad \text{and} \quad m_Z = \frac{1}{2}v\sqrt{g^2 + g'^2}. \tag{4.17}$$

Due to $v \neq 0$ ensured by SSB, these terms illustrate how elegant the Higgs mechanism introduces masses for the $\mathrm{W}^\pm$ and Z bosons.

The introduced Higgs field Φ also interacts with spin $\frac{1}{2}$ particles, i.e., fermions, thereby generating their masses through Yukawa couplings [83]. The corresponding Lagrangian density can be written as

$$\mathscr{L}_\mathrm{Yukawa} = -y_f\, \bar{\psi}_f\, \Phi\, \psi_f + \text{h.c.}, \tag{4.18}$$

where y_f represents the strength of the fermion-Higgs coupling. In contrast to the lepton sector, where y_f is a scalar for each particle, in the quark sector, y_f is represented by a matrix, allowing for quark mixing. After SSB, the mass terms for fermions depend on y_f and the VEV and can be identified as $m_f = y_f \frac{v}{\sqrt{2}}$.

5. Experimental setup

high-energy physics (HEP) provides a broad landscape of applications for ML methods. The immense amount of collected data at HEP experiments contains highly complex data structures that cannot be analyzed by traditional methods. Moreover, powerful ML methods, such as those described in Part I, need to be applied to exhibit the structure of underlying physics processes.

HEP events are simulated by Monte Carlo (MC) methods; this is costly but provides truth information at every stage of the simulation process and, therefore, allows an optimal training of ML models for different tasks. These could range from classifiers for physics processes or objects to generative models for faster simulation processes, autoencoders for anomaly detection, or dedicated handling of systematic uncertainties.

However, after introducing theoretical foundations in Part II, this chapter will introduce the experimental environment in which the analyses for this work are embedded.

5.1. Large Hadron Collider

The LHC is a particle accelerator situated 100 m beneath the surface in Geneva, Switzerland. From the first conceptual plans to build the LHC in 1984 with the purpose of searching for the missing pieces in the SM: the top (t) quark, tau neutrino ν_τ, and most prominently the SM, the Higgs boson. It took 25 years until the first pp collisions started in 2008, marking the starting point for the Higgs hunting, which was finally discovered four years later, in 2012, by the ATLAS and CMS collaborations [3, 4]. This phenomenal discovery was honored by a Nobel Prize for Peter Higgs for his description of the Higgs mechanism (see Section 4.3) and the postulation of the Higgs particle linked to it.

With a circumference of 27 km, the LHC accelerates protons for collisions at a center-of-mass energy of $\sqrt{s} = 13.6\,\text{TeV}$ and is, therefore, a high-energy frontier in particle physics. To achieve such high energies, multiple pre-accelerators are needed. For instance, protons are first accelerated with the Linear Accelerator 3 (LINAC3), followed by the Proton Synchrotron (PS) and the Super Proton Synchrotron (SPS). The latter ones accelerate the protons to a maximal energy of 26 GeV and 450 GeV, respectively. The PS and the SPS are also used as accelerators for other nuclear physics experiments, like Isotope Separator On Line DEvice (ISOLDE) [84] or antimatter experiments like Antiproton Decelerator (AD) [85]. The SPS made significant discoveries in 1984 with the detection of the $\mathrm{W}^\pm$ and Z

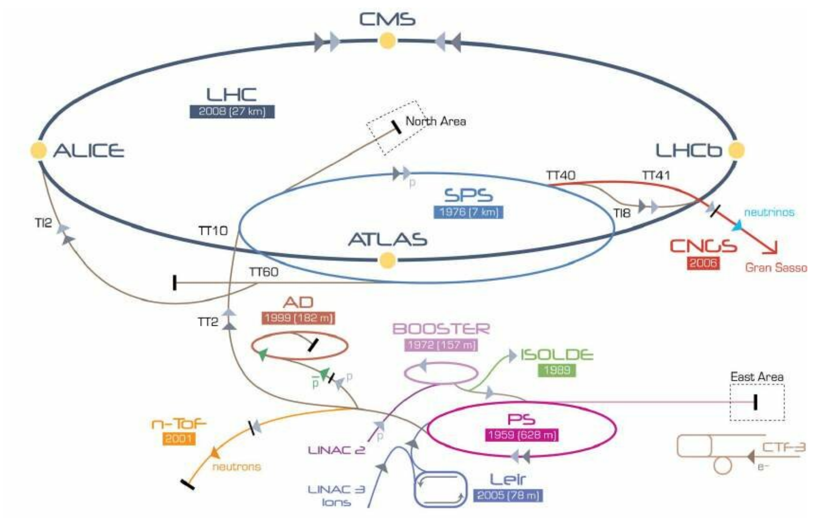


Figure 5.1.: Illustration of the accelerator complex at CERN. LINAC3, PS, and SPS are pre accelerators for LHC with its main experiments ATLAS, CMS, ALICE, and LHCb. Besides, several other experiments, like ISOLDE or AD. Taken from Ref. [95].

bosons in proton-antiproton collisions, as predicted by the SM, an achievement that was later recognized with a Nobel Prize [86, 87, 88, 89].

After the SPS, the protons are finally injected in the LHC and ramped to energies up to 6.8 GeV. The LHC has four collision points, each for one experiment, which are the two general-purpose experiments ATLAS [90] and CMS [91], the heavy iron experiment A Large Ion Collider Experiment (ALICE) [92] and lastly Large Hadron Collider beauty (LHCb) [93], which is specialized for bottom (b) quark and flavor physics. A complete picture of the accelerator infrastructure is illustrated in Fig. 5.1.

To fulfill the statistical needs of modern physics analysis, the LHC operates at a rate of roughly 40 MHz, corresponding to one collision every 25 ns and one petabyte of raw data per second. Since it is not possible to process this amount of data, the CMS experiment has, for instance, a two-level trigger system: The Level-1 trigger is hardware-based and filters interesting events, resulting in a rate of 100 kHz. The second trigger system is the high-level trigger (HLT), which is software-based to apply more advanced filters and reduce the event rate to 1 kHz. Since this is still a vast amount of data, traditional ML methods are no longer sufficient. Cutting-edge ML methods, like deep learning, need to be applied to filter events, identify patterns in events efficiently, or identify events that do not match the remaining data (outlier detections). However, with the High Luminosity upgrade set to begin in 2028, the luminosity is expected to increase by a factor of 7.5 [94], providing even more data to analyze, allowing to analyze deeper patterns. The following chapter gives an overview of how the detector collects the data for this analysis procedure.

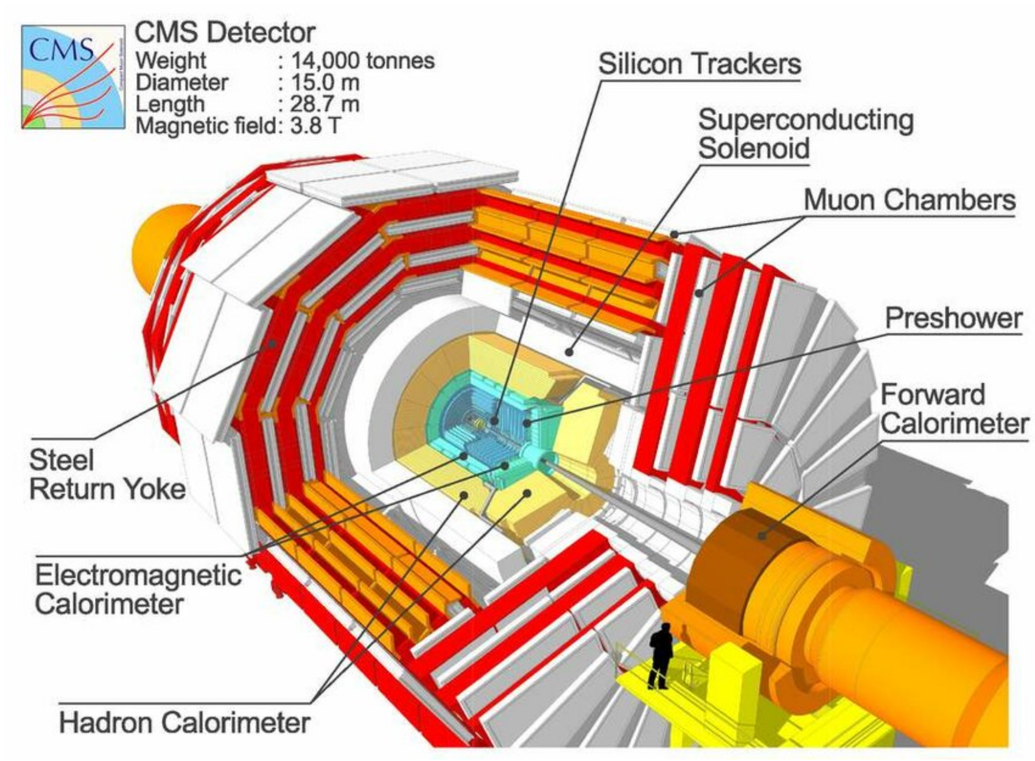


Figure 5.2.: Illustration of the detector of the Compact Muon Solenoid (CMS) collaboration. main parts are the tracker system, preshower and ECAL, followed by the HCAL, the steel return yoke and the muon system. Taken from Ref. [95].

5.2. CMS detector

The CMS detector is one of the two general-purpose detectors at the LHC. With a weight of 14 000 tons, approximately 15 meters in diameter and 21 meters in length, it was an engineering masterpiece to install the detector roughly 100 meters underground at the LHC collision point.

The detector consists of a main barrel centered around the beam axis and two endcaps, providing a nearly complete 4π coverage for detecting particle candidates produced at pp collisions provided by the LHC. The CMS experiment utilizes multiple specialized subdetectors to measure the properties of collision products. All information is finally combined to correctly identify the type of particle. These individual layers are built around the beam axis from inside to outside: tracker, electromagnetic calorimeter (ECAL), hadron calorimeter (HCAL), a superconducting solenoid, steel return yoke, and finally, the muon chambers. A graphical overview of the CMS detector is given in Fig. 5.2, and a detailed description can be found in Ref. [91].

The coordinate system of the CMS detector is defined around its interaction point. The beam axis defines the z-axis while x and y span a transverse plane, while the x-axis points to the center of LHC and the y-axis to the surface. Due to the cylindrical shape of the detector, it is preferable to switch to cylindrical coordinates

$$\mathbf{p} = \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} = \begin{pmatrix} p \sin\theta \cos\phi \\ p \sin\theta \sin\phi \\ p_z \end{pmatrix}, \tag{5.1}$$

with polar angle θ and azimuth ϕ. Since the CMS detector is symmetric in ϕ, physics is invariant in the transversal plane, therefore, the transversal momentum is defined as

$$p_\mathrm{T} = \sqrt{p_x^2 + p_y^2}. \tag{5.2}$$

To describe the angular properties in θ direction, the pseudorapidity η is defined as

$$\eta = -\ln\left(\tan\frac{\theta}{2}\right). \tag{5.3}$$

Its advantage over θ is that angular differences $\Delta\eta$ are a Lorentz invariant quantity. The following sections explain the single detector components and their purpose.

5.2.1. Components

5.2.1.1. Silicon tracker

The purpose of the silicon tracker [96] is to measure high-resolution energy depositions (hits), which are employed to reconstruct the trajectory of charged particles. Since the tracker is located in the magnetic field of the superconducting solenoid, the curvature of the particle tracks can be used to identify the charge of the particle. Furthermore, the intersection of multiple tracks is used to reconstruct decay vertices.

This is important for vertex reconstruction, concluding on the particle's mass and momentum. Hence, tracking near any vertex is especially important to increase the resolution of the momentum measurement.

The silicon tracker is the innermost part of the CMS detector and is subject to heavy radiation effects. Therefore, the tracker system must be a highly radiation-resistant detector and simultaneously acquire the best possible measurements. These conditions are met by using radiation-resistant silicon for all three parts of the tracking system: the pixel, the strip detector, and the barrel detector. In addition, the subdetector is operated at roughly $-10°C$, which reduces the leakage current from defects and trap states in the crystal lattice.

The innermost pixel detector is located around the beam pipe with a radius $r = 16\,\mathrm{cm}$. It consists of four layers built up by silicon pixels of size $100 \times 150\,\mu\mathrm{m}^2$, allowing for a high resolution. In contrast, the outer part, the strip detector, is built by three layers of silicon strips of size $10\,\mathrm{cm} \times 80\,\mu\mathrm{m}$ on each side. By tilting the strip layers relative to each other, the strip detector allows for precise track reconstruction. Analogously, the tracker system in the endcaps is built of three layers with the same-sized silicon strips, allowing the reconstruction of tracks in forward direction.

5.2.1.2. Electromagnetic calorimeter and preshower

The ECAL [97] surrounds the silicon tracker and aims for and measures the energy of electrons, positrons, and photons. Since muons are minimum ionizing particles, they leave a trajectory in the tracker system, but deposit no significant energy depositions in the ECAL. The energy measurement is done in a destructive way, meaning that the electromagnetic particle interacts with the detector material and starts an electromagnetic shower of pair production $\gamma \longrightarrow e^+e^-$ and bremsstrahlung $e^\pm \longrightarrow \gamma e^\pm$.

The ECAL is a homogenous calorimeter of scintillating lead-tungsten (PbWO4) crystals covering $26X_0$. Avalanche diodes capture scintillated photons from the showering process to produce an electrical signal that can be calibrated to the deposited energy for the readout electronics.

To distinguish photons originating from bremsstrahlung and photons from $\pi^0 \longrightarrow \gamma\gamma$ decays, a so-called preshower detector is installed before the ECAL. This sampling calorimeter consists of two layers of lead and three layers of active silicon covering three radiation lengths $3X_0$.

5.2.1.3. Hadron caloriemeter

The HCAL [98] is wrapped around the ECAL and aims for energy measurement of charged and neutral hadrons, e.g., pions, neutrons, protons, and kaons. Regarding neutral hadrons, it is noteworthy that the HCAL is the only subdetector able to interact with these candidates.

In contrast to the ECAL, the HCAL is a sampling calorimeter consisting of brass as passive material and plastic scintillators as active material. First, the passive material is employed to cause strong interactions with the hadrons, leading to a hadronic shower. Second, the active material is utilized to capture the electromagnetic component of the hadronic shower, induced by $\pi^0 \longrightarrow \gamma\gamma$ decays, to achieve an energy measurement.

Since hadronic showers are more expansive than electromagnetic ones, they also undergo considerable fluctuation regarding their size. To mitigate this circumstance, brass is chosen as a passive material since a high atomic number Z leads to higher interaction probabilities and, therefore, to relatively smaller shower sizes. In addition, the size of the HCAL is chosen to cover 5.82 nuclear interaction lengths λ_n to cover nearly all occurring hadronic showers.

5.2.1.4. Superconducting solenoid and return yoke

The superconducting solenoid is a central feature in the CMS detector, contributing to the name of the collaboration. It is located around the HCAL and operated at $\approx 4.5\,\mathrm{K}$, allowing it to provide a magnetic field inside the coil of $3.8\,\mathrm{T}$. Its purpose is to force charged particles into curved tracks, allowing for charge identification and momentum measurements of these particles. The steel iron yoke is installed to shape the magnetic field while acting in parallel as an absorber, allowing only muons to enter the muon system, which is described in the next section.

5.2.1.5. Muon system

Due to its compact form, the muon system of the CMS detector [99, 100] is a characteristic feature of the detector. Since muons leave a track in the tracker system but interact merely with the other subdetectors, a dedicated muon system is installed, allowing for reliable muon identification and triggering. This system is especially valuable in one of the channels for the Higgs boson observation $H \rightarrow 4\mu$. The system consists of drift tubes (DTs) filled with ionizing gas. While crossing the DTs, muons ionize this gas, and anodes located in the center of the tubes measure the ionized electrons. This resolution is comparably lower than that of one from the tracker, but more efficient regarding material costs, which is a more significant factor for the external detector parts. In contrast to the barrel region, the endcap regions use cathode strip chambers (CSCs). By using orthogonal layers of anode wires, spatial and timing information can be inferred at a higher precision as the DTs. This is necessary since a higher rate of muons is expected at higher η.

Both the DTs in the barrel and the CSCs in the end-cap regions are installed in an alternating system with resistive plate chambers (RPCs). These are built of a gas volume embedded between resistive plates. When a muon ionizes the gas, electrons initiate an avalanche, which is detected as a signal by readout strips. RPCs deliver precise timing resolution of about 3 ns and are therefore combined with DTs and CSCs.

5.3. Track and vertex reconstruction

The track and vertex reconstruction is primarily based on the information collected in the silicon tracker; see Section 5.2.1.1. This process aims to identify the tracker hits from each charged particle and apply a fit to its trajectory through the detector. To start this process, hits from neighboring layers with at least two hits are clustered and used as initial seeds to start the track finding. Based on these seeds, the track of the candidate is then extrapolated across the tracker using iterative Kalman filters [101]. In the second step, the so-called track finding, additional hits are sequentially added to the track. Here, the best hit candidates are added to the track in an inside-out order. This is repeated to improve consistency, but from outside to inside. Finally, all hits associated with the track candidate are used to identify a final track using a fitting procedure, accounting for inhomogeneities in the magnetic field or energy losses through material interactions. All hits used for this candidate are removed and not used for the next track candidate. A detailed description of the tracking algorithm can be found in Ref. [102, 103].

Identifying the vertex for a given set of tracks is done by extrapolating them towards the beam axis. An annealing algorithm [104] is used to cluster all tracks in an event to apply a dedicated vertex fit using the tracks in the corresponding cluster. The primary vertex (PV) of an event is finally identified as the vertex with the highest sum of transverse momenta p_{T}^2.

5.4. Particle-flow algorithm

The unique approach for the Particle Flow (PF) algorithm [105, 106] of the CMS experiment is to combine information from all subdetectors to identify the type of particle instead of taking only isolated subdetector information into account. Since particle identification gives the foundation for higher-level reconstruction steps, the PF algorithm is particularly important for the performance of the event reconstruction.

The PF algorithm classifies particles into five categories: muons, electrons, photons, charged hadrons, and neutral hadrons. Therefore, the data used by PF is provided by the tracker, or more precisely, the reconstructed trajectories (explained in Section 5.3) of charged particles, the ECAL information for electrons and photons, HCAL information for hadrons and finally signals from the muon chambers.

In the first step, single energy deposits in calorimeter cells are clustered. Therefore, local maxima in energy deposits are identified and used as seeds to cluster neighboring cells together and combine them for an energy estimate. Next, these clusters are linked to reconstructed tracks based on their spatial proximity and are only kept if the linking corresponds to the closest connection. In turn, these linked objects are summarized in blocks, where a block represents reconstructed objects rising from the same underlying particle, for example, an isolated muon would correspond to a block with fewer entries, but a hadronic shower would correspond to a more convoluted block.

Finally, each block is sequentially processed to determine the most likely particle, while easy-to-reconstruct particles are checked first. Each time a candidate is assigned to a particle class, the associated objects are removed from the corresponding block. **Muons** can be identified by a trajectory in the inner tracker and by hits in the muon system. Muons are categorized into three different types. A muon corresponds to a tracker muon if a trajectory extrapolated from the tracker matches a hit in the DTs or CSCs. If merely hits from the muon system are used to fit a muon track, following the procedure in Section 5.3, this is referred to as a standalone muon. The two scenarios combined, when tracker information is combined with hits in the muon system to fit a muon trajectory, this is referred to as a global muon.

Photons require energy depositions in the ECAL. Since they do not carry electric charge, these energy depositions cannot be linked to any reconstructed tracks. In contrast, **electrons** require a track that is linked to the shower in the ECAL. While the direction for photons is determined based on the shower information, the direction of electrons is determined by the reconstructed trajectory. Electrons, positrons, and photons are reconstructed in the same step.

All clusters in the HCAL without any links to tracks are considered to originate from **neutral hadrons**. Analogously to photons, the energy and direction are determined based on the shower information. Lastly, tracks linked to HCAL clusters and remaining tracks are assigned to **charged hadrons**. Momentum and direction can be determined based on the track.

The final list of particles serves as a basis for further high-level reconstructions, including tau lepton reconstruction, jet clustering (see Section 5.5.1), and calculation for missing transverse momentum $p_{\mathrm{T}}^{\mathrm{miss}}$, which is explained in the next step.

Since the initial pp collision contains no transverse momentum, $p_{\mathrm{T}}^{\mathrm{miss}}$ is expected to be zero. However, due to neutrinos rising from weak interactions, detector calibration, and momentum resolution, or potential new physics, $p_{\mathrm{T}}^{\mathrm{miss}}$ is usually non-zero. By summing over all particles i, $p_{\mathrm{T}}^{\mathrm{miss}}$ is calculated by

$$p_{\mathrm{T}}^{\mathrm{miss}} = -\sum_i \boldsymbol{p}_{\mathrm{T},i}. \tag{5.4}$$

5.5. Jets

Color-charged partons, i.e., quarks or gluons, can be produced during the hard scattering process of the pp collision. Due to confinement, stating that color-charged partons do not occur in isolated states, these partons start the process of parton showering, meaning that they trigger a cascade of additional gluons and quarks. At the beginning of the showering process, this happens in a collinear way, meaning that the radiated gluons have small angles relative to the initial particle. After the decay products undergo a certain energy scale, the hadronization process sets in, and they start forming stable and color-neutral hadrons. In HEP, such a collimated stream of particles is referred to as a jet. The jet reconstruction, i.e. the jet clustering, is part of the PF algorithm (see Section 5.4) and uses tracker information of charged constituents as well as ECAL and HCAL information. Based on this information, the full four-momentum of the jet can be derived. The subsequent sections will describe the most essential properties of jet reconstruction in more detail.

5.5.1. Jet clustering

The CMS experiment uses the anti-k_T algorithm [107, 108] in its PF framework to cluster particles to a jet. To ensure that a jet cluster algorithm is stable and reliable, there are two requirements it has to fulfill:

- **Collinear safety** ensures that the clustering algorithm remains unaffected when a hard particle splits into multiple collinear fragments. Such a splitting should not

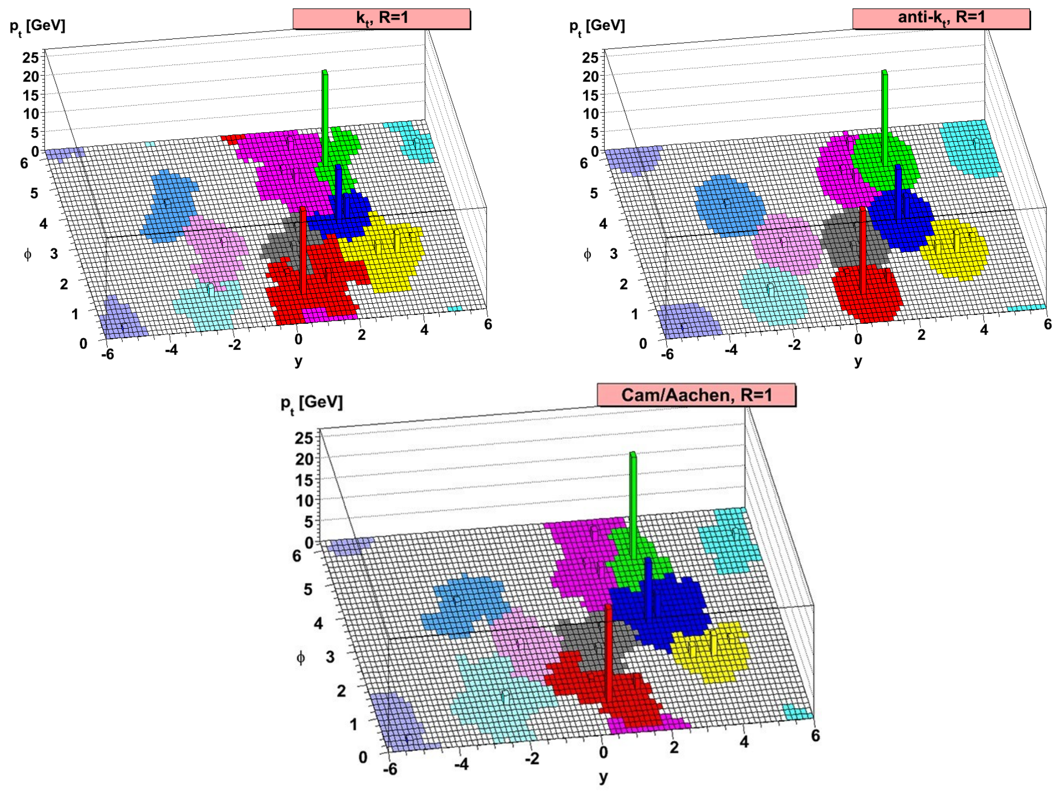


Figure 5.3.: Comparison between the k_T (upper left), anti-k_T (upper right) and Cambridge-Aachen (CA) (lower) algorithm. The plots show the cluster results of the corresponding algorithms by showing the p_T as a 2D histogram dependent on the constituent coordinates in ϕ and y. Taken from Ref. [109].

result in the reconstruction of multiple jets instead of one correctly reconstructed jet.

- **Infrared safety** guarantees that the algorithm is insensitive to low-energy (soft) gluon emission. This means that adding a soft gluon should not alter the clustering outcome, preventing two distinct jets from being incorrectly merged into one.

The k_T [107, 108], anti-k_T [109], and Cambridge-Aachen (CA) [110] algorithms are commonly used sequential cluster algorithms. Here, the clustering is done based on the distance d_{ij} between two constituents i and j. To do so, d_{ij} is computed for all constituents

$$d_{ij} = \min\left(p_{T,i}^{2k}, p_{T,j}^{2k}\right)\frac{\Delta R_{ij}^2}{R^2} \tag{5.5}$$

$$\text{with } \Delta R_{ij}^2 = (\Delta\eta_{ij})^2 + (\Delta\phi_{ij})^2 \tag{5.6}$$

with the angular distance ΔR_{ij}^2 between i and j, and the radius parameter R, which is usually set to 0.4. The k parameter is dependent on the clustering algorithm, as clarified below. The two constituents with the smallest d_{ij} are then used as the initial seed and merged into a new combined object. Based on this object, the algorithm sequentially computes d_{ij} and adds the next constituent to the jet object until $p_{T,i}^{2k}$ is smaller than the remaining d_{ij}.

The k_T algorithm sets $k = 1$, leading to soft particles being clustered first and, therefore, are more dominant for the shower reconstructions. While this results in a good representation of the jet substructure, it also introduces considerable fluctuations in its shape. In contrast, the anti-k_T algorithm uses $k = -1$, which clusters hard constituents first and increases their impact on the reconstruction. Consequently, this leads only to slight fluctuations in the shower shape but to a less efficient depiction of the jet substructure. The CA algorithm clusters p_T independently by applying $k = 0$. By relying only on the angular relations, this algorithm depicts the substructure best out of the three algorithms.

A comprehensive overview of the three algorithms is given in Table 5.1. Additionally, Fig. 5.3 shows a comparison of the jet shapes and their momenta. These plots show the jet momenta as a 2D histogram dependent on ϕ and y. These studies show especially that the anti-k_T algorithm tends to shape clusters spherical around their seed with the highest p_T. In contrast, k_T and CA are able to depict the substructure more efficiently.

Cone algorithms should also be mentioned for completeness. Compared to sequential algorithms, which cluster in momentum space, cone algorithms cluster in $\eta - \phi$ space. For this, a fixed-sized jet cone, such as the SISCone algorithm, is used to add particles sequentially to the jet. The main drawback of cone algorithms is that they suffer from not being collinear and infrared safe. A detailed overview and comparison of jet algorithms are given in Ref. [112].

5.5.2. Jet momentum corrections

Accurate object reconstruction requires that the measured jet momenta closely match the true momenta of their originating particles. However, various effects, such as pileup, non-linear detector responses, and mismodeling in MC simulations, can distort the reconstructed jet momentum. To address these discrepancies, the CMS experiment applies a factorized jet-momentum corrections (JMC) approach, in which multiple corrections are applied sequentially in a fixed order.

Algorithm	k	Domination	Fluctuations	Substructure
k_T	1	Soft particles	Considerable	Good
Cambridge-Aachen	0	p_T-independent	Moderate	Best
anti-k_T	-1	Hard particles	Slight	Worst

Table 5.1.: Comparison between the k_T, anti-k_T and CA algorithms. Adapted from Ref. [111]

The first stage of corrections addresses pileup effects, which arise from additional pp collisions occurring near the considered collision. As these contributions increase the jet momentum, pileup correction factors are derived by comparing MC simulations with and without pileup and are applied as functions of the median energy density in an event ρ, the jet pseudorapidity η, and the jet transverse momentum p_{T}.

The second stage corrects for the non-linear responses of individual detector components. Since the CMS detector is uniform in the azimuthal angle ϕ, these corrections are applied as functions of η and p_{T} only.

Finally, residual corrections are applied to account for any remaining discrepancies between data and simulation that the previous steps did not cover. These corrections are generally small and are applied in two steps: one dependent on η and the other on p_{T}.

The complete set of corrections is applied universally to all jet types, ensuring consistency in jet momentum reconstruction across the CMS experiment. A more detailed explanation of the individual corrections is provided in Ref. [113, 114].

The JMC is essentially a scaling of the four-momentum vector of a jet. Since this scaling is typically determined based on p_{T}, this work will stick to the technically correct term as JMC. However, since the jet energy is also corrected, other resources refer to these corrections as jet energy corrections.

5.5.3. Bottom quark jets

Compared to light-flavor jets, which originate from up, down, charm, and strange quarks, jets from b quarks play a special role in many physics analyses. On the one hand, many searches target signatures involving t quark decays, such as those discussed in Section 7.2.1. Due to the structure of the Cabibbo–Kobayashi–Maskawa (CKM) matrix, which favors decays within the same generation, and the large mass difference between the top and bottom quark ($m_{\mathrm{t}} = 172.76$ GeV vs. $m_{\mathrm{b}} = 4.18$ GeV), the t quark decays almost instantaneously via $\mathrm{t} \longrightarrow \mathrm{b}W^{+}$.

On the other hand, b quarks play a crucial role in H boson decays. Since b quarks are the heaviest on-shell particles H can decay into, the decay mode $\mathrm{H} \longrightarrow \mathrm{b\bar{b}}$ has the highest branching ratio among all H decays. These examples illustrate the frequent presence of b quarks in physics signatures, making it essential for many analyses to distinguish between jets originating from b quarks and those from light-flavor quarks.

While the CKM element $|V_{tb}|$ favors transitions from b to t quarks, the large mass difference suppresses transitions in this direction. As a result, the dominant b quark decay channels are the off-diagonal transitions $\mathrm{b} \longrightarrow \mathrm{uW}^{-}$ and $\mathrm{b} \longrightarrow \mathrm{cW}^{-}$, both of which are suppressed by the CKM matrix.

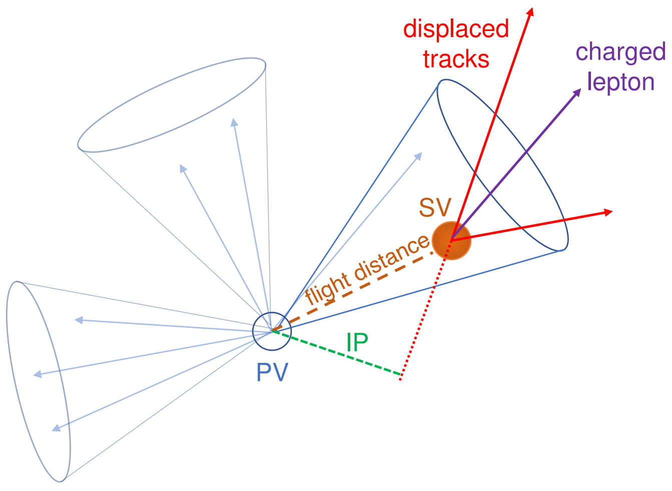


Figure 5.4.: Illustrative comparison between a b jet (right cone) with light flavor jets (other cones). Displaced tracks (solid red lines) between the primary vertex (PV) and secondary vertex (SV) may occur due to the flight distance of the b quark. The flight distance projected on the z-axis is denoted as impact parameter (IP) distance. Weak interactions may lead to the presence of charged leptons (purple) and p_T^{miss}. Taken from Ref. [115].

Consequently, b quarks have a relatively long lifetime compared to lighter quarks, leading to a decay length of approximately $c\tau \approx 450\mu$m and a displaced secondary vertex (SV). The displacement is described by the impact parameter (IP), which corresponds to the projection of the flight distance onto the z-axis. Figure 5.4 provides an illustration of this jet signature.

Additionally, the hadronization process of b jets can contain weak decays, leading to the production of neutrinos within the jet. Since neutrinos escape the detector, this results in an undetectable energy fraction, contributing to missing transverse energy p_T^{miss}. These unique properties of b jets necessitate dedicated energy corrections for b jets to reconstruct their energy, as further discussed in Part IV.

5.6. Monte Carlo methods and event simulation

The simulation of pp collisions is a fundamental part of the work within the CMS collaboration. Since the underlying physics processes in events recorded in actual collisions are unknown, all considered processes are simulated. Given that the true information of the underlying process is available, these simulated events can be used as datasets for optimizing ML models. This thesis focuses on the model optimization stage, laying the groundwork for analyses that compare simulation-based model results with data-based results.

Such simulations are based on Monte Carlo (MC) generators, computational tools that use random number sampling to simulate processes based on stochastic quantum mechanics. The first interaction that needs to be simulated is the quantum mechanical amplitude for a specific outcome of the hard scattering process. This amplitude is referred to as matrix element (ME) and is calculated by `MadGraph` [116] or `Powheg` [117]. In principle, there are infinitely many paths that have to be taken into account for the simulation of the hard scattering process, but this can be approximated by an expansion in a series ordered by α_s. Here, hard processes are simulated at the lowest order of α_s, referred to as leading order (LO) in QCD, and next to leading order (NLO), which includes one additional gluon loop or radiation. Decays of heavy particles, like massive bosons or t quarks, are simulated by a dedicated submodule named `MadSpin` [118].

In the next step, the showering and hadronization of partons into lighter, stable particles are simulated by `PYTHIA` [119]. Such stable particles are the foundation for the measurement by the CMS detector. There are two programs for the detector simulation: The first one, `Geant4` [120], simulates direct particle-detector interactions in a detailed but computationally costly way. `Geant4` simulations are standard for physics analyses. As an alternative, `DELPHES` [121] is a detector simulation program optimized for simulation speed: to achieve this, generalized response functions are utilized instead of detailed physics tracking. `DELPHES` is the optimal simulation candidate when large datasets for ML studies, rather than the highest possible accuracy for each event, are required.

Part III.

Taylor Coefficient Analysis

6. Neural network introspection tools

6.1. Motivation and landscape

The strength of NNs are their ability to identify complex patterns within an input space X. In turn, they require a large number of learnable parameters to effectively capture these patterns. A large number of parameters may obscure the impact of specific features or sets of features on the final prediction. As a result, the reasoning behind NNs is often non-trivial, and NNs are often regarded as black box models.

The lack of interpretability can undermine the trust in NN predictions, particularly when NNs are applied in sensitive decision-making contexts, in which a good understanding of both the input space and the decision-making process is crucial. In HEP, a thorough understanding of the NN input space is essential for producing sustainable and trustworthy results. Therefore, reliable estimates and confidence interval (CI) with high frequentist coverage are essential for drawing robust conclusions. Consequently, there is a growing focus on NN reasoning and the development of what is often referred to as explainable AI.

A widely used, though more traditional, approach to address this issue is the principal component analysis (PCA) [122]. In this method, the input vector $\mathbf{x}$ is decomposed into uncorrelated axes, known as its principal components, where the eigenvalues can be used to assess the amount of information along each axis. However, the main limitation of PCAs is that they are restricted to the input space itself and do not account for how the model utilizes its inherent features.

An alternative method is Leave One Out (LOO) [123], where each feature is iteratively removed from the feature set, and the model is retrained without this feature. The performances of the retrained models are then compared to estimate the importance of the excluded features. While the LOO method provides insight into the relevance of a given feature for the NN prediction, it is computationally expensive and time-consuming. Additionally, there is no guarantee that the individual retrainings will converge to optimal solutions. In the context of HEP, this method may face challenges, especially when some features are correlated with other features.

The SHapley Additive exPlanations (SHAP) method [124], derived from game theory, assigns each feature in X a contribution to the output of the model, analogous to how players contribute to the payout of a game. In this paradigm, the features are treated as players, and the model output as payout. Marginal distributions are computed for all subsets α of the feature space X and a Shapley value is computed as the weighted average of the marginal contributions of α. The Shapley values are then used to weight the marginal distribution of α and reconstruct the model output based on all weighted marginal distributions.

A more detailed and broader overview of introspection methods for NNs can be found in Ref. [125]. While both LOO and SHAP are model-agnostic approaches that yield promising results, they both rely on empirical methods to analyze NN predictions. In contrast, approaches that assess feature impact based on analytical methods offer the advantage of being mathematically better justified.

6.2. The Taylor Coefficient Analysis

The Taylor Coefficient Analysis (TCA), first presented in Ref. [6, 126], determines the impact of a feature or set of features based on its gradients. It offers an analytical alternative to the approaches mentioned in Section 6.1. Since an NN is an analytical function

$$\Omega_{\hat{y}}(\mathbf{x}) : \mathbf{x} \mapsto \mathbf{y}, \tag{6.1}$$

a Taylor series (TS) can be used to approximate $\Omega_{\hat{y}}$. Using a development point $\mathbf{x} \in X$ and an evaluation point $\mathbf{a}$, this series reads

$$\begin{aligned}
\Omega_{\hat{y}}(\mathbf{a}) = {} & \Omega_{\hat{y}}(\mathbf{x}) \\
& + \sum_i \underbrace{\left.\frac{\partial \Omega_{\hat{y}}(\mathbf{x})}{\partial x_i}\right|_{\mathbf{x}}}_{t_{x_i}(\mathbf{x})} (a_i - x_i) \\
& + \sum_{i,j} \underbrace{\left.\frac{1}{2}\frac{\partial^2 \Omega_{\hat{y}}(\mathbf{x})}{\partial x_i \partial x_j}\right|_{\mathbf{x}}}_{t_{x_i,x_j}(\mathbf{x})} (a_i - x_i)(a_j - x_j) \\
& + \sum_{i,j,k} \underbrace{\left.\frac{1}{6}\frac{\partial^3 \Omega_{\hat{y}}(\mathbf{x})}{\partial x_i \partial x_j \partial x_k}\right|_{\mathbf{x}}}_{t_{x_i,x_j,x_k}(\mathbf{x})} (a_i - x_i)(a_j - x_j)(a_k - x_k) \\
& + \mathcal{O}((\mathbf{x} - \mathbf{a})^4).
\end{aligned} \tag{6.2}$$

In this context, the prefactors and the derivatives of $\Omega_{\hat{y}}$, evaluated at $\mathbf{x}$, are identified as Taylor coefficients

$$t_\alpha(\mathbf{x}) : \mathbf{x} \to \mathbb{R}, \tag{6.3}$$

for a feature, or a set of features α. The length of α determines the order in the TS the t_α belongs to. Analogous to SHAP, the coefficients $\alpha = \{x_i\}$ corresponding to the first order in the TS are refered to as first-order coefficients, $\alpha = \{x_i x_j\}$ and $\alpha = \{x_i x_j x_k\}$ as the second and third order coefficients respectively.

First-order coefficients t_{x_i} quantify the slope of the NN decision plane regarding x_i. Consequently, they obtain large values in regions where x_i has a high impact on the NN prediction. Since a good decision boundary in classification tasks is typically characterized by a steep transition, first order t_α highlights this area.

Second-order coefficients $t_{x_i x_j}$ indicate the curvature of the decision plane and reveal nonlinear dependencies on α. For nonlinear decision planes, the curvature is also near the decision boundary. While curvature terms $\alpha = \{x_i x_i\}$ indicate curvature in one dimension, $\alpha = \{x_i x_j\}$ terms highlight curvature in the $x_i x_j$-plane.

Third-order coefficients $t_{x_i x_j x_k}$ highlight changes in the curvature of the decision plane. These coefficients tend to peak in regions exhibiting a sensitivity to α beyond quadratic order.

In conclusion, different orders in t_α highlight regions in the decision plane sensitive to changes in t_α. The ability to catch the sensitivity of higher-order impacts in α analytically is a unique feature of the TCA. In regions with a constant decision plane, all t_α are expected to be zero. The TCA thus quantifies local feature sensitivity in X, making it particularly suitable for identifying impactful regions in X. Consequently, the dataset used to evaluate $t_\alpha\,(X)$ must be chosen to cover these regions of interest. If a t_α does not peak within the dataset at all, then α does not contribute to the NN prediction in the phase space covered by the dataset.

To apply the TCA, its properties must be well understood. For instance, the choice of activation functions is crucial, since higher-order coefficients in t_α require activation functions that are differentiable up to the corresponding order. This requirement can be fulfilled by applying $\tanh(\cdot)$, but not by using ReLU or Leaky ReLU. Since the derivative is linear

$$\frac{\partial}{\partial x_i}\Omega_{\hat{y}}(c\mathbf{x}) = c\frac{\partial}{\partial x_i}\Omega_{\hat{y}}(\mathbf{x}), \tag{6.4}$$

all x_i should be scaled to the same size. This can be achieved by standardizing the input features to zero mean and unity variance

$$x_i \mapsto \frac{x_i - \mu}{\sigma}, \quad \text{with} \quad \mu = \mathbb{E}[x_i], \quad \sigma = \sqrt{\mathrm{Var}(x_i)}. \tag{6.5}$$

This is crucial to mitigate biases from distorted feature spaces. Furthermore, the Schwarz theorem applies to Taylor coefficients

$$t_{x_i,x_j} = \frac{\partial\Omega_{\hat{y}}(\mathbf{x})}{\partial x_i, x_j} = \frac{\partial\Omega_{\hat{y}}(\mathbf{x})}{\partial x_j, x_i} = t_{x_j,x_i} \tag{6.6}$$

which ensures that curvature and higher-order terms are symmetric regarding the order of t_α.

Based on the definition of t_α and the characteristics of the TCA, the TS is constructed for each event $\mathbf{x}_i \in X \subseteq \mathbb{R}^n$, with $i = 1, \dots, N$, to compute $t_\alpha(\mathbf{x}_i)$. A summary statistic can be derived to quantify the mean impact of α, e.g.

$$\langle t_\alpha \rangle = \sum_i^N \frac{|t_\alpha(\mathbf{x}_i)|}{N}, \tag{6.7}$$

across X. Here, it is essential to understand that the choice of X can significantly impact $\langle t_\alpha \rangle$. For instance, if $\Omega_{\hat{y}}$ is overfitted to X^{train}, then $t_\alpha(X^{\text{train}})$ may be biased compared to $t_\alpha(X^{\text{val}})$, even if both follow the same distribution. While different distributions lead to different t_α, this property conversely allows for checking whether two distributions look similar in terms of $\Omega_{\hat{y}}$, which could be utilized to flag overfitting, for example.

Additionally, Eq. (6.2) illustrates that each coefficient t_α is weighted by a different power of distance terms $(x_i - a_i)$, corresponding to the order in the expansion. As a result, the contribution of higher-order t_α increases (decreases) for $\mathbf{x}$ where the corresponding distance terms are large (small), turning direct comparisons across different orders difficult.

In the scope of this thesis, a dedicated package `TaylorAnalysis` has been developed and published [127] for the programming language `Python` [128]. `TaylorAnalysis` provides a user-friendly and efficient computation of t_α and is utilized for the studies presented in the following sections. It is set up for NNs implemented using the `PyTorch` library [23] and utilizes the automatic differentiation mechanism of `PyTorch` to calculate t_α up to the third order. Furthermore, `TaylorAnalysis` is optimized in its calculations: all coefficients are grouped in a tree structure, allowing to calculate t_{x_0,x_0,x_k} in a single step for all k. Additionally, the package is optimized for performance, such that all t_α can be computed in parallel on GPU devices. These implementations guarantee a performant and efficient calculation of t_α in data-intensive fields. The flexibility of `TaylorAnalysis` in its application to fundamentally different NN architectures is demonstrated in the following chapters.

7. Applications of the Taylor Coefficient Analysis

7.1. Toy dataset

This chapter demonstrates the TCA based on two simple 2 D toy datasets. Both are set up with a Gaussian distributed signal S and background sample B of 100 000 events each. For S and B, the means for the samples are $\mu_\mathrm{S} = -\mu_\mathrm{B} = (0.5, 0.5)$. The first dataset is called the baseline dataset: here, S and B have no correlations. In contrast, the second dataset introduces a positive correlation for S and a negative correlation for B. This dataset is referred to as the correlated setup. Covariance matrices for both scenarios are shown in Eq. (7.1). The baseline dataset is depicted on the left, and the correlated dataset is shown on the right in Fig. 7.1.

$$\begin{array}{cc} \textbf{Baseline} & \textbf{Correlation} \\ \mathrm{cov_S} = \begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix} & \mathrm{cov_S} = \begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix} \\ \mathrm{cov_B} = \begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix} & \mathrm{cov_B} = \begin{pmatrix} 1 & -0.5 \\ -0.5 & 1 \end{pmatrix} \end{array} \tag{7.1}$$

Two MLPs with the same architecture are trained to separate S from B. Their architecture consists of two input features $h^{(0)} = (x_1, x_2)^\mathrm{T}$, one hidden layer of 100 neurons, and a single output neuron. To compute higher orders of t_α, the activation function is set to tanh, except for the last layer, where a sigmoid function is used. The training is performed with the ADAM optimizer to minimize the CE with an initial learning rate of $\eta = 10^{-4}$.

During the training, the development of the $\langle t_\alpha \rangle$ is tracked, as depicted in Fig. 7.2. It can be observed that the development of $\langle t_\alpha \rangle$ is equivalent for both trainings up to epoch 120. The losses and the first order $\langle t_\alpha \rangle$ converge to the values given in Table 7.1. It holds

$$\langle t_{x_1} \rangle \approx \langle t_{x_2} \rangle, \quad \langle t_{x_1,x_1} \rangle \approx \langle t_{x_2,x_2} \rangle, \tag{7.2}$$

which coincides with the $x_1 x_2$ symmetry in both datasets. The development of $\langle t_\alpha \rangle$ based on the correlated dataset continues on a plateau, coincidentally with a kink in the loss curves. Starting at epoch 300, the second-order coefficients, especially t_{x_1,x_2}, gain importance, while the first-order coefficients bend slightly.

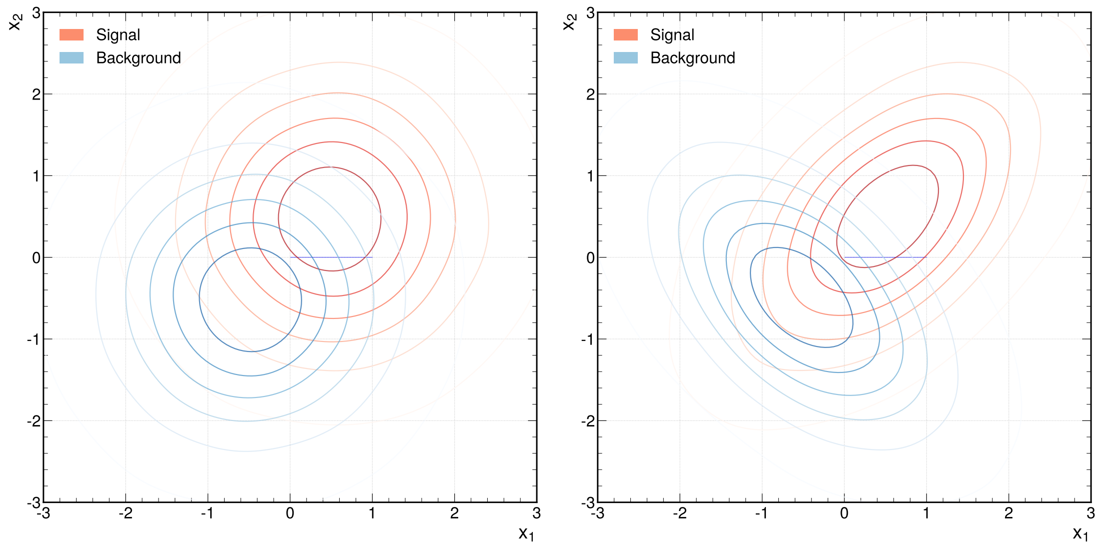


Figure 7.1.: Distributions of toy datasets. The signal sample S is shown in red and the background B in blue. The baseline dataset (left) consists of two overlapping Gaussians without correlation. The correlated dataset introduces positive and negative correlations for S and B, respectively.

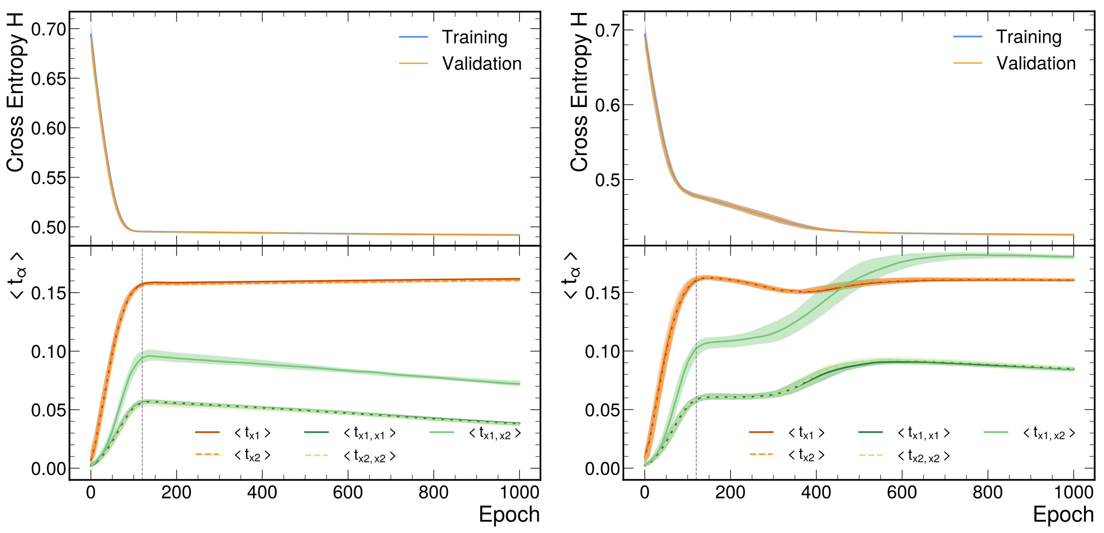


Figure 7.2.: Evolutions of losses for baseline (left) and correlated (right) datasets. The upper panels of the subfigures show the training (blue) and validation (orange) losses. The lower panels show the evolution of the $\langle t_\alpha \rangle$ with all combinations of x_1 and x_2 up to the second order. Each line is given with uncertainty bands, indicating the 68 % CI determined on an ensemble size of 100 training repetitions. The developments of $\langle t_\alpha \rangle$ are matching up to epoch 120, which is marked as a dotted, grey line.

After 1 000 epochs, all $\langle t_\alpha \rangle$ tracked on the correlation dataset converge. The corresponding CIs are determined by the 16 % and 84 % quantiles of an ensemble of 100 training trials varying in the choices of random weight initializations. Their spread is narrow around their convergent points.

The values of $\langle t_\alpha \rangle$ after 1 000 training epochs are shown in the right columns of Table 7.1. It is apparent that the first-order coefficients exhibit approximately equal importance, each slightly exceeding 0.16. Relative to the baseline dataset, the impact of the second-order coefficients in the correlated case approximately doubles to $\langle t_{x_1x_1} \rangle = 8.47 \times 10^{-2}$ and $\langle t_{x_2x_2} \rangle = 8.53 \times 10^{-2}$. Additionally, $\langle t_{x_1x_2} \rangle$ increases significantly from 7.21×10^{-2} to 18.04×10^{-2}, indicating a much stronger influence on the MLP output. All 68 % CIs lie within a 5 % range of their respective medians. The most considerable asymmetries appear in both scenarios for $\langle t_{x_1} \rangle$, though their relative impact remains relatively small, suggesting a stable training result.

In the upper panel of Fig. 7.3, the decision boundary and the t_α are presented in the x_1x_2-plane after 1 000 training epochs. The decision boundary exhibits that $\Omega_{\hat{y}}$ separates S from B with a linear discriminant. The confidence of the model in the predictions is low near to the boundary and increases in confidence with larger distance from the boundary. Due to the symmetry with respect to unity, the coefficients t_{x_1} and t_{x_2} reveal the same overall structure and magnitude. Similarly, the second-order coefficients $t_{x_1x_1}$, $t_{x_1x_2}$, and $t_{x_2x_2}$ follow the same pattern, while also capturing small asymmetries of the boundary. The lower panel of Fig. 7.3 presents the state of the correlated training after 120 epochs. The overall structure of the decision boundary and the distributions of t_α appear nearly identical to those in the uncorrelated example. Noting the scales of the subplots, all t_α appear with higher values compared to the baseline dataset. The values of $\langle t_\alpha \rangle$ at epoch 120 are listed for both datasets in the left columns of Table 7.1.

The evolution of t_α over the first 120 training epochs, along with an analysis of their distributions, indicates that the MLP tend toward learning the same underlying features in the first epochs for distinguishing S and B in both setups. In both cases, the NN relies

Table 7.1.: Values of $\langle t_\alpha \rangle$ at epoch 120 and after 1 000 training epochs for the baseline and correlation dataset. For each $\langle t_\alpha \rangle$, the 68 % CI is given.

	Epoch 120 ($\times 10^{-2}$)		Epoch 1 000 ($\times 10^{-2}$)	
α	Baseline	Correlation	Baseline	Correlation
x_1	$15.19^{+0.25}_{-0.17}$	$16.04^{+0.23}_{-0.16}$	$16.18^{+0.02}_{-0.01}$	$16.06^{+0.05}_{-0.10}$
x_2	$15.11^{+0.20}_{-0.25}$	$16.11^{+0.26}_{-0.24}$	$16.03^{+0.03}_{-0.02}$	$16.11^{+0.09}_{-0.05}$
x_1x_2	$8.54^{+0.76}_{-0.54}$	$10.24^{+0.41}_{-0.80}$	$7.21^{+0.24}_{-0.15}$	$18.04^{+0.14}_{-0.15}$
x_1x_1	$5.20^{+0.32}_{-0.24}$	$5.86^{+0.22}_{-0.28}$	$3.83^{+0.08}_{-0.15}$	$8.47^{+0.12}_{-0.17}$
x_2x_2	$5.14^{+0.36}_{-0.35}$	$5.87^{+0.25}_{-0.28}$	$3.79^{+0.13}_{-0.14}$	$8.53^{+0.17}_{-0.24}$

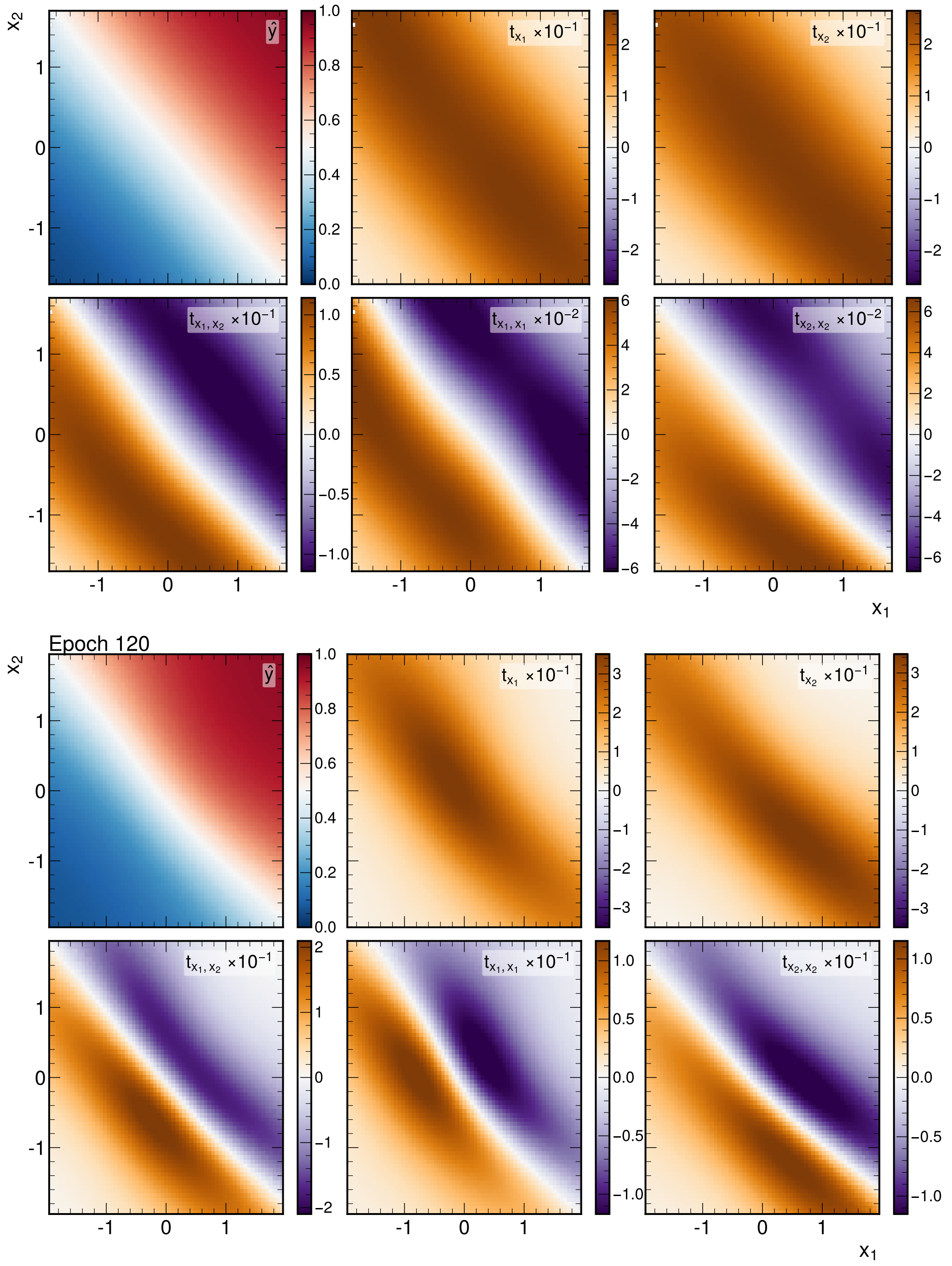


Figure 7.3.: Decision planes and t_α distributions for the baseline dataset (upper panel) after the complete training of 1 000 epochs and correlated dataset (lower) after 120 training epochs. For each subfigure in the two panels, x_2 is shown as a function of x_1. The upper left subfigure shows the MLP output $\hat{y}$ with signal-like outputs in red and background-like outputs in blue. The upper middle and right subfigures show the first order t_α, the lower subfigures show the second order coefficients. High positive values are indicated in dark bronze, and high negative values in dark purple. The subfigures are scaled as indicated in the legends.

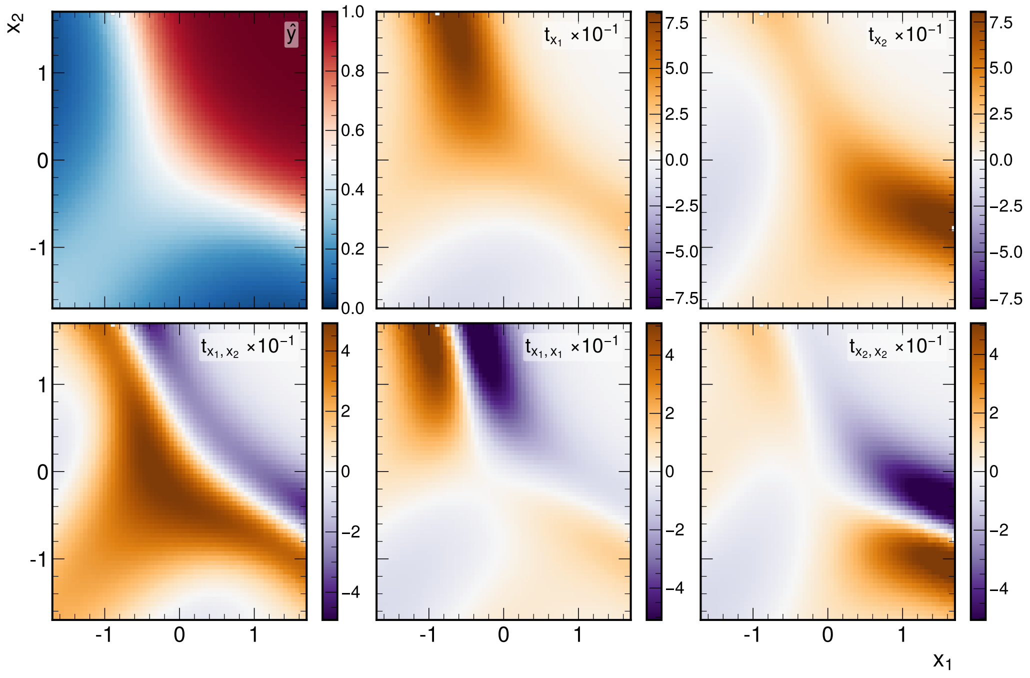


Figure 7.4.: Decision planes and t_α distributions for the correlated dataset after the complete training of 1 000 epochs. For each subfigure, x_2 is shown as a function of x_1. The upper left subfigure shows the MLP output $\hat{y}$ with signal-like outputs in red and background-like outputs in blue. The upper middle and right subfigures show the first order t_α, the lower subfigures show the second order coefficients. High positive contributions are indicated in dark bronze, and high negative values in dark purple. The subfigures are scaled as indicated in the legends.

on a linear discriminant function, and the decision-making strategy is the same for both configurations. When trained on the correlated dataset, the training starts to differ from the baseline scenario around epoch 120 and the MLP starts to identify the difference between correlation in the S and B samples.

The results after the complete training of 1 000 epochs are presented in Fig. 7.4. Compared to the state after epoch 120, the decision plane and the boundary are fully adapted to the different correlations of S and B. The first-order coefficients again concentrate on the boundary. In contrast to Fig. 7.3, the curvature in the decision boundary leads to the effect that t_{x_1} peaks in regions where $x_2 > 0$ and vice versa for t_{x_2}. A similar structure develops for $t_{x_1x_1}$ and $t_{x_2x_2}$: negative values concentrate on the S area and positive values appear on the B area.

An interesting region is where the tails of S reach into the background distribution. Here, the MLP gives B predictions, but only with values around 0.7, indicating less confidence, as in other regions classified as B. In this critical region, where the tails of S overlap with B, the decision plane forms a convex shape. Consequently, this is captured by high values of the cross curvature $t_{x_1x_2}$. This region also impacts the output distributions of the

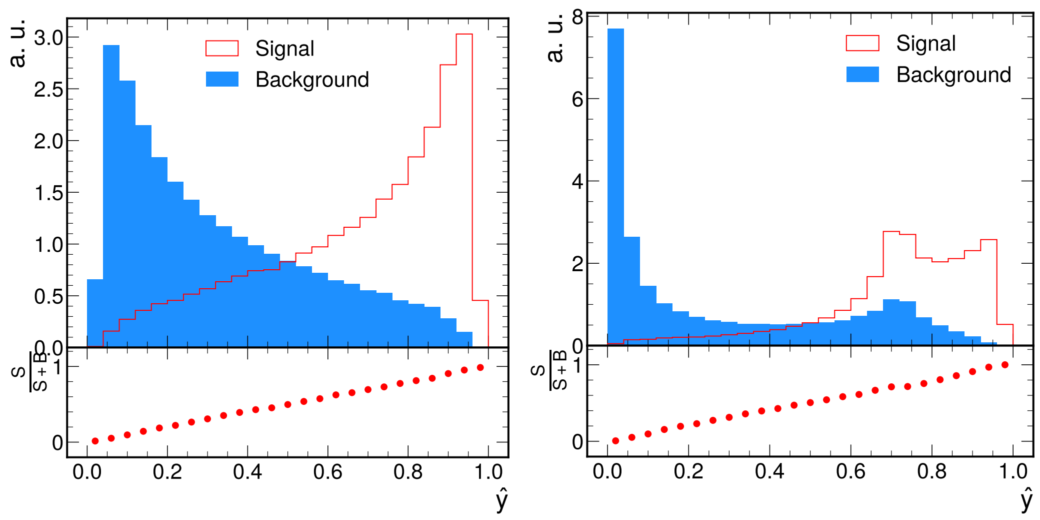


Figure 7.5.: Distribution of the output neuron $\hat{y}$ for the training on the baseline dataset (left) and the correlated one (right). The panels show the background (filled blue) and signal (lined red) distributions. The lower sections show the corresponding $\frac{\mathrm{S}}{\mathrm{S+B}}$ ratio.

model, which are depicted for both dataset scenarios in Fig. 7.5. Since the baseline dataset is symmetric regarding the decision boundary, the discrimination of S and B reveals a symmetric form, too. In contrast, a high peaking B distribution indicates that the correlated case seems more confident for B classifications than for S, which does not peak at values near one but is distorted towards 0.7. Additionally, the decision boundary exposes a bump for values around 0.7. While this isolated observation may appear suspicious and raise concerns about the reliability of the MLP, the TCA provides a reasonable explanation by highlighting the discussed convex shape in the decision plane as an area of less confident decisions.

In conclusion, this chapter demonstrated an example of a simple use case for the TCA. The learning process of an MLP, from rough structures to more detailed patterns such as correlations, is revealed by analysing $\langle t_\alpha \rangle$. In addition, the distributions of t_α were visualized to study their behavior and learn about their impacts. Finally, the curvature $t_{x_1 x_2}$ explained a peculiar irregularity in the output distribution of the MLP.

7.2. $\mathrm{t\bar{t}} + X$ dataset

This section describes the discrimination of two processes in a dataset based on pp collisions. With this example as a typical HEP task, a TCA is performed on an MLP and afterwards compared to the application on a GNN. Finally, a TS is utilized for an approximation of the MLP to inspect the relevance of the different orders in α for the MLP output.

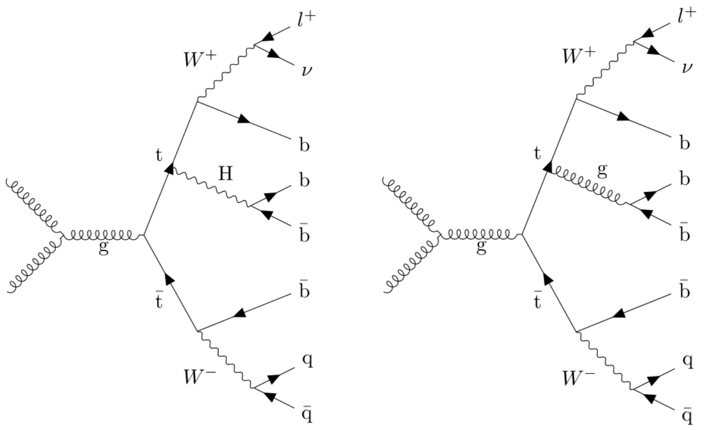


Figure 7.6.: Comparison of the $\mathrm{t\bar{t}H}$ (left) and the $\mathrm{t\bar{t}b\bar{b}}$ processes (right). Although both processes result in identical final-state particles, they differ in the underlying interactions: the $\mathrm{t\bar{t}H}$ process mediates a Higgs boson, while the $\mathrm{t\bar{t}b\bar{b}}$ process is induced by QCD interactions.

7.2.1. Dataset

The classification of events based on their final states in pp collisions at the LHC is crucial for understanding fundamental SM interactions such as the top (t) quark Yukawa coupling. For the following studies, t quark antitop-quark ($\mathrm{t\bar{t}}$) pairs have been simulated. Top quarks decay with a probability of nearly 100 % into a W boson and a bottom (b) quark ($\mathrm{t \longrightarrow Wb}$) [129]. Further, one of the two W bosons is required to decay into a lighter quark-antiquark pair (hadronically, $\mathrm{W \longrightarrow q\bar{q}}$). At the same time, the other one is assumed to decay into a charged lepton ℓ and neutrino ν (leptonically, $\mathrm{W} \longrightarrow \ell\nu$). This semileptonic $\mathrm{t\bar{t}}$ process allows for additional Higgs boson (H) radiation with H most likely decaying to a bottom quark-antiquark pair ($\mathrm{H \longrightarrow b\bar{b}}$), which is denoted as $\mathrm{t\bar{t}H}$. Such H radiation can easily be mimicked by QCD multijet processes, where a gluon (g) can be radiated and decay into $\mathrm{b\bar{b}}$. Events of this type are referred to as $\mathrm{t\bar{t}b\bar{b}}$. Figure 7.6 shows possible Feynman diagrams [130] at LO for these two processes. Both contain six jets, at least four of which are expected to originate from b quarks, one ℓ, and $p_{\mathrm{T}}^{\mathrm{miss}}$ originating from the neutrino ν, which escapes the detector. Both processes result in the same final-state configuration. Discriminating $\mathrm{t\bar{t}H}$ from $\mathrm{t\bar{t}b\bar{b}}$ is challenging since differences in the kinematic patterns of the $\mathrm{H \longrightarrow b\bar{b}}$ and the $\mathrm{g \longrightarrow b\bar{b}}$ decay have to be identified. Since it is not easy to extract these characteristics using traditional methods, this task is well-suited for the application of advanced ML algorithms.

Samples of the $\mathrm{t\bar{t}H}$ and $\mathrm{t\bar{t}b\bar{b}}$ processes have been simulated by the authors of Ref. [131] using MC methods at LO in perturbative QCD. The hard scattering process at the matrix element level [116, 132] has been simulated with `MadGraph5_aMC@NLO` version 2.9.9. The `Pythia` event generator [119] at version 8.306 has been used to match the partonic final state to the stable particle level after hadronization. Interactions of these particles with the CMS detector in the setup of the LHC Run 2 (2016-2018) are obtained from the `DELPHES` framework [121]. DELPHES uses simplified interaction models for detector components to enhance computing time efficiency compared to Geant4 [120]. This approach allows for a fast and efficient simulation of detector responses while maintaining essential features for each of the objects used for the trained NNs:

- Transverse momentum p_T perpendicular to the beam axis
- Pseudorapidity η describing the angle to the beam axis
- The azimuth angle perpendicular to the beam axis ϕ
- A discriminant variable for jet candidates originating from b quarks. This variable is discrete BTag $\in \{0, 1, 2, 3\}$ and determines under which working point α ($\alpha \geq$ BTag) a jet is identified as a b jet. The working points $\alpha = 1$ (loose), $\alpha = 2$ (medium), $\alpha = 3$ (tight) correspond to probabilities misidentifying light quark or gluon jets as b jets by roughly 10 %, 1 % and 0.1 %.

After the simulation, only events with six jets are retained, requiring at least four of them to be identified as b jets with BTag ≥ 2. Additionally, exactly one lepton $\ell \in \{e^\pm, \mu^\pm\}$ per event, originating from the semileptonic $\mathrm{t\bar{t}}$ deacy, is required. In summary, this adds up to eight reconstructed final-state objects: six jets, one ℓ, and $p_\mathrm{T}^\mathrm{miss}$. To use these objects in the following applications, the BTag value for $p_\mathrm{T}^\mathrm{miss}$ and ℓ objects is set to zero, and $p_\mathrm{T}^\mathrm{miss}$ is handled analogously to the p_T of all other objects. Except for $p_\mathrm{T}^\mathrm{miss}$, all objects have to fulfill the requirements of

$$p_\mathrm{T} > 20\,\mathrm{GeV} \quad \text{and} \quad |\eta| < 2.4. \tag{7.3}$$

After this selection, mirroring techniques in η and ϕ augment the dataset, resulting in approximately 556 000 $\mathrm{t\bar{t}H}$ and 211 000 $\mathrm{t\bar{t}b\bar{b}}$ events.

During the training of the presented models, 60 % of the data are used, while 20 % is used as a validation dataset to monitor overfitting. After the training, the model performance is evaluated on the remaining 20 % of the data. Each dataset has the same fraction of $\mathrm{t\bar{t}H}$ and $\mathrm{t\bar{t}b\bar{b}}$ events. Since the presented studies concentrate on the learning behavior of the MLPs, all presented t_α are based on the training dataset.

7.2.2. Multilayer perceptron

To distinguish $\mathrm{t\bar{t}H}$ events from $\mathrm{t\bar{t}b\bar{b}}$ events, an MLP with two hidden layers of 100 neurons each is employed. Except for the final output neuron, where a sigmoid activation function is implemented, the remaining neurons are subject to a $\tanh(\cdot)$ activation function. Overall, 35 standardized input features $\mathbf{x}$ are passed to the NN for the separation task

$$\Omega_\mathrm{MLP}(\mathbf{x}) : \mathbf{x} \in \mathbb{R}^{35} \mapsto \hat{y} \in [0, 1]. \tag{7.4}$$

For readability in sub- and superscripts, the naming convention for features in this part follows the scheme $\mathrm{object}_i^\mathrm{property}$ where i indicates the object with the i^th highest p_T, e.g.,

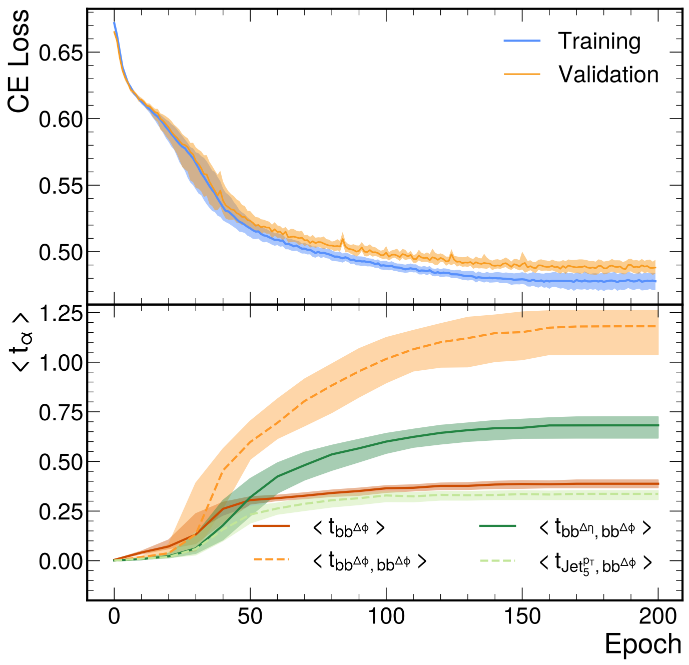


Figure 7.7.: The upper section shows the loss evolution for the training (blue) and validation (orange) datasets. The development of the t_α is shown in the lower section. Their uncertainties correspond to the 68 % CI determined based on an ensemble of 100 training repetitions.

$\mathrm{Jet}_2^{p_T}$ for neuron or vertex features and $\ell \leftrightarrow \mathrm{Jet}_1{}^{\Delta\eta}$ for the edge feature $\Delta\eta$ between lepton ℓ and Jet_1 of a GNN.

The input features used for the MLP include p_T, η, the BTag value, and the angular difference to the lepton in the event $\Delta\phi_\ell$ for each of the eight reconstruction objects. The lepton always holds $\Delta\phi_\ell = 0$. In addition, truth information is used to identify the two b jets originating from H boson or gluon g, respectively. For this $b\bar{b}$ system, the invariant mass bb^m and the angular differences $bb^{\Delta\eta}$ and $bb^{\Delta\phi}$ are used as additional inputs. Since these variables exploit kinematic information needed to distinguish the $t\bar{t}H$ from the $t\bar{t}b\bar{b}$ process, they are expected to stand out in a TCA. Furthermore, the $b\bar{b}$ features are compared with the edge features of a GNN. In a more realistic analysis, bb^m, $bb^{\Delta\eta}$, and $bb^{\Delta\phi}$ for all jet combinations would be input for the MLP. Since this leads to a significant combinatoric background in the input space for the MLP and since this study focuses on the feature impact rather than the NN performance, selecting these variables is a justified choice.

The training uses the ADAM optimizer with an initial learning rate of $\eta = 10^{-4}$. Mini-batches of 1 000 events are used to minimize the CE loss, which is extended by the L_2 regularization with the Lagrange multiplier $\lambda = 10^{-4}$. Figure 7.7 shows the development of the loss and $\langle t_\alpha \rangle$ for an ensemble of size 100 during the training. It shows that training and validation losses converge below a CE threshold of 0.5. To indicate the development of the t_α, the four most important $\langle t_\alpha \rangle$ are depicted. For the $t\bar{t}H$ identification, $\langle t_{bb^{\Delta\phi} bb^{\Delta\phi}} \rangle = 1.18$

is identified as having the highest impact on Ω_{MLP}, followed by $\langle t_{\mathrm{bb}^{\Delta\phi}\mathrm{bb}^{\Delta\eta}} \rangle$ and $\langle t_{\mathrm{bb}^{\Delta\phi}} \rangle$. The fourth-highest impact is attributed to $\langle t_{\mathrm{Jet}_5^{p_T}\mathrm{bb}^{\Delta\phi}} \rangle$. The numerical values for all traced $\langle t_\alpha \rangle$ are given in the right part of Table 7.2. This ranking supports the previously discussed expectation that the $\mathrm{b\bar{b}}$ variables play a significant role in the discrimination of $\mathrm{t\bar{t}H}$ and $\mathrm{t\bar{t}b\bar{b}}$.

It is important to note that the individual rankings in α may vary across different ensemble candidates, especially in regions where the CIs of the coefficients overlap. Since all $\langle t_\alpha \rangle$ converge to stationary points during training and show no further signs of movement, their behavior confirms the conclusion of a converged training process.

Analogous to the toy example, the upper part of Fig. 7.8 shows the feature plane of $\mathrm{bb}^{\Delta\phi}$ and $\mathrm{bb}^{\Delta\eta}$. Here, the decision plane and corresponding first- and second-order coefficients for the standardized angular features are shown. There are no events with small ΔR between the two b jets, which may be due to the anti-k_T algorithm not being able to separate overlapping jets. Events near the origin are classified as background, and the decision boundary is shaped elliptically around the region of missing events. The MLP shows reduced confidence in distinguishing $\mathrm{t\bar{t}H}$ and $\mathrm{t\bar{t}b\bar{b}}$ events in the visualized phase space compared to the toy examples.

At values $\mathrm{bb}^{\Delta\eta} = \pm 1.6$, small regions are predicted as $\mathrm{t\bar{t}b\bar{b}}$ while regions in between are predicted as $\mathrm{t\bar{t}H}$ events. This results in a positive slope from $t_{\mathrm{bb}^{\Delta\eta}} = -1.6$ toward zero, and in a negative slope when moving further from zero to positive values. The elliptical decision boundary in the center of the predictions leads to a half-moon-shaped pattern in $t_{\mathrm{bb}^{\Delta\eta}}$. Analogous considerations can be made for the remaining t_α. The values for $t_{\mathrm{bb}^{\Delta\phi}}$ and $t_{\mathrm{bb}^{\Delta\phi}\mathrm{bb}^{\Delta\phi}}$ are one order of magnitude larger compared to the remaining coefficients, indicating a large impact of $\Delta\phi$. This is also reflected in their impact on the rankings presented in Table 7.2. Slope and curvature terms generally exhibit similar behavior around the central regions, which can be attributed to the asymmetry of the decision boundary. The curvature $t_{\mathrm{bb}^{\Delta\phi}\mathrm{bb}^{\Delta\eta}}$ exhibits a symmetric, quartered structure of two positive and two negative peaking regions, the decision plane is not curved at $\mathrm{bb}^{\Delta\phi} = 0$ and $\mathrm{bb}^{\Delta\eta} = 0$.

GNN		MLP	
α	$\langle t_\alpha \rangle$ ($\times 10^{-3}$)	α	$\langle t_\alpha \rangle$ ($\times 10^{-3}$)
$\mathrm{Jet}_0 \leftrightarrow \mathrm{Jet}_1^{\Delta\phi}$, $\mathrm{Jet}_0 \leftrightarrow \mathrm{Jet}_1^{\Delta\phi}$	$62.07^{+26.94}_{-16.28}$	$\mathrm{bb}^{\Delta\phi}$, $\mathrm{bb}^{\Delta\phi}$	$1181.00^{+79.94}_{-142.05}$
$\mathrm{Jet}_0 \leftrightarrow \mathrm{Jet}_1^{\Delta\phi}$	$34.72^{+6.22}_{-6.96}$	$\mathrm{bb}^{\Delta\eta}$, $\mathrm{bb}^{\Delta\phi}$	$681.91^{+42.90}_{-64.23}$
$\mathrm{Jet}_0 \leftrightarrow \mathrm{Jet}_1^{\Delta\phi}$, $\mathrm{Jet}_0 \leftrightarrow \mathrm{Jet}_1^{\Delta\eta}$	$21.49^{+6.65}_{-3.56}$	$\mathrm{bb}^{\Delta\phi}$	$387.80^{+19.07}_{-20.18}$
$\mathrm{Jet}_0 \leftrightarrow \mathrm{Jet}_1^{\Delta\eta}$	$20.85^{+3.77}_{-3.99}$	$\mathrm{Jet}_5^{p_T}$, $\mathrm{bb}^{\Delta\phi}$	$336.63^{+16.51}_{-28.71}$

Table 7.2.: Results of the ensemble trained for the GNN (left) after 300 training epochs and the MLP (right) after the full training of 200 epochs. All uncertainties correspond to the 68 % CI.

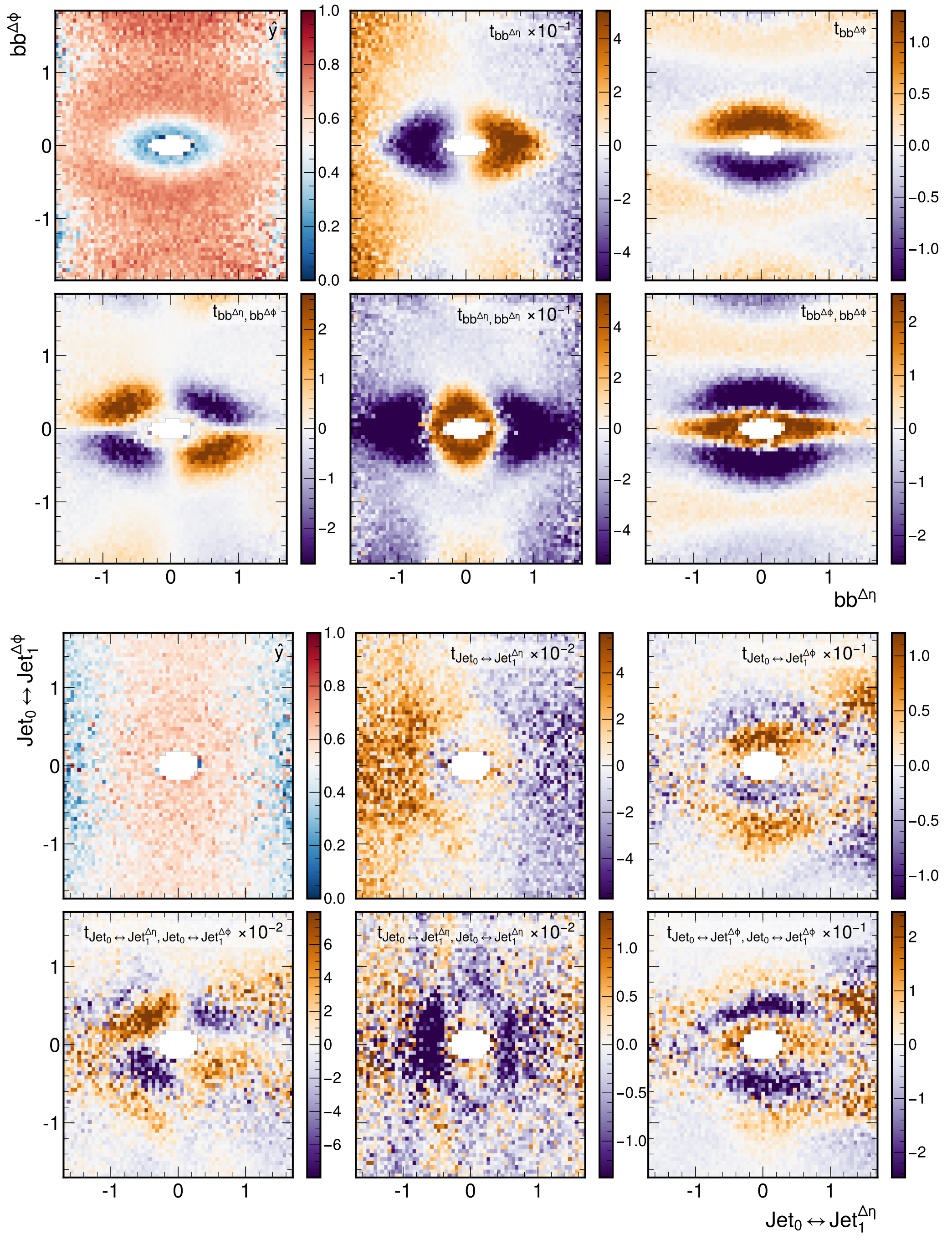


Figure 7.8.: Desicion plane and t_α comparison between the MLP (upper) and the GNN (lower). The results of the MLP are obtained after 200 training epochs and show $\mathrm{bb}^{\Delta\phi}$ as a function of $\mathrm{bb}^{\Delta\eta}$. For the GNN, the model state at epoch 300 is selected and $\mathrm{Jet}_0 \leftrightarrow \mathrm{Jet}_1^{\Delta\phi}$ is shown as function of $\mathrm{Jet}_0 \leftrightarrow \mathrm{Jet}_1^{\Delta\eta}$. In both panels, the upper left subfigure shows the MLP output $\hat{y}$ with $t\bar{t}H$-like outputs in red and $t\bar{t}b\bar{b}$-like outputs in blue. The upper middle and right subfigures show the first order t_α, while the lower ones show the second order coefficients. High positive contributions are indicated in dark bronze, and high negative ones in dark purple. The subfigures are scaled as indicated in their legends.

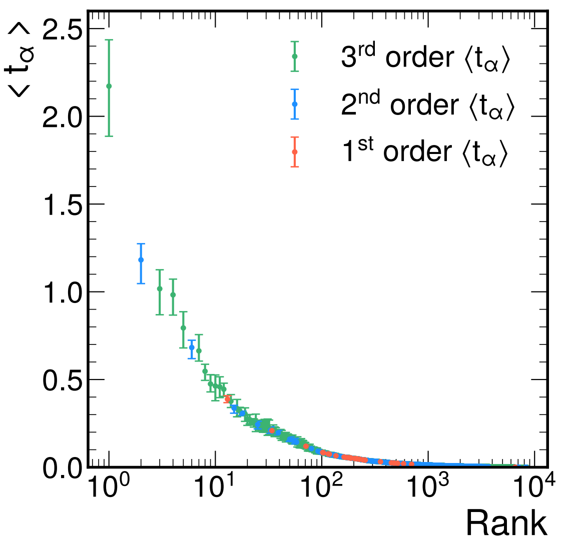


Figure 7.9.: The $\langle t_\alpha \rangle$ importance ranking for Ω_{MLP} up to the third order. For each t_α, the median value, including its 68 % CI, is determined based on an ensemble of size 100.

The symmetry of $t_{\mathrm{bb}^{\Delta\phi}\mathrm{bb}^{\Delta\eta}}$, $t_{\mathrm{bb}^{\Delta\phi}}$ and $t_{\mathrm{bb}^{\Delta\eta}}$ highlights the importance of a careful definition of $\langle t_\alpha \rangle$: If the absolute value were not taken into account, most contributions in these examples would cancel out.

All $\langle t_\alpha \rangle$ up to third order are compared in Fig. 7.9 regarding their magnitude. For an input space of 35 variables, this ranking comprises 8 434 coefficients, excluding component permutations in the higher-order terms. The most impactful parameter in the ranking is the coefficient with $\alpha = \mathrm{bb}^{\Delta\phi}\mathrm{bb}^{\Delta\phi}\mathrm{bb}^{\Delta\phi}$. Previously discussed coefficients $t_{\mathrm{bb}^{\Delta\phi}\mathrm{bb}^{\Delta\phi}}$ and $t_{\mathrm{bb}^{\Delta\eta}\mathrm{bb}^{\Delta\phi}}$ can be identified with the second and sixth largest impacts. Except for these, the highest ranks are predominantly composed of third-order coefficients.

7.2.3. Graph neural network

As a second application, the TCA is applied to a more complex NN architecture: a graph neural network (GNN). In contrast to the previously used MLP, which discriminates between $\mathrm{t\bar{t}H}$ and $\mathrm{t\bar{t}b\bar{b}}$ based on a vector $\mathbf{x}$ of 35 input variables, the GNN uses the embedded reconstructed objects in a graph structure, explained in Chapter 2. Since the selection criteria are set to a fixed number of reconstructed objects in the final state, the GNN architecture is equivalent for each event, consisting of eight vertices $\mathbf{v}_i$: six for the jets, one for the lepton ℓ, and one for $p_{\mathrm{T}}^{\mathrm{miss}}$ in the event. Each vertex carries a feature set $\mathbf{v}_i = (p_{\mathrm{T}}, \mathrm{BTag}, \mathrm{Mass})^{\mathrm{T}}$ of the corresponding object. All vertices are connected with edges $\mathbf{e}_{ij}$, which encode relative, angular information between objects i and j. These edges, defined as $\mathbf{e}_{ij} = (\Delta\eta_{ij}, \Delta\phi_{ij})^{\mathrm{T}}$, represent features that already demonstrated high t_α in the previous analysis. Angular information between two objects can inherently

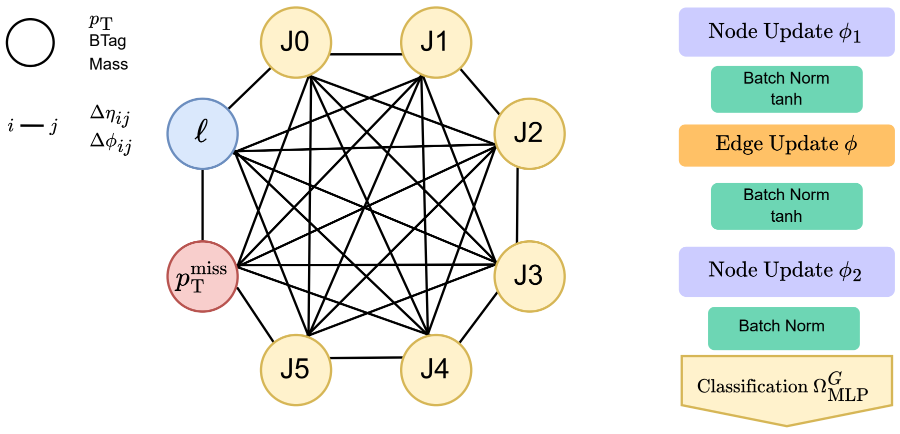


Figure 7.10.: The graph architecture of the GNN (left) consists of eight vertices $\mathbf{v}_i$ (six jets, one charged lepton ℓ, and p_T^{miss}). All vertices are connected. A forward pass for the GNN (right) consists of vertex- and edge updates.

be represented as edges of the GNN, therefore, a truth-selection of variables, as in the MLP, is not necessary. Since a pair of edges must be applied to each connection to obtain bidirectional behavior, the features of all edge pairs are averaged after each update step to keep them synchronized throughout the updates. Figure 7.10 depicts the full GNN graph architecture in its left panel.

A forward evaluation pass of the GNN consists of multiple update steps. First, all $\mathbf{v}_i$ are updated by applying Eq. (2.3), for the first update holds

$$\phi_1(\mathbf{v}_i, \mathcal{F}_{\mathrm{Agg}}(\mathbf{v}_i)) : \mathbb{R}^{3+2} \mapsto \mathbb{R}^{200}. \tag{7.5}$$

After that, edges $\mathbf{e}_{ij}$ are updated by

$$\psi(\mathbf{v}_i, \mathbf{v}_j, \mathbf{e}_{ij}) : \mathbb{R}^{200+200+2} \mapsto \mathbb{R}^{200}. \tag{7.6}$$

The second update for $\mathbf{v}_i$ is equivalent to ϕ_1, but with different input dimensions

$$\phi_2(\mathbf{v}_i, \mathcal{F}_{\mathrm{Agg}}(\mathbf{v}_i)) : \mathbb{R}^{200+200} \mapsto \mathbb{R}^{200}. \tag{7.7}$$

All update functions, ϕ_1, ψ, and ϕ_2, are implemented using MLPs composed of two layers, with the number of neurons matching the specified input and output dimensions of each function. Each update is followed by a BN layer, and the activation function used throughout is the hyperbolic tangent (tanh). For both ϕ_1 and ϕ_2, the aggregation steps $\mathcal{F}_{\mathrm{Agg}}(\cdot)$ are implemented using a mean function. After the last update, a global mean pool operation is used at the vertex level to finally apply

$$\Omega^G_{\mathrm{MLP}} : \mathbb{R}^{200} \mapsto [0, 1], \tag{7.8}$$

for a graph classification. This forward pass is illustrated in the right section of Fig. 7.10. For the implementation of this GNN, the `PyTorch` based framework `PyTorch Geometric` [133] is used.

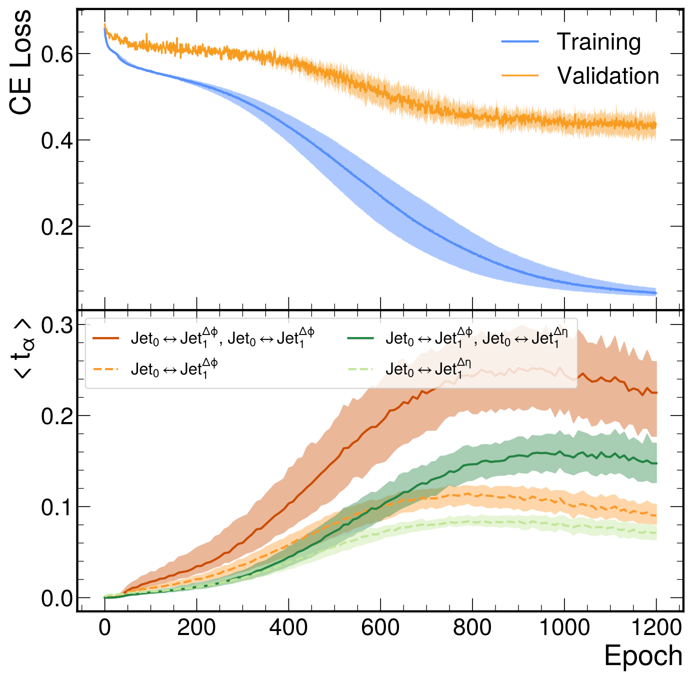


Figure 7.11.: The upper section shows the evolution of the training (blue) and validation (orange) losses for the training of the GNN. The lower section of the figure shows the development of a selected pair of $\langle t_\alpha \rangle$. All uncertainty bands correspond to the 68 % CI determined on an ensemble of size 100. The training is stopped after 300 epochs.

An important technical difference between the MLP and the GNN is that the latter model has two different input tensors, one for the edges $\mathbf{e}_{ij}$ and one for the vertices $\mathbf{v}_i$. Computing t_α for feature combinations of these tensors is currently not implemented in the `TaylorAnalysis` package. Since the angular features $\mathrm{bb}^{\Delta\phi}$ and $\mathrm{bb}^{\Delta\eta}$ showed significant impacts on the MLP, the following analyses are restricted to the edge features, holding this information.

To optimize the weights in ϕ_1, ψ, ϕ_2 and Ω^G_{MLP}, ADAM is used to minimize the CE for Ω_{GNN}. This is done based on minibatches of size 500 and an initial $\eta = 10^{-4}$ for 1 200 training epochs. In addition, an L_2 norm is added to the loss with a Lagrange multiplier $\lambda = 10^{-4}$.

Figure 7.11 shows the training process of the GNN. The loss in the upper panel indicates the onset of overfitting around epoch 300, as evidenced by a growing gap between the training and validation losses. By epoch 1 200, both losses stabilize with narrowing uncertainty bands, suggesting convergence. This observation is further supported by the behavior of the t_α in the lower panel, which shows either convergence or a slight decline at the end of the training. To mitigate effects from overfitting, the state of one GNN at epoch 300 is used for subsequent analyses of t_α, as this point precedes the beginning of pronounced overfitting. The ensemble candidate serving as the basis for the presented t_α achieves a receiver operating characteristic (ROC) area under curve (AUC) score of 0.81, indicating sufficient discriminative power. The edge $\mathbf{e}_{01}$, connecting Jet_0 and Jet_1, is analyzed for a single candidate within the ensemble. At epoch 300, $t_{\text{Jet}_0\leftrightarrow\text{Jet}_1^{\Delta\phi}}$, has the highest impact on the predictions of the GNN, followed by $t_{\text{Jet}_0\leftrightarrow\text{Jet}_1^{\Delta\phi}}$, $t_{\text{Jet}_0\leftrightarrow\text{Jet}_1^{\Delta\phi}\text{Jet}_0\leftrightarrow\text{Jet}_1^{\Delta\eta}}$, and $t_{\text{Jet}_0\leftrightarrow\text{Jet}_1^{\Delta\eta}}$. All coefficients can be compared in a ranking listed in Table 7.2 together with the results of the MLP.

Both models come to the result that $\alpha = \{\Delta\phi\}$ leads to the most important Taylor coefficient, or the most important one under the angular features, respectively. While the slope of $\Delta\phi$ ranks second for the GNN, the corresponding feature for the MLP, $\text{bb}^{\Delta\phi}$, ranks third. The same holds for the combination of $\Delta\phi$ and $\Delta\eta$, which ranks second for the MLP but third for the GNN.

Finally, the coefficients are depicted in the $\Delta\phi\Delta\eta$-plane in the lower part of Fig. 7.8. It can be observed that the distributions are noisier compared to the ones from the MLP. The decision plane has a similar shape to that of the MLP, but the signal area is narrower, and the elliptical decision boundary in the central region is not distinguishable. It shows that the GNN is less confident in its decisions. This can be traced back to the fact that $\mathbf{e}_{01}$ does not necessarily connect the true b jet daughters from the H boson but also jets from the top, gluon, or $\text{W}^{\pm}$ decays. In contrast, the $\text{b}\bar{\text{b}}$ variables utilized in the MLP training contained truth information.

The half-moon shapes in the slope of $\Delta\eta$ and $\Delta\phi$ are still recognizable but less pronounced than the remaining areas in the background, showing the same overall behavior. The same observation holds for $t_{\text{Jet}_0\leftrightarrow\text{Jet}_1^{\Delta\phi}\text{Jet}_0\leftrightarrow\text{Jet}_1^{\Delta\phi}}$, but for $t_{\text{Jet}_0\leftrightarrow\text{Jet}_1^{\Delta\eta}\text{Jet}_0\leftrightarrow\text{Jet}_1^{\Delta\eta}}$, the noise is to high to identify meaningful similarities. The subfigure for $t_{\text{Jet}_0\leftrightarrow\text{Jet}_1^{\Delta\eta}\text{Jet}_0\leftrightarrow\text{Jet}_1^{\Delta\phi}}$ is also noisy, but the quartered structure of negative and positive contributions is clearly visible.

Even though the different model architectures make direct comparisons challenging, this study analyzed the model decision planes in a reduced two-dimensional parameter space based on the corresponding t_α. The results showed an outstanding impact by the curvature of $\alpha = \{\Delta\phi\Delta\phi\}$ parameters during and after the training. Therefore, this study not only demonstrated the TCA method on an advanced GNN architecture, but it also proved its ability to unveil a consistent learning behavior between an MLP and a GNN architecture on a realistic $\text{t}\bar{\text{t}} + X$ dataset.

7.3. Relevant order analysis

Due to the powers of the distance factors $(x_i - a_i)$ in Eq. (6.2), higher orders are scaled with smaller factors. Hence, the resulting magnitude of $\langle t_\alpha \rangle$ is not directly comparable between different orders. This raises the question, how many orders in α should be considered for a TCA. To answer this question, the following study investigates the relevance of

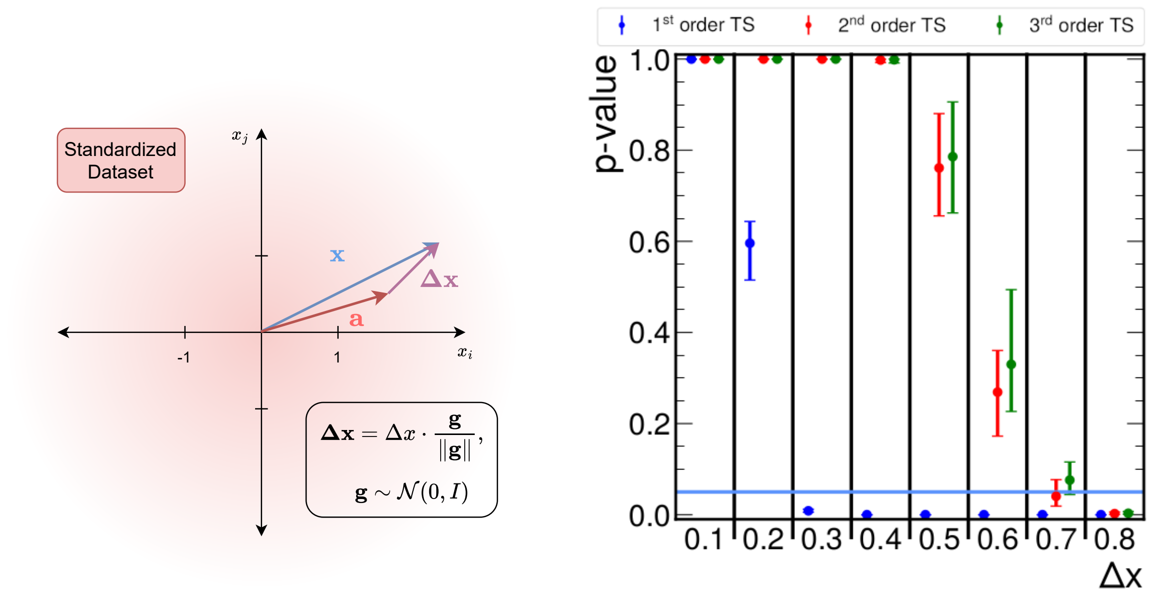


Figure 7.12.: On the left, the illustration for the shifting procedure is shown. The red distribution indicates the exemplary, standardized dataset A. The random direction of a shift is sampled and scaled with a fixed length Δx, resulting in $\Delta\mathbf{x}$ (purple arrow). Each point $\mathbf{a}$ (red arrow) in A is shifted by $\Delta\mathbf{x}$, which is exemplarily shown for $\mathbf{x} = \mathbf{a} + \Delta\mathbf{x}$ (blue arrow). On the right, the p-value for the Kolmogorov–Smirnov (KS) test is given for different shift lengths Δx. For each of those, an ensemble of 50 TS is calculated up to the first (blue), second (red), and third (green) order to determine the median and 68 % CI for each combination. The significance level $\alpha = 0.05$ is indicated by a horizontal blue line.

different orders in the TS, based on the MLP presented for t$\bar{\text{t}}$H and t$\bar{\text{t}}$b$\bar{\text{b}}$ discrimination in Section 7.2.2.

To this end, $\Omega_{\text{MLP}}(\mathbf{x}) : \mathbf{x} \mapsto \hat{y}$ is evaluated for 50 000 events of the training dataset. For the expansion of the TS, these events serve as development points of the TS and are denoted as $\mathbf{a} \in A \subset X^{\text{train}} \subseteq \mathbb{R}^{35}$. The TS is evaluated for the first order only, up to the second order, and finally up to the third order to approximate the MLP output. For the chosen input dimension of 35 input variables, this results in 35 first-order terms, 1 225 second-order terms, and 42 875 third-order terms.

Since the TS is developed around $\mathbf{a}$ and its accuracy decreases with growing distance, this test is performed for different deviations from $\mathbf{a}$, denoted as $\Delta\mathbf{x}$. To generate $\Delta\mathbf{x}$, a direction is randomly sampled in each dimension and scaled with a fixed length Δx, following

$$\Delta\mathbf{x} = \Delta x \cdot \frac{\mathbf{g}}{\|\mathbf{g}\|} \quad \text{with } \mathbf{g} \sim \mathcal{N}(0, \mathcal{I}_{35}). \tag{7.9}$$

Each point $\mathbf{a} \in A$ is shifted by $\Delta\mathbf{x}$, resulting in $\mathbf{x} = \mathbf{a} + \Delta\mathbf{x}$. The set of these points forms the evaluation dataset X. Both the TS and the MLP are evaluated on X to assess the quality

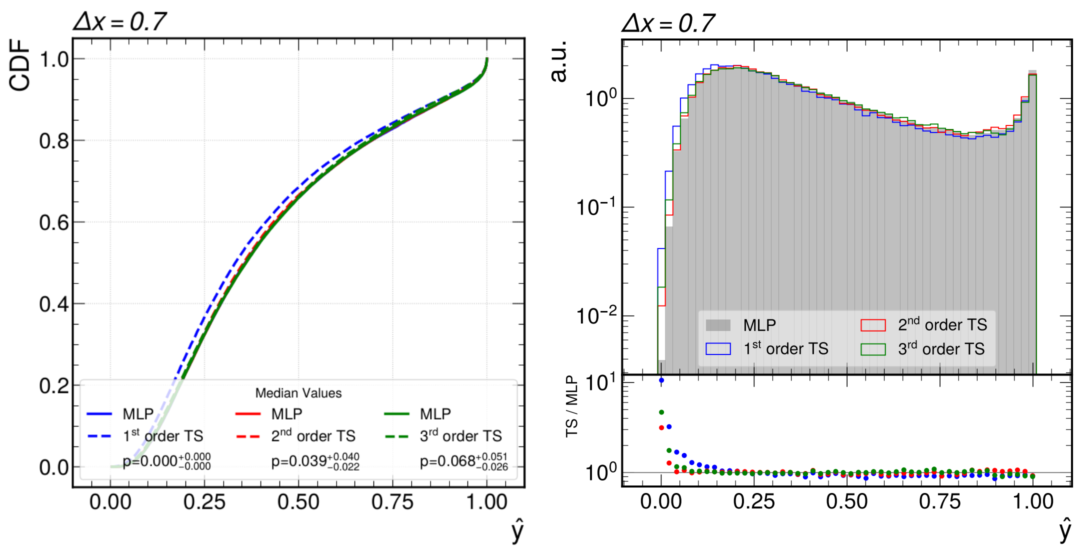


Figure 7.13.: The left figure shows the comparison of the cumulative density function (CDF), computed for the KS test, between the median values of the MLPs and TS of the ensemble. Additionally, the median and the 68 % CI for the determined p-values of the KS tests are given. The right section presents the corresponding histogrammed output values. Both figures account for $\Delta x = 0.7$

of the TS approximating the MLP. An illustration for this procedure is given in Fig. 7.12, the red background distribution represents the standardized dataset A, and the vectors indicate an exemplary shift applied to one event.

For the first order, the distribution of $\Omega_{\text{MLP}}(X)$ is compared with

$$\Omega_{\text{MLP}}(\mathbf{x}) \approx \Omega_{\text{MLP}}(\mathbf{a}) + \sum_i \frac{\partial \Omega_{\text{MLP}}(\mathbf{a})}{\partial x_i}(x_i - a_i), \tag{7.10}$$

for each $\mathbf{a} \in A$ and $\mathbf{x} \in X$. Due to the sigmoid activation function in the final MLP layer, its output is constrained to the range $\Omega_{\text{MLP}}(X) : X \mapsto [0, 1]$. In contrast, the TS is not constrained and can yield values outside the $[0, 1]$ interval. Accounting for this inequality, values smaller than zero or larger than one are not taken into account for the following comparison. To assess the similarity of these distributions, a Kolmogorov–Smirnov (KS) test is conducted, determining the associated p-value. A significance level of $\alpha_{\text{KS}} = 0.05$ is chosen. If the p-value is less than α_{KS}, the null hypothesis, stating that both samples follow the same distribution, is rejected. This procedure is repeated analogously for the TS up to the second and up to the third orders. Since $\Delta\mathbf{x}$ is a constant shift for the whole dataset A, it may align with a direction the MLP is (in)sensitive to. Thus, each test is repeated for an ensemble of 50 sampled directions $\Delta\mathbf{x}$ to mitigate this possibility.

The result of this procedure is shown as a scan $\Delta x = \{0.1, 0.2, \ldots, 0.8\}$ in the right section of Fig. 7.12. Since the dataset is assumed to be standardized with a standard deviation of one, the mean length of the vectors is one.

For the first shift $\Delta x = 0.1$, no significant effects can be observed. In the second step, $\Delta x = 0.2$, the p-value for the first order TS decreases to $0.595^{+0.063}_{-0.088}$ and drops to $0.009^{+0.003}_{-0.004}$ for $\Delta x = 0.3$ while the second and third order TS remain unchanged with p values of $1.000^{+0.000}_{-0.000}$. The first order TS falls below the significance level $\alpha_{\mathrm{KS}} = 0.05$ when shifting A with $\Delta x = 0.3$. For $\Delta x = 0.5$, the second order TS shows a p-value of $0.766^{+0.092}_{-0.122}$ while the one for the third order TS remains slightly higher at $0.790^{+0.151}_{-0.194}$.

At Δx =0.7, the second order TS fails the KS test, reaching a p-value of $0.039^{+0.040}_{-0.022}$ although the upper bound of the CI remains above the significance threshold. The results for the KS test at $\Delta x = 0.7$ and the corresponding cumulative density functions (CDFs) are shown as median values of their ensemble in the left section of Fig. 7.13. On the right section of Fig. 7.13, the distributions of the MLP and TS are presented for a random candidate of the ensemble. While all TS perform well in approximating the MLP in the central region and for values near one, the ratio plot shows that all TS have shortcomings in the steep region near small class scores.

This behavior arises from the fact that the MLP shows higher confidence in its predictions for the $\mathrm{t\bar{t}H}$ class compared to $\mathrm{t\bar{t}b\bar{b}}$. The sharp cutoff at class scores close to one is partially supported by the applied cutting of the TS values that exceed one. In contrast, the transition near zero is not step-like but remains sufficiently steep, posing a challenge for the TS to accurately approximate this region.

This section compared the output of an MLP, trained for the discrimination of $\mathrm{t\bar{t}H}$ and $\mathrm{t\bar{t}b\bar{b}}$ events, with its TS up to the third order. The comparison was made for multiple distances Δx from the development point $\mathbf{a}$. While the first-order TS failed a KS test at $\Delta x = 0.3$, the second order passed the KS test up to $\Delta x = 0.6$, and the third up to $\Delta x = 0.7$. Considering the computational cost required to calculate 42 875 third-order terms, the conclusion made from this analysis is that it is sufficient to use a TS up to the second order to describe the MLP under the described test circumstances.

8. Directions for future analyses

Research on the TCA remains an open and actively evolving field. While the present study demonstrates fundamental applications and analyses of the TCA, numerous directions for future exploration remain. In this work, the relevance of different orders in the context of a $\mathrm{t\bar{t}} + X$ dataset was investigated. An obvious continuation would be to study how t_α can be systematically compared across different orders.

Furthermore, this study compared the distributions of t_α between the $\mathrm{t\bar{t}H}$ and $\mathrm{t\bar{t}b\bar{b}}$ processes. Extending such comparisons could reveal class-dependent effects and provide deeper insight into the importance of class-specific features and their interplay.

A logical next step would be to benchmark the TCA against other interpretability methods, such as SHAP and LOO, using a common dataset. Such a comparison could investigate differences in interpretability, computational efficiency, and sensitivity to feature dependencies and redundancies.

Importantly, the utility of the TCA extends beyond a posteriori interpretability. The insights obtained through t_α can be fed back into the training process. For instance, a TCA-based early stopping criterion could monitor discrepancies between $t_\alpha(X^{\text{train}})$ and $t_\alpha(X^{\text{val}})$, potentially detecting overfitting and also attributing it to specific components.

Moreover, since t_α quantifies the influence of individual features and their combinations on the model output, this can be employed to take control of the impact of t_α. This opens the door to integrating t_α into the training objective, for example, to explicitly penalize the reliance on certain variables or combinations of the features. Such a regularization strategy could mitigate unwanted bias, enforce decorrelation from sensitive attributes, and prevent the model from exploiting unintended feature interactions, leading to more robust and interpretable NNs.

The results presented here emphasize the potential of the TCA as a powerful tool for both understanding and improving NN models.

Part IV.

B jet momentum corrections

9. Analysis setup and machine learning models

9.1. B jet momentum regression

Accurate momentum determination of jets is essential in HEP experiments to reconstruct their parent particle properties precisely. This is demonstrated, for instance, in the inference of the invariant mass m_{bb} of $\mathrm{H} \rightarrow \mathrm{b\bar{b}}$ decays. However, detector effects, such as nonlinear detector responses, inherently limit the accuracy and precision of corresponding estimates. Since jets are reconstructed objects and subject to these effects, their momentum measurements must be corrected using dedicated JMC, as outlined in Section 5.5.2.

In contrast to light-flavor jets originating from u, d, s quarks or gluons, b jets can undergo semileptonic weak decays involving neutrinos ν. Neutrinos leave the detector without a measurement, introducing an additional source of missing momentum. This degrades the resolution of the b jet momentum itself and impacts the reconstruction of invariant masses m_{bb}. Consequently, the inclusive JMCs applied to all jets are insufficient for b jets, which require a specialized momentum regression applied in addition to the inclusive corrections.

A well-established implementation of such a regression is provided by Ref. [134], where an Ω_{MLP} model is trained on 100 million simulated b jets based on the CMS detector configuration of 2016. This model achieves a relative improvement in momentum resolution of 12–15% beyond the inclusive JMCs. Furthermore, it provides per-jet momentum resolution estimates, necessary for the determination of the resolution of composite observables such as m_{bb}. This MLP-based regression is well established within the CMS reconstruction and analysis software (CMSSW) and is therefore standard in current CMS analyses.

This chapter proposes a novel approach based on an NF model, which enables modeling the full PDF of the corrected jet p_{T}. As a result, it allows a robust estimate of the corrected b jet momentum, including an assessment of its uncertainty.

Simulated data corresponding to the CMS detector configuration of 2018 is utilized to evaluate and compare both methods. For this, the MLP model in Ref. [134] has been reimplemented to evaluate the performance of its JMC and per-jet resolution. The JMCs obtained from the NF-based approach are then directly compared to Ω_{MLP}, highlighting direct performance differences between both methods.

9.2. Dataset

This analysis is based on b jets originating from simulated pp collisions at the LHC, operating at a center-of-mass energy of 13 TeV. Collisions are simulated based on the configuration of the CMS detector in 2018.

Since t quarks almost exclusively decay into a Wb pair, this process serves as source of b jets. Events are simulated either with both W bosons decaying hadronically ($\mathrm{W} \longrightarrow \bar{q}q$), or with one W decaying hadronically and the other leptonically ($\mathrm{W} \longrightarrow \ell\nu_\ell$). The simulation of the hard scattering process is performed using `POWHEG v2` [117], which provides NLO accuracy in perturbative QCD. Parton showerings are simulated using `Pythia8.2` [119], and the response of the CMS detector is modeled with the software package `Geant4` [120].

A distinction is made between b jets at generator-level (gen) and reconstruction-level (reco). The anti-k_T algorithm clusters both generator-level and reconstruction-level jets. While neutrinos are included in the clustering at the generator level, they are not considered at the reconstruction level, as they do not interact with the CMS detector. Only b jets satisfying

$$p_\mathrm{T}^\mathrm{gen} > 15\,\mathrm{GeV}, \quad p_\mathrm{T}^\mathrm{reco} > 15\,\mathrm{GeV}, \quad \text{and} \quad |\eta| < 2.4, \tag{9.1}$$

are retained for further analysis. For the $\mathrm{t\bar{t}}$ dataset, this results in 78.78 million b jets available for training and 26.26 million for validation and testing each. Here, $p_\mathrm{T}^\mathrm{reco}$ refers to the momentum with inclusive JMC, but without b jet specific corrections. For the b jet momentum correction, various jet-related features are provided following Ref. [134]. Kinematic input variables include $p_\mathrm{T}^\mathrm{reco}$, pseudorapidity η, mass m, and transverse mass

$$m_\mathrm{T} = \sqrt{E^2 + p_\mathrm{T}^2}, \tag{9.2}$$

computed using only the transverse momentum components of the b jet. The median energy ρ for a fixed spatial angle of the event is also included to account for pileup effects.

Regarding lepton information within the jet, its angular separation from the jet axis ΔR^ℓ is used, along with an encoding of the lepton flavor in $\ell_{\mathrm{is}\,e}$, $\ell_{\mathrm{is}\,\mu}$, or ℓ_isOther if none is present. Concerning the SV of the jet, several features are used: the number of tracks associated with a SV, its transverse momentum $p_\mathrm{T}^\mathrm{vtx}$ and mass m_vtx, the three-dimensional displacement from the PV $\mathrm{d}_\mathrm{vtx,3D}$, and its associated uncertainty [135, 136].

To describe the composition of the jet, the highest p_T of the charged hadron candidates within the jet is utilized. Additionally, energy fractions $f^{\Delta R}_\mathrm{component}$ of jet constituents are computed within various radial intervals ΔR around the jet axis

$$\Delta R \in \{[0.00-0.05],\ [0.05-0.10],\ [0.10-0.20],\ [0.20-0.30],\ [0.30-0.40]\}. \tag{9.3}$$

This is done for the charged hadronic, neutral hadronic, electromagnetic, and muon components of the jet.

The total number of PF candidates associated with the jet is also considered. The variable

$$p_{\mathrm{T}}^{\mathrm{D}} = \sqrt{\frac{\sum_i p_{T,i}^2}{\sum_i p_{T,i}}}, \tag{9.4}$$

is used to characterize how the transverse momentum is distributed among the jet constituents, providing a measure of whether the jet p_{T} is concentrated or uniformly distributed.

By design, the NF model requires the azimuthal angle ϕ as an additional input feature, as will be discussed in Section 9.4. In summary, Ω_{MLP} receives a total of 43, while Ω_{RNVP} is provided with 44 inputs.

9.3. Multilayer perceptron

As a benchmark, the studies presented in Ref. [134], corresponding to the implementation within the CMSSW framework, have been repeated. In Ref. [134], the b jet momentum regression uses an MLP comprising six hidden layers. The first three layers consist of 1,024 neurons each, followed by layers containing 512, 256, and 128 neurons, respectively. This design follows a progressive reduction in dimensionality, aiming to compress and refine the information representation throughout the network.

The Leaky ReLU activation function, defined in Eq. (1.8), is employed with a negative slope parameter of $\beta = -0.2$. A total of 43 input features are used, forming the input vector $\mathbf{x} \in \mathbb{R}^{43}$, which is mapped to a three-dimensional output vector $\hat{\mathbf{y}} \in \mathbb{R}^3$ by the network function $\Omega_{\mathrm{MLP}}(\mathbf{x}) : \mathbf{x} \mapsto \hat{\mathbf{y}}$. These three outputs correspond to the central regression target for the correction factor for $p_{\mathrm{T}}^{\mathrm{reco}}$ and estimate q_{25} and q_{75}, corresponding to the 25 % and 75 % quantiles of its associated PDF.

Dropout with a probability of 10 % is applied to each hidden layer to prevent overfitting. In addition, BN is employed after each layer.

The loss function L_{MLP} is a composition of three terms: the first one is the Huber loss [137]

$$L_\delta(z) = \begin{cases} \frac{1}{2}z^2 & \text{for } |y - \hat{y}| \leq \delta, \\ \delta|z| - \frac{1}{2}\delta^2 & \text{otherwise.} \end{cases} \tag{9.5}$$

For residuals z below a threshold δ, the Huber loss transitions from a quadratic to a linear form, leading to more robustness against outliers while maintaining sensitivity and precision in regions with small z. The two remaining terms in L_{MLP} are quantile losses [138]

$$L_\tau(z) = \begin{cases} \tau z & \text{for } z > 0, \\ (1-\tau)z & \text{otherwise.} \end{cases} \tag{9.6}$$

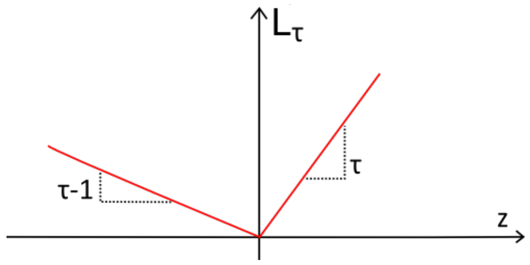


Figure 9.1.: Illustration for the quantile loss function (solid red line). The slope of the linear parts penalizes underestimates $(\tau - 1)z$ differently from overestimates τz. Adapted from Ref. [139].

where τ determines the quantile in question. The parameter τ controls the asymmetry in penalizing the overestimates and underestimates of the target variable z. If the loss L_τ cannot be minimized to zero, the model instead converges to the τ-quantile estimate of z. An illustration of L_τ is given in Fig. 9.1.

All three loss terms are added and equally weighted, with $\delta = 1$, and choosing the q_{25} and q_{75} quantiles, which leads to

$$L_{\mathrm{MLP}}(y, \hat{\mathbf{y}}) = L_{\delta=1}\left(y - \frac{p_{\mathrm{T}}^{\mathrm{gen}}}{p_{\mathrm{T}}^{\mathrm{reco}}}\right) + L_{\tau=25}(y - \hat{y}_{25}) + L_{\tau=75}(y - \hat{y}_{75}), \tag{9.7}$$

$$\text{with } \hat{\mathbf{y}} = \begin{pmatrix} \frac{p_{\mathrm{T}}^{\mathrm{gen}}}{p_{\mathrm{T}}^{\mathrm{reco}}} \\ \hat{y}_{25} \\ \hat{y}_{75} \end{pmatrix}. \tag{9.8}$$

Instead of standardizing, the training targets are set to the ratio $p_{\mathrm{T}}^{\mathrm{gen}}/p_{\mathrm{T}}^{\mathrm{reco}}$, which also corresponds to values close to one.

The training is conducted using the ADAM optimizer with an initial learning rate of 10^{-4} to optimize a total of 2.8 million parameters. Early stopping is implemented to stop the training if the validation loss does not improve over 15 consecutive epochs.

9.4. Normalizing flow

A conditional NF is particularly suited for modeling complex, analytically intractable target distributions p_Y, making it a natural choice for generative tasks. In the context of detector measurements, a limited resolution of such point estimates is present due to smearing effects of the detector components, finite granularity, and limited training data, implying that perfect predictions of the transverse momentum p_{T} are expected to be fundamentally unattainable. Instead, NFs can infer probabilistic estimates of p_{T} based on a PDF, inherently capturing uncertainties of the underlying measurement process.

This assumption is supported by the UAT, which suggests that, in the idealized scenario of a sufficient amount of noise-free data, a probabilistic model could learn a sharply peaking function centered at $p_\mathrm{T}^\mathrm{gen}$ by increasing the number of trainable parameters. Although the model cannot represent an exact delta distribution due to its restriction to continuous functions, it can approximate a peaking function around $p_\mathrm{T}^\mathrm{gen}$, which provides sufficient precision for the intended application.

Therefore, the objective of the NF model utilized for the b jet regression is to approximate the PDF of the p_T for given jet features $\mathbf{x}$. This is accomplished by learning a bijective transformation from a Gaussian latent space $\mathbf{z} \in \mathbb{R}^2$ to a positive target distribution defined by the squared transverse momentum components

$$\mathbf{y} = (p_x^2, p_y^2)^\mathrm{T} \in \mathbb{R}_+^2. \tag{9.9}$$

Based on the results of the studies discussed in Section 3.4, an RNVP architecture is chosen to estimate $\hat{\mathbf{y}}$

$$\Omega_\mathrm{RNVP}^{-1} : (\mathbf{z}, \mathbf{x}) \mapsto \hat{\mathbf{y}}, \tag{9.10}$$

where $\mathbf{x}$ is used as a conditional input for Ω_RNVP. This results in a parametrization of Ω_RNVP, allowing the model to generate a per-jet PDF tailored to $\mathbf{x}$. The left section in Fig. 9.2 illustrates this parametrization.

Since all input variables are independent from the azimuthal angle ϕ, the Ω_RNVP model cannot sufficiently resolve the momentum components in x and y. Therefore, it is necessary to add the azimuthal angle of the b jet ϕ to the set of input features, resulting in $\mathbf{x} \in \mathbb{R}^{44}$ for Ω_RNVP. This formulation results in an indirect but still accurate reconstruction of the transverse momentum

$$p_\mathrm{T} = \sqrt{p_x^2 + p_y^2}. \tag{9.11}$$

The right part of Fig. 9.2 illustrates the predicted PDF for a representative jet in the test dataset. The distribution is obtained by sampling latent variables from a Gaussian $\mathbf{z} \sim \mathcal{N}$ and applying $\Omega_\mathrm{RNVP}^{-1}$ to obtain $\mathbf{y}$, followed by the computation of p_T. While the PDF in Fig. 9.2 is based on 100 000 samples, other PDF estimates throughout this chapter are obtained from sample sizes of 200. From the resulting distribution, the median is used as the central prediction, with q_{25} and q_{75} defining the resolution interval. The median is preferred over the mean or mode as an estimator due to its robustness to outliers and statistical noise, particularly relevant in sparsely sampled regions of the PDF.

By representing p_T in its squared Cartesian components, the model is guided to learn physically meaningful structures in the data, while the target space is still constrained to the first quadrant.

The implemented Ω_RNVP architecture consists of 14 affine coupling flows $\mathcal{T}_\mathrm{RNVP}$, alternating between updates to p_x^2 and p_y^2, transforming each feature seven times. The scaling $\mathbf{s}$ and translation $\mathbf{t}$ parameters in each coupling flow are computed using an MLP with one hidden

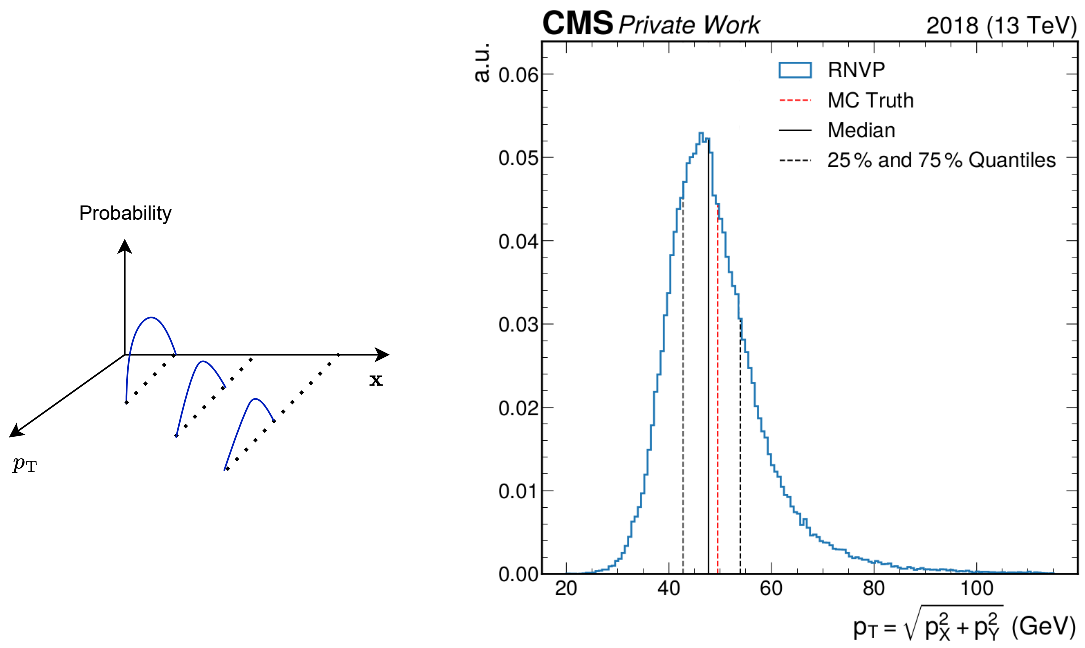


Figure 9.2.: The left section illustrates the parametrization for the PDF of the jet p_T (blue distributions) w.r.t. a vector $\mathbf{x}$. The right figure shows the PDF predicted by Ω_{RNVP} for an exemplary jet. The 25 % and 75 % quantiles of the PDF are shown as dotted black lines, and the MC truth is shown as a dotted red line.

layer of 150 units. For the implementation of Ω_{RNVP}, the `PyTorch` based `FrEIA` library [140] is used. In total, the model includes approximately 848,400 trainable parameters, significantly fewer than the 2.8 million parameters used in the benchmark MLP architecture described in Ref. [134].

The model training is conducted by minimizing the KL divergence between the learned and target distributions using the ADAM optimizer, initialized with a learning rate of 10^{-3}. A learning rate scheduler reduces η by a factor of 0.7 whenever the training loss stagnates for four consecutive epochs. The training is stopped based on the early stopping criterion, which is triggered if no improvement in the validation loss is observed for more than 15 epochs.

10. Analysis results

10.1. Model comparison

Following the training, the b jet specific JMCs are determined by both models for b jets in the test dataset and applied on top of the inclusive JMCs. The fully corrected $\hat{p}_{\mathrm{T}}^{\mathrm{reco}}$, or the correction factors $p_{\mathrm{T}}^{\mathrm{gen}}/\hat{p}_{\mathrm{T}}^{\mathrm{reco}}$, of the b jets are compared between both models throughout the following sections. As an initial evaluation metric, the relative residual

$$\frac{\hat{p}_{\mathrm{T}}^{\mathrm{reco}} - p_{\mathrm{T}}^{\mathrm{gen}}}{p_{\mathrm{T}}^{\mathrm{gen}}}, \tag{10.1}$$

is computed. The corresponding distributions for Ω_{RNVP} and Ω_{MLP} are shown in the upper panel of Fig. 10.1.

The results reveal that predictions from the Ω_{RNVP} model are nearly symmetrically distributed around zero, with a median residual of 0.001, indicating an almost unbiased estimate of $\hat{p}_{\mathrm{T}}^{\mathrm{reco}}$. In contrast, the Ω_{MLP} model shows a slight systematic overestimate, with a median residual of 0.028, corresponding to a consistent bias in $\hat{p}_{\mathrm{T}}^{\mathrm{reco}}$. The resolution, quantified by the interquartile half-width

$$\sigma = \frac{q_{75} - q_{25}}{2}, \tag{10.2}$$

of the 25 % and 75 % quantiles, is 0.199 for Ω_{RNVP} and 0.202 for Ω_{MLP}, suggesting that both models achieve similar levels of precision.

To further investigate the bias of both models, the lower panel of Fig. 10.1 shows the residuals as a function of $p_{\mathrm{T}}^{\mathrm{gen}}$. Both models tend to overestimate $\hat{p}_{\mathrm{T}}^{\mathrm{reco}}$ at low $p_{\mathrm{T}}^{\mathrm{gen}}$, with the Ω_{MLP} exhibiting a more pronounced bias than the Ω_{RNVP}. As $p_{\mathrm{T}}^{\mathrm{gen}}$ increases, the bias progressively decreases. The Ω_{RNVP} model achieves nearly unbiased predictions in the range $p_{\mathrm{T}}^{\mathrm{gen}} \in [59, 69)$ while the Ω_{MLP} achieves its best performance at higher momenta at $[92, 110)$. At higher $p_{\mathrm{T}}^{\mathrm{gen}}$, both models begin to systematically underestimate the transverse momentum, but the Ω_{MLP} slightly less.

Despite the opposite-sign biases at low and high $p_{\mathrm{T}}^{\mathrm{gen}}$, these effects largely cancel out in the aggregated residual distributions or the Ω_{RNVP}. For Ω_{MLP}, the effects of rising underestimations predominate and do not cancel out, resulting in the systematic overall bias.

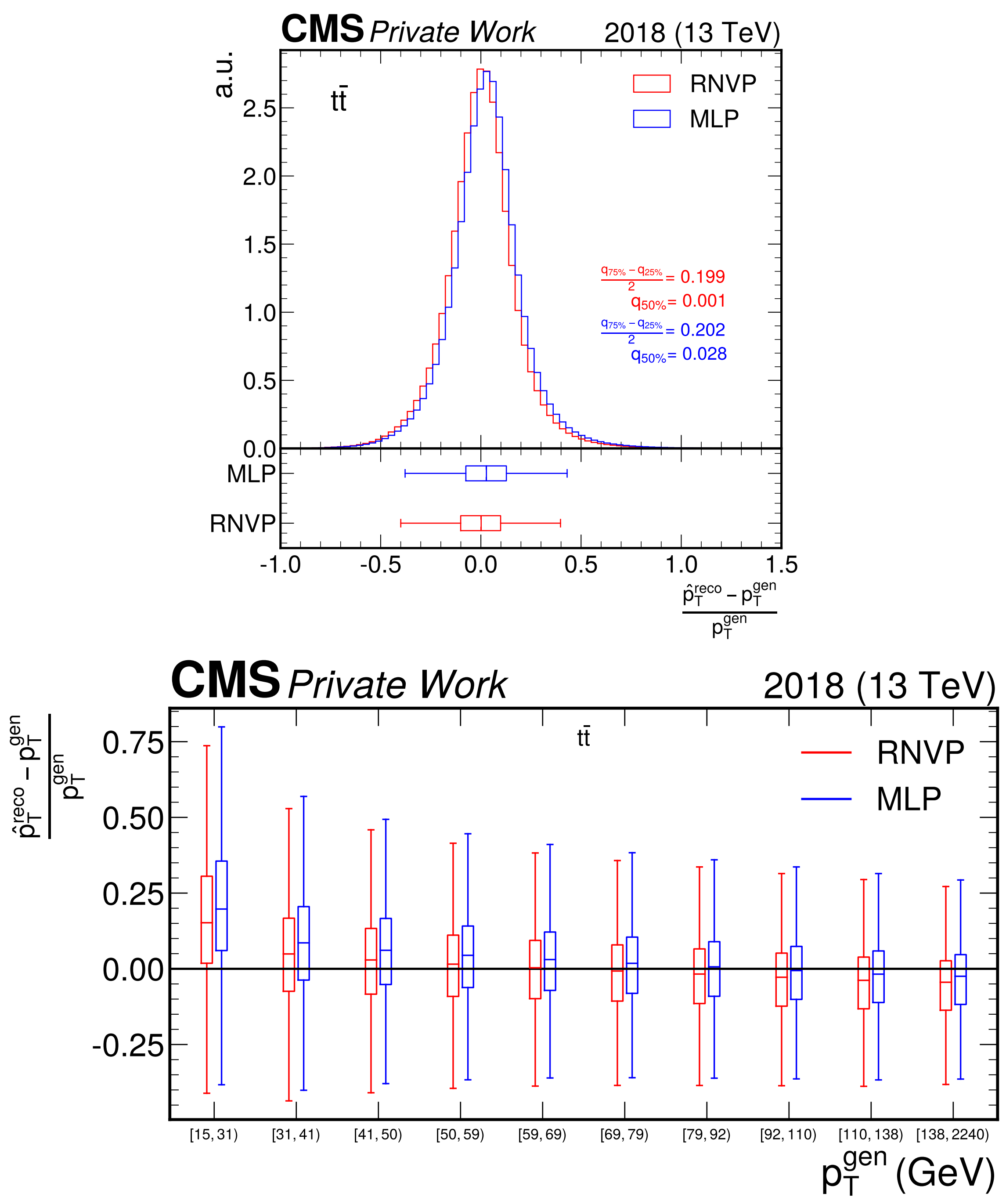


Figure 10.1.: The upper part shows the comparison of the residual differences of the p_T prediction for Ω_{RNVP} (red) and Ω_{MLP} (blue). In addition, the resolution and the median are noted for each model. The lower panel summarizes the distributions for both models as boxplots. In the lower part, the comparison of p_T^{gen} for Ω_{RNVP} (red) and Ω_{MLP} (blue) in terms of residual differences is shown as a function of p_T^{gen}. For this, p_T^{gen} is binned in equally populated intervals, for each of them, and both models are shown as boxplots.

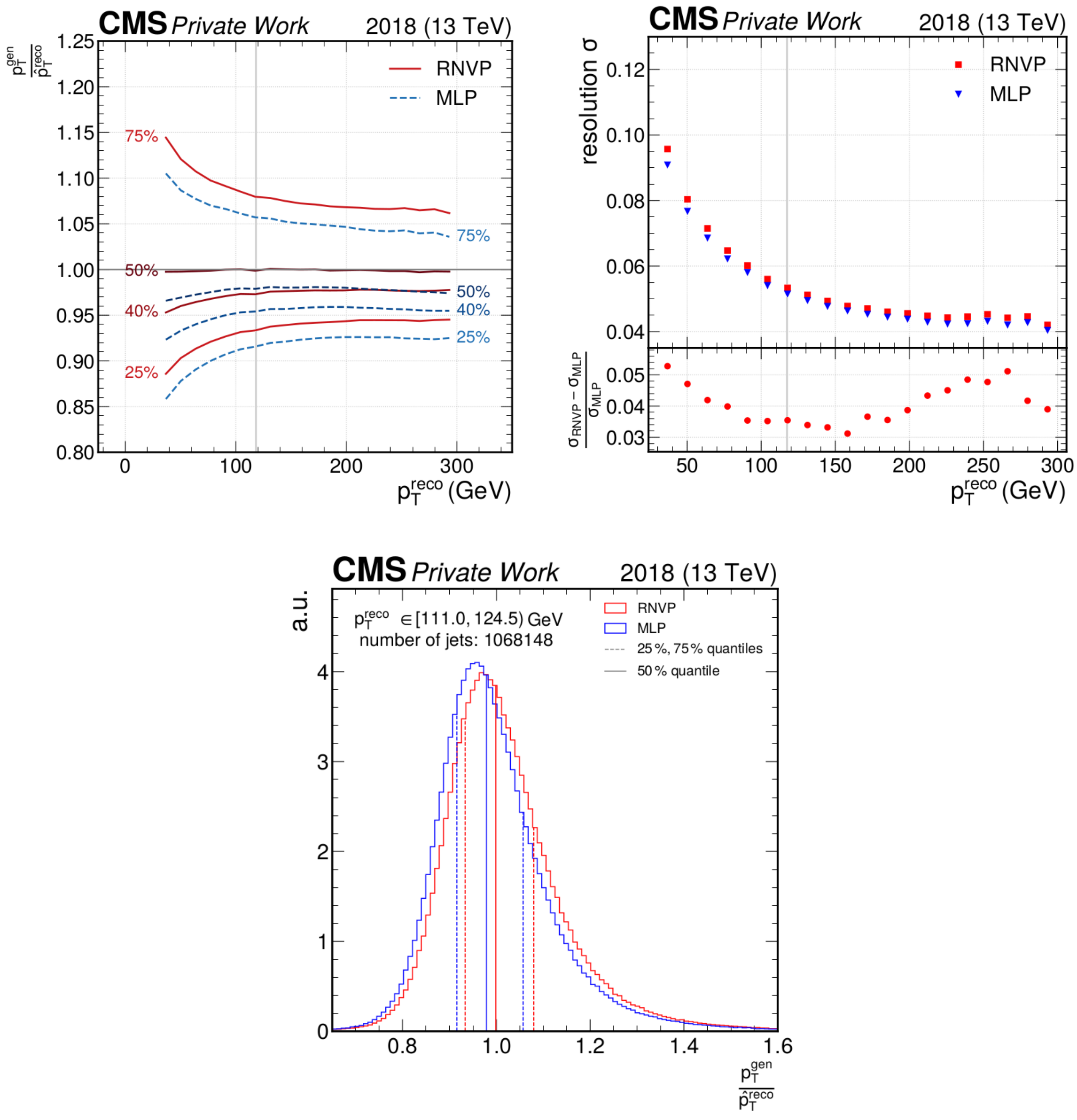


Figure 10.2.: The upper left panel shows the b jet momentum scales as a function of p_T^{reco} without b jet momentum corrections applied for Ω_{RNVP} (red) and Ω_{MLP} (blue). The JMS is binned in p_T. For each bin, the q_{25}, q_{50}, q_{75}, and q_{40} are indicated, with the latter approximately corresponding to the mode of the distribution. The upper right panel illustrates the JMR based on q_{25} and q_{75} visualized in the panel for the JMS. The JMS distribution within one selected bin $[111, 124.5]$ GeV, containing approximately one million b jets, is shown in the lower part. The corresponding bin position is marked as a vertical grey line in the upper panels.

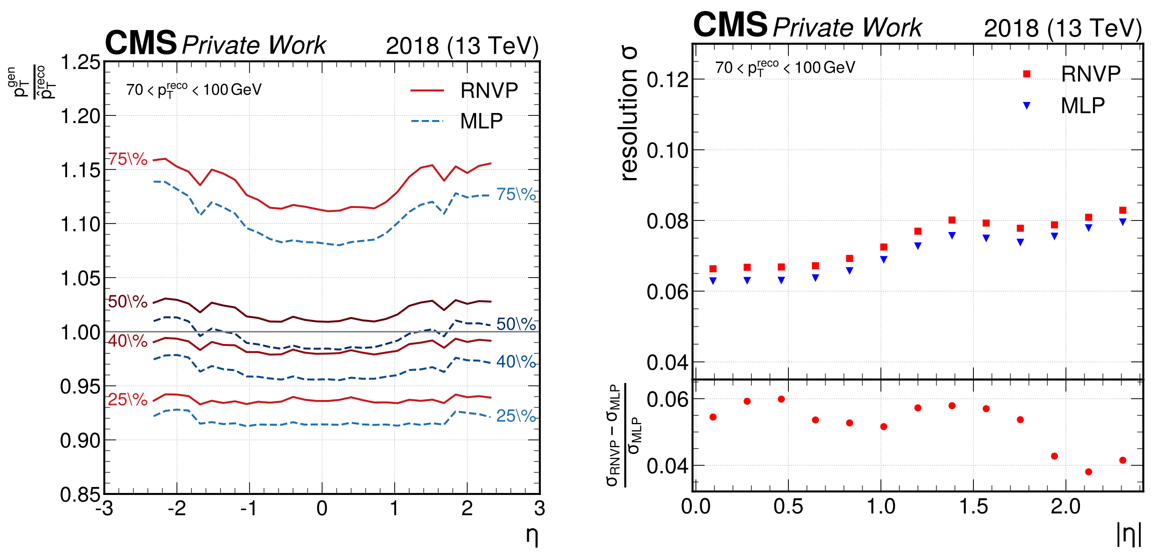


Figure 10.3.: The left panel shows the b jet momentum scales as a function of η for Ω_{RNVP} (red) and Ω_{MLP} (blue). The JMS is binned in η. For each bin, the q_{25}, q_{50}, q_{75}, and q_{40} are indicated, with the latter approximately corresponding to the mode of the distribution. The right panel illustrates the JMR based on q_{25} and q_{75} shown in the panel for the JMS.

As discussed in Section 5.5.2, JMC are primarily derived as functions of the jet pseudorapidity η, transverse momentum p_T, and the median event energy density ρ. Consequently, after applying the b jet momentum corrections, the resulting jet-momentum scale (JMS) and jet-momentum resolution (JMR) are systematically validated by testing their dependence on these variables. Comparisons of the JMS, defined as the scale

$$\frac{p_T^{\text{gen}}}{\hat{p}_T^{\text{reco}}}, \tag{10.3}$$

and the corresponding resolutions σ are presented in Fig. 10.2 as a function of p_T^{reco}. The upper left panel displays the quantile lines of the JMS as a function of p_T^{reco} for both Ω_{RNVP} and Ω_{MLP}. The overall behavior of the quantiles for both models shows the same pattern. At higher p_T^{reco}, both models exhibit stable and narrow JMS distributions, but at lower p_T^{reco}, the distributions broaden, showing a roughly equal amount of over- and underestimations. The entire JMS distribution of Ω_{MLP} is systematically shifted towards smaller values, which is consistent with the overestimation observed in the residuals. In contrast, the median of Ω_{RNVP} remains close to unity across the entire range of p_T^{reco}. The median of Ω_{MLP} shows a downward deviation at low p_T^{reco}, indicating underestimations of $\hat{p}_T^{\text{reco}}$ in this region.

To inspect the width of the JMS at different p_T^{reco} regions in more detail, the JMR is determined and shown in the upper right panel of Fig. 10.2. Here, the JMR obtains resolution values of 0.1in the low-p_T^{reco} regime and converges to values below 0.05 at higher p_T^{reco}, indicating more precise predictions. The ratio suggests that Ω_{MLP} exceeds the Ω_{RNVP} in terms of resolution about 2 % to 6 %.

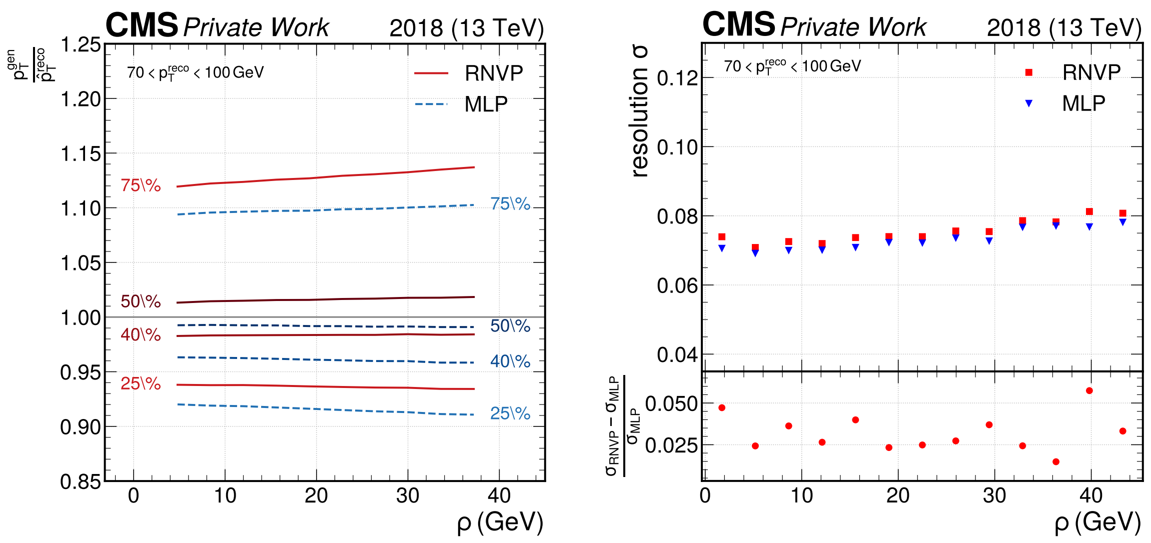
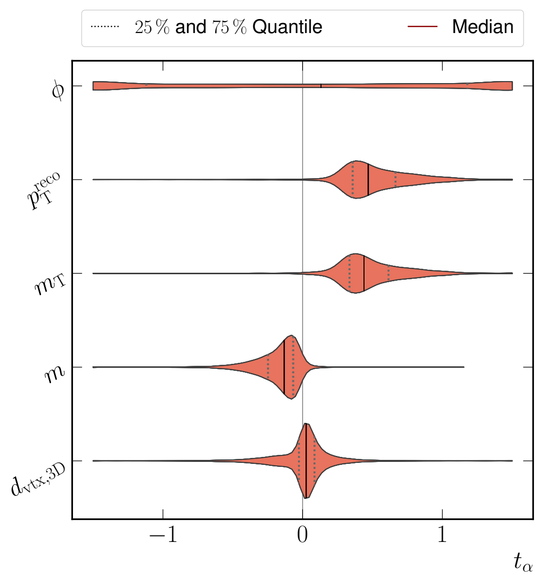


Figure 10.4.: The left panel shows the b jet momentum scales as a function of ρ for the Ω_{RNVP} (red) and Ω_{MLP} (blue) models. The JMS is binned in ρ, while for each bin, the q_{25}, q_{50}, q_{75}, and q_{40} are indicated, with the latter approximately corresponding to the mode of the distribution. The right panel illustrates the JMR based on q_{25} and q_{75} visualized in the panel for the JMS.

As a representative case, the distribution within the bin $p_{\mathrm{T}}^{\mathrm{reco}} \in [111, 124.5]$ GeV is shown in the lower part of Fig. 10.2. With about one million b jets, this bin provides sufficient statistical power to determine reliable quantiles. In contrast to the quantile lines of the JMS, an inspection of the distribution in this bin reveals that both models exhibit an asymmetric JMS shape, with the peak shifted towards lower values and a longer tail extending towards higher JMS values.

Analogous figures for the JMS and JMR are presented as a function of the pseudorapidity η in Fig. 10.3. For the JMS (left), the median and lower quantile curves display an approximately flat behaviour, showing only small dependence on η and therefore on the setup of the CMS detector. However, the 75 % quantiles show increasing values in the forward regions of the CMS detector. The median of the JMS remains closer to unity for Ω_{MLP}, while Ω_{RNVP} shows a slight upward deviation. The right panel, showing the resolution as a function of $|\eta|$, reveals resolution values with an upward trend from 0.065 to 0.09, where Ω_{MLP} outperforms Ω_{RNVP} by up to 6 % in resolution.

Finally, the impact of the pileup conditions on the b jet momentum regression is examined considering the median event energy density ρ, as shown in Fig. 10.4. Here, all JMS quantiles exhibit a slight linear trend in ρ, whereas the outer quantiles, q_{25} and q_{75}, show more pronounced slopes. In contrast, the q_{40} and q_{50} remain mainly constant, indicating that the core of the distribution is stable and largely independent of pileup effects. The resolution in the right panel of the figure confirms the small linear dependence on ρ, with values increasing from 0.07 to 0.085. Again, Ω_{MLP} performs slightly better than Ω_{RNVP}, with advantages of up to 6 %.



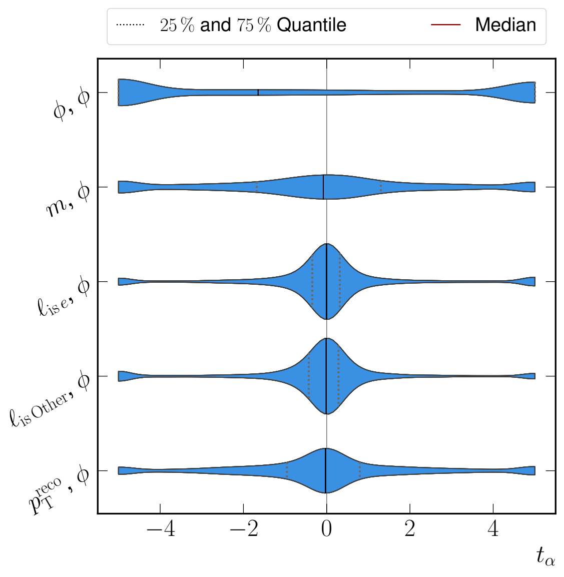


Figure 10.5.: Violin distributions of the first-order (left) and second-order (right) values of t_α, each computed using 10 000 b jets.

In summary, dependencies on p_T^{reco}, η, and ρ were studied in detail. Comparisons of residual distributions revealed similar resolutions for both models, with Ω_{RNVP} exhibiting a nearly unbiased momentum regression. While correlations with η and ρ were minor, a notable dependence on p_T^{reco} was observed, particularly in the low-p_T^{reco} region. The results showed that both models achieve an unbiased response at high p_T^{reco}, with Ω_{RNVP} offering improved performance in terms of bias and Ω_{MLP} showing slightly better resolution in all cases throughout.

10.2. Taylor Coefficient Analysis for the multilayer perceptron

The TCA method, as introduced in Section 6.2, is applied to the Ω_{RNVP} model to investigate which input variables are most relevant for the correction of the b jet momenta. This is

Table 10.1.: The five t_α coefficients with the highest influence on Ω_{RNVP} are shown, sorted by magnitude within each respective order.

1st Order		2nd Order		3rd Order	
α	$\langle t_\alpha \rangle$	α	$\langle t_\alpha \rangle$	α	$\langle t_\alpha \rangle$
ϕ	1.43	$\phi\phi$	33.88	$\phi\phi\phi$	1 035.58
p_T^{reco}	0.54	$m\phi$	3.36	$\ell_{is\,e}\phi\phi$	94.60
m_T	0.50	$\ell_{is\,e}\phi$	2.56	$\ell_{is\,Other}\phi\phi$	89.50
m	0.19	$\ell_{is\,Other}\phi$	2.53	$\ell_{is\,\mu}\phi\phi$	79.60
$d_{vtx,3D}$	0.13	$p_T^{reco}\phi$	2.35	$f_{EM}\phi\phi$	52.61

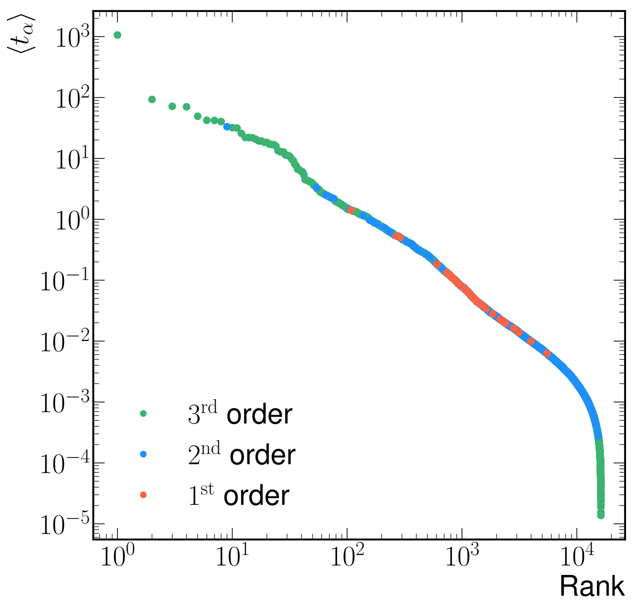


Figure 10.6.: Ranking of all $\langle t_\alpha \rangle$ computed for Ω_{RNVP}. Each $\langle t_\alpha \rangle$ is determined based on 10 000 b jets.

implemented as a proof of concept to demonstrate that the `TaylorAnalysis` framework can also be applied to NF models to compute t_α.

In contrast to the examples presented in Part III, where models were designed with a single output neuron, the Ω_{RNVP} model used in this chapter consists of two output neurons. As a result, the t_α values are computed as the sum of the derivatives with respect to each neuron. All Taylor coefficients are computed up to the third order. The resulting values are ranked by $\langle t_\alpha \rangle$, and the most significant first- and second-order terms are visualized as violin distributions in Fig. 10.5. In addition, Table 10.1 summarizes the values of the five most important coefficients t_α in each order, ranked by their contribution.

Among the first-order coefficients, the most influential feature is the azimuthal angle ϕ of the b jet. As the CMS detector is invariant in ϕ, this sensitivity is not physics motivated: the apparent importance of ϕ arises from the architecture of Ω_{RNVP}, which predicts the squared momentum components $(p_x^2, p_y^2)^{\mathrm{T}}$. To accurately resolve the magnitudes of these components, the azimuthal angle ϕ needs to be known, as discussed previously. Additionally, the distribution of t_ϕ is significantly broader, with larger values, than those of the other t_α. This can be understood since variations in ϕ impact the magnitudes of p_x^2 and p_y^2. In particular, this effect is more pronounced for high-p_{T} jets, where the larger momentum components experience larger changes compared to jets with low p_{T}. Additionally, squaring the components further amplifies this effect.

Another influential feature is $p_{\mathrm{T}}^{\mathrm{reco}}$, which serves as the starting point for the momentum correction and shows a pronounced shift toward positive t_α values. The transverse mass m_{T} exhibits a similar distribution, which can be understood since both $p_{\mathrm{T}}^{\mathrm{reco}}$ and m_{T} are functions of the x and y momentum components of the jet. In contrast, the mass, determined based on the energy and p_T^{reco} without b jet momentum corrections, which additionally includes information of the z momentum component, seems to have less impact on Ω_{MLP}. The fifth most important feature is the 3D distance to the secondary vertex, $\mathrm{d}_{\text{vtx,3D}}$. Since jet lifetime correlates with p_T, this is a consistent result.

For the second-order coefficients, combinations including ϕ dominate the ranking. The coefficient $t_{\phi,\phi}$ has the largest impact and shows a broad distribution. When ϕ is combined with the mass m, the coefficient $t_{m\phi}$ becomes relevant. This pairing allows the model to access energy and momentum regarding all axes through the invariant mass. Although lepton-related features do not appear among the ranked first-order inputs, second-order coefficients coefficients including ϕ and the lepton indicators $\ell_{\text{is},e}$ and $\ell_{\text{is,Other}}$ show a remarkable impact on the model. The similar shapes of the violin plots for both t_α further confirm that the underlying information encoded in these features follows a comparable structure. The fifth most relevant second-order variable is the coefficient $t_{p_T^{\text{reco}}\phi}$. Moreover, the second-order coefficients exhibit a more symmetric structure compared to their first-order counterparts.

The complete ranking of all computed t_α coefficients is shown in Fig. 10.6. As observed in previous studies, the highest values occur among the third-order terms. The most dominant coefficient is $t_{\phi\phi\phi}$ with a value of 1 035.58, followed by $t_{\ell_{\text{is}\,e}\phi\phi}$ and $t_{\ell_{\text{is Other}}\phi\phi}$.

10.3. Per-jet resolution

Both Ω_{MLP} and Ω_{RNVP} are designed to provide per-jet resolution estimates in addition to predicting the transverse momentum of a jet. In the case of Ω_{MLP}, this is achieved by incorporating two quantile loss terms to directly predict q_{25} and q_{75} to be used for resolution estimates. The Ω_{RNVP} model learns a conditional PDF for each jet, from which samples can be drawn to extract the corresponding quantiles q_{25} and q_{75}.

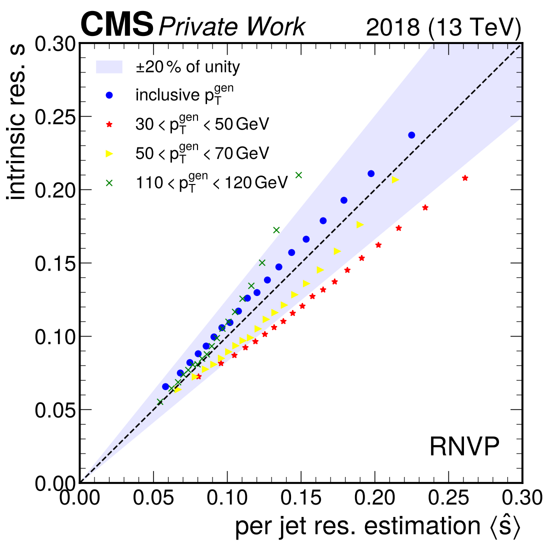
This section presents a comparative analysis of the per-jet resolution estimates produced by both models. The predicted resolution is calculated and grouped into equally populated bins for each jet. Within each bin, the analysis is further subdivided into three p_T regions:

$$\begin{aligned}
\text{low}: &\quad 30 < p_T^{\text{gen}} < 50\,\text{GeV} \\
\text{medium}: &\quad 50 < p_T^{\text{gen}} < 70\,\text{GeV} \\
\text{high}: &\quad 110 < p_T^{\text{gen}} < 120\,\text{GeV}.
\end{aligned}$$

For each bin and p_T range, the intrinsic resolution s is derived based on the p_T^{gen} distribution. These are compared to the mean predicted per-jet resolution

$$\langle s \rangle = \langle \frac{q_{75} - q_{25}}{2} \rangle, \tag{10.4}$$

determined by the models. The results of this comparison are shown in Fig. 10.7. The left panel, corresponding to the Ω_{RNVP} model, indicates that small predicted resolutions fall



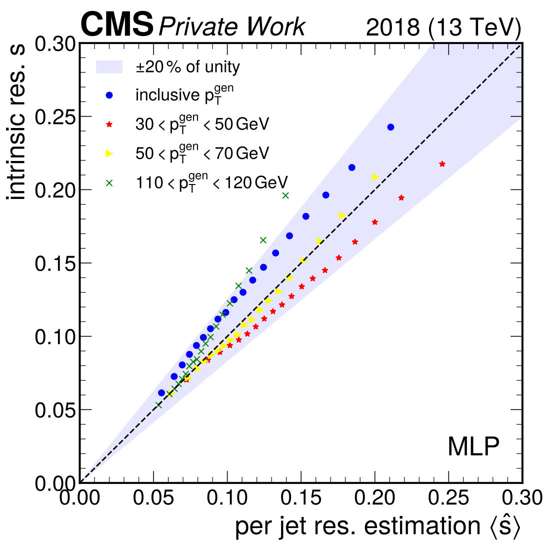


Figure 10.7.: Per-jet resolution for Ω_{RNVP} (left panel) and Ω_{MLP} (right panel). The p_T^{gen}-based intrinsic resolution s is shown as a function of the mean predicted per-jet resolution $\langle \hat{s} \rangle$ for each model. In addition, it is separated into low (red), medium (yellow), and high (green) p_T regions. The ±20% deviations from unity are shown as a blue band for comparison.

within the ±20% deviation band from unity. For broader resolutions, however, the estimates of low and high-p_T regions tend to fall outside this band. In contrast, the predictions in the medium p_T region and the inclusive case remain close to the unity line, suggesting that deviations at the boundaries of the p_T spectrum partially cancel out when aggregated.

The right panel, corresponding to the Ω_{MLP} model, shows that predictions remain within the ±20% bands for both the low and medium p_T regions. Notably, the medium p_T region aligns closely with the unity line, implying accurate resolution estimations. Although the high-p_T region also shows a degree of underestimation, similar to Ω_{RNVP}, the inclusive prediction reveals a slight overestimation, more pronounced than for Ω_{RNVP}.

In summary, the results demonstrate that the approach introduced in Section 9.4, which models a conditional per-jet PDF to derive resolution estimates, is effective and yields competitive results. While both models tend to overestimate the resolution in the high-p_T regime, Ω_{RNVP} further underestimates resolution in the low-p_T region when compared to the Ω_{MLP}. This discrepancy leads to partial compensation in the inclusive p_T category, ultimately resulting in a more balanced overall resolution performance for Ω_{RNVP}.

10.4. H → b$\bar{\text{b}}$ reconstruction

While all previous studies have focused on the single-jet level, many physics analyses depend on the invariant mass m_{bb} of a dijet system. To compare the resolution of m_{bb} as

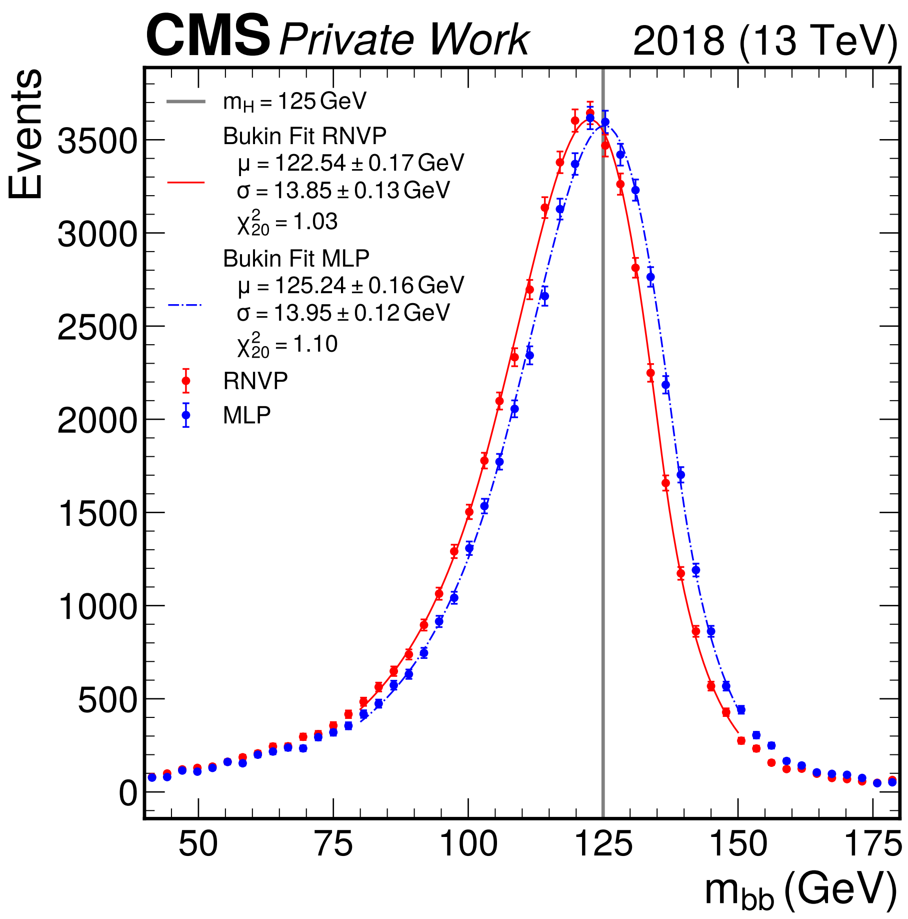


Figure 10.8.: Invariant mass estimation for a dijet system of an H boson candidate based on Ω_{RNVP} (red) and Ω_{MLP} (blue). The optimized parameters for the peak μ, the resolution σ, and corresponding reduced χ^2_{dof} values of the applied fits to obtain these values are given in the legend.

provided by the Ω_{RNVP} and Ω_{MLP} models, simulated $\mathrm{H}(\rightarrow \mathrm{b\bar{b}})Z(\rightarrow \ell\ell)$ events are used to obtain the invariant mass of the H boson derived from the reconstructed jets. Only events containing two oppositely charged leptons $\ell \in \{e^{\pm}, \mu^{\pm}\}$ are selected. Furthermore, the invariant mass of the dilepton system is required to be within 20 GeV of the nominal Z boson mass, i.e., $|m_{\ell\ell} - m_Z| \leq 20\,\mathrm{GeV}$, and the transverse momentum of the reconstructed Z boson is required to exceed 150 GeV.

Additionally, two b-tagged jets passing the medium working point criterion are required. The invariant mass m_{bb} of these two jets is then used to reconstruct the H boson candidate.

The resulting m_{bb} distributions for the Ω_{RNVP} and Ω_{MLP} models are shown in Fig. 10.8. A Bukin function [141] is used to model each mass peak using five parameters: the peak position μ, its width $\sigma = \mathrm{FWHM}/2.35$, a peak asymmetry parameter ρ_{P}, and the parameters ρ_{L} and ρ_{R} describing the left and right tails of the distributions, respectively. The parameter minimization is performed using the `MINUIT` algorithm [142] in 25 bins spanning the invariant mass range from 80 to 150 GeV. For both models, the resulting values of μ and σ, along with the reduced χ^2_{dof} of the modeled Bukin function, are shown in the legend of Fig. 10.8.

The peak obtained from the Ω_{RNVP}-corrected b jets is located at $\mu = 122.54 \pm 0.17$ GeV. In contrast, the Ω_{MLP}-corrected jets yield a peak at $\mu = 125.24 \pm 0.16$ GeV, which is only 0.2 % off the H boson mass of 125 GeV used to simulate the dataset.

In terms of resolution, the Ω_{RNVP} model provides a slightly smaller width, with $\sigma = 13.85 \pm 0.13$ GeV, compared to $\sigma = 13.95 \pm 0.12$ GeV for the Ω_{MLP}. Both curves agree with the data, as indicated by the reduced χ^2_{dof} values of $\chi^2_{20} = 1.03$ for Ω_{RNVP} and $\chi^2_{20} = 1.10$ for Ω_{MLP}.

The outstanding peak position obtained with the Ω_{MLP} corrections can be attributed, in part, to the kinematic distribution of the jets used in the reconstruction: approximately 61.8% of the b jets have a generator-level transverse momentum $p_{\mathrm{T}}^{\mathrm{gen}}$ exceeding 79 GeV. As discussed in Fig. 10.1, this high-p_{T} regime is exactly where the Ω_{MLP} model outperforms Ω_{RNVP} in terms of accuracy. The strength of Ω_{RNVP}, on the other hand, lies in low-p_{T} regions, where it may outperform the Ω_{MLP} model, for example, in scenarios related to top quark decays.

Notably, the Ω_{MLP} model, trained as a benchmark in this study, achieves improved results for both the peak position μ and resolution σ compared to the study in Ref. [134]. This improvement is likely due to advancements in the CMS simulation campaigns between 2016 and 2018.

Conclusion

Machine learning (ML) techniques are increasingly integrated at various stages of high-energy physics (HEP) workflows. Achieving high-precision analyses requires the deployment of powerful models and a solid understanding of how these models make their predictions. This thesis is motivated by the need for robust interpretation methods tailored to the complexity of ML applications in HEP.

The Taylor Coefficient Analysis (TCA) method was applied to address this challenge across a range of representative use cases. As an initial study, the TCA was applied to investigate the learning process of a multilayer perceptron (MLP) trained on two synthetic Gaussian datasets. The analysis revealed how the MLP gradually captures both coarse and fine-grained correlations during training.

The method was then applied to a more realistic classification problem: the discrimination between Higgs boson production in association with a top-quark pair and a background process involving gluon radiation. In this context, physics-motivated key features associated with the Higgs boson were used as inputs to the MLP and compared against edge features in a graph neural network (GNN), which served as their analogous counterparts within the graph-based architecture. The TCA was able to demonstrate the consistent contribution of these features in both architectures.

A dedicated study to assess the relevant orders in the TCA Taylor series demonstrated that first- and second-order coefficients might be sufficient to capture the core dependencies of the given task, which represents the complexity and difficulty of a typical modern classification task in HEP.

Furthermore, a novel approach was developed to correct the missing momentum of b jet candidates due to semileptonic weak decays. A normalizing flow (NF) architecture was utilized to model the probability density function (PDF) of individual b jets directly. The full-scale implementation was benchmarked against a state-of-the-art momentum correction approach implemented in the analysis and reconstruction software of the CMS Experiment. While both models showed comparable robustness with respect to the pseudorapidity and pileup-related energy density per event, deviations emerged in low transverse momentum regions for both models.

The NF generated an almost unbiased estimate for momentum corrections and demonstrated a response and resolution competitive with the benchmark. Still, it did not exceed the benchmark model across all observables. Applying the TCA to the NF model further validated its learned dependencies, especially on transverse quantities and the azimuthal angle mandated by construction.

In summary, this work demonstrates the applicability and versatility of the TCA method across three fundamentally very different ML architectures. Two case studies relevant to HEP further validated its effectiveness, illustrating the scalability of the method with respect to model and data complexity.

Based on the methodological foundation documented in this thesis, future research can extend this work by directly comparing the TCA and other neural network interpretability approaches. In addition, the TCA offers a promising framework for advancing studies on GNNs, particularly in exploring extensions that integrate combinations of vertex-level and graph-level representations.

Abbreviations

AD Antiproton Decelerator

ADAM adaptive moment estimation

AI artificial intelligence

ALICE A Large Ion Collider Experiment

AUC area under curve

BN batch norm

CA Cambridge-Aachen

CDF cumulative density function

CE cross entropy

CERN Conseil européen pour la Recherche Nucléaire

CI confidence interval

CKM Cabibbo–Kobayashi–Maskawa

CMS Compact Muon Solenoid

CMSSW CMS reconstruction and analysis software

CNN convolutional neural network

CSC cathode strip chamber

CSF cubic spline flow

DNN deep neural network

DT drift tube

ECAL electromagnetic calorimeter

GNN graph neural network

GPU graphics processing unit

HCAL hadron calorimeter

HEP high-energy physics

HLT high-level trigger

i.i.d. independent and identically distributed

INN invertible neural network

IP impact parameter

ISOLDE Isotope Separator On Line DEvice

JMC jet-momentum corrections

JMR jet-momentum resolution

JMS jet-momentum scale

KL Kullback-Leibler

KS Kolmogorov–Smirnov

LHC Large Hadron Collider

LHCb Large Hadron Collider beauty

LINAC3 Linear Accelerator 3

LLM large language model

LN layer norm

LO leading order

LOO Leave One Out

MADE Masked Autoencoder for Distribution Estimation

MAF masked autoregressive flow

MC Monte Carlo

ME matrix element

ML machine learning

MLE maximum likelihood estimation

MLP multilayer perceptron

MSE mean squared error

NF	normalizing flow
NLO	next to leading order
NN	neural network
PCA	principal component analysis
PDF	probability density function
PF	Particle Flow
pp	proton-proton
PS	Proton Synchrotron
PV	primary vertex
QCD	quantum chromodynamics
QED	quantum electrodynamics
QFT	quantum field theory
RNVP	Real-value Non-Volume Preserving
ROC	receiver operating characteristic
RPC	resistive plate chamber
SGD	stochastic gradient descent
SHAP	SHapley Additive exPlanations
SM	standard model
SPS	Super Proton Synchrotron
SSB	spontaneous symmetry breaking
SV	secondary vertex
TCA	Taylor Coefficient Analysis
TS	Taylor series
UAT	universal approximation theorem
VEV	vacuum expectation value

Bibliography

[1] CMS Collaboration, "CMS Luminosity - Public Results". Online, 2024. accessed: 2025-04-17. https://twiki.cern.ch/twiki/bin/view/CMSPublic/LumiPublicResults#2024_proton_proton_collisions_at.

[2] CMS Collaboration, "A profile likelihood approach to measure the top quark mass in the lepton+jets channel at $\sqrt{s} = 13$ TeV",.

[3] CMS Collaboration, "Observation of a New Boson at a Mass of 125 GeV with the CMS Experiment at the LHC", *Phys. Lett. B* **716** (2012) 30–61, doi:10.1016/j.physletb.2012.08.021, arXiv:1207.7235.

[4] ATLAS Collaboration, "Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC", *Phys. Lett. B* **716** (2012) 1–29, doi:10.1016/j.physletb.2012.08.020, arXiv:1207.7214.

[5] Royal Swedish Academy of Sciences, "The nobel prize in physics 2023", 2023. accessed: 2025-04-04. https://www.nobelprize.org/prizes/physics/2024/summary/.

[6] S. Wunsch et al., "Identifying the relevant dependencies of the neural network response on characteristics of the input space", *Comput. Softw. Big Sci.* **2** (2018), no. 1, 5, doi:10.1007/s41781-018-0012-1, arXiv:1803.08782.

[7] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain.", *Psychological review* **65 6** (1958) 386–408.

[8] M. Minsky and S. Papert in *Neurocomputing, Volume 1: Foundations of Research*. The MIT Press, 04, 1988. doi:10.7551/mitpress/4943.003.0015.

[9] A. Vaswani et al., "Attention is all you need", *CoRR* **abs/1706.03762** (2017) arXiv:1706.03762.

[10] J. Jumper et al., "Highly accurate protein structure prediction with alphafold", *Nature* **596** (2021), no. 7873, 583–589, doi:10.1038/s41586-021-03819-2.

[11] J. Neyman and E. S. Pearson, "On the Problem of the Most Efficient Tests of Statistical Hypotheses", *Phil. Trans. Roy. Soc. Lond. A* **231** (1933), no. 694-706, 289–337, doi:10.1098/rsta.1933.0009.

[12] V. Nair and G. Hinton, “Rectified linear units improve restricted boltzmann machines vinod nair”, volume 27, pp. 807–814. 06, 2010.

[13] A. L. Maas, “Rectifier nonlinearities improve neural network acoustic models”, 2013.

[14] K. Hornik, M. Stinchcombe, and H. White, “Multilayer feedforward networks are universal approximators”, *Neural Networks* **2** (1989), no. 5, 359–366, `doi:10.1016/0893-6080(89)90020-8`.

[15] G. Cybenko, “Approximation by superpositions of a sigmoidal function”, *Math. Control Signals Syst.* **2** (1989), no. 4, 303–314, `doi:10.1007/BF02551274`.

[16] A. Heinecke, J. Ho, and W.-L. Hwang, “Refinement and universal approximation via sparsely connected relu convolution nets”, *IEEE Signal Processing Letters* **27** (2020) 1175–1179, `doi:10.1109/LSP.2020.3005051`.

[17] D.-X. Zhou, “Universality of deep convolutional neural networks”, 2018. `https://arxiv.org/abs/1805.10769`.

[18] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, “How powerful are graph neural networks?”, *CoRR* **abs/1810.00826** (2018) `arXiv:1810.00826`.

[19] R. B. Gabrielsson, “Universal function approximation on graphs using multivalued functions”, *CoRR* **abs/2003.06706** (2020) `arXiv:2003.06706`.

[20] M. H. Stone, “Applications of the theory of boolean rings to general topology”, *Transactions of the American Mathematical Society* **41** (1937), no. 3, 375–481.

[21] M. H. Stone, “The generalized weierstrass approximation theorem”, *Mathematics Magazine* **21** (1948), no. 4, 167–184.

[22] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors”, *Nature* **323** (1986), no. 6088, 533–536, `doi:10.1038/323533a0`.

[23] A. Paszke et al., “Pytorch: An imperative style, high-performance deep learning library”, 2019. `https://arxiv.org/abs/1912.01703`.

[24] M. Abadi et al., “TensorFlow: Large-scale machine learning on heterogeneous systems”, 2015. Software available from tensorflow.org. `https://www.tensorflow.org/`.

[25] Y. Liu, Y. Gao, and W. Yin, “An improved analysis of stochastic gradient descent with momentum”, 2020. `https://arxiv.org/abs/2007.07989`.

[26] J. Duchi, E. Hazan, and Y. Singer, “Adaptive subgradient methods for online learning and stochastic optimization”, *Journal of Machine Learning Research* **12** (2011), no. 61, 2121–2159.

[27] M. D. Zeiler, “ADADELTA: an adaptive learning rate method”, *CoRR* **abs/1212.5701** (2012) `arXiv:1212.5701`.

[28] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization”, *CoRR* **abs/1412.6980** (2014).

[29] anemptyarchive. Online, 09, 2024. accessed: 2025-03-13. `https://www.anarchive-beta.com/entry/2022/11/16/193000`.

[30] D. M. Himmelblau, “Applied Nonlinear Programming”. McGraw-Hill, 1972. ISBN 0070289212.

[31] J. Hadamard and P. M. Morse, “Lectures on cauchy’s problem in linear partial differential equations”, 1924.

[32] L. Prechelt, “Early stopping-but when?”, in *Neural Networks*. 1996.

[33] S. Sangole, “Understanding L1 and L2 Regularization: The Guardians Against Overfitting”. Online, 2024. accessed: 2025-03-10. `https://medium.com/codex/understanding-l1-and-l2-regularization-the-guardians-against-overfitting-175fa69263dd`.

[34] S. Raschka. Online. accessed: 2025-03-10. `https://sebastianraschka.com/`.

[35] A. N. Tikhonov and V. Y. Arsenin, “Solutions of ill-posed problems”, 1977.

[36] R. Tibshirani, “Regression shrinkage and selection via the lasso”, *J. R. Stat. Soc., B* **58** (1996), no. 1, 267–288.

[37] N. Srivastava et al., “Dropout: A simple way to prevent neural networks from overfitting”, *Journal of Machine Learning Research* **15** (2014), no. 56, 1929–1958.

[38] A. Torralba, P. Isola, and W. Freeman, “Foundations of Computer Vision”. Adaptive Computation and Machine Learning series. MIT Press, 2024. ISBN 9780262378666.

[39] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift”, 2015. `https://arxiv.org/abs/1502.03167`.

[40] J. L. Ba, J. R. Kiros, and G. E. Hinton, “Layer Normalization”, `arXiv:1607.06450`.

[41] L. Euler, “Solutio problematis ad geometriam situs pertinentis”, *Commentarii Academiae Scientiarum Imperialis Petropolitanae* **8** (1736) 128–140.

[42] P. Veličković et al., “Graph attention networks”, 2018. `https://arxiv.org/abs/1710.10903`.

[43] S. R. Qasim et al., “Learning representations of irregular particle-detector geometry with distance-weighted graph networks”, *Eur. Phys. J. C* **79** (2019), no. 7, 608, `doi:10.1140/epjc/s10052-019-7113-9`, `arXiv:1902.07987`.

[44] S. R. Qasim et al., “Multi-particle reconstruction in the high granularity calorimeter using object condensation and graph neural networks”, 2021. https://arxiv.org/abs/2106.01832.

[45] Römert, “Graph example”. Online, 04, 2013. accessed: 2025-03-15. https://commons.wikimedia.org/wiki/File:Graph_example_%28Graph_theory%29.png.

[46] D. J. Rezende and S. Mohamed, “Variational inference with normalizing flows”, 2016. https://arxiv.org/abs/1505.05770.

[47] I. Kobyzev, S. J. Prince, and M. A. Brubaker, “Normalizing flows: An introduction and review of current methods”, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **43** (November, 2021) 3964–3979, doi:10.1109/tpami.2020.2992934.

[48] L. Dinh, J. Sohl-Dickstein, and S. Bengio, “Density estimation using real NVP”, *CoRR* **abs/1605.08803** (2016) arXiv:1605.08803.

[49] G. Papamakarios, T. Pavlakou, and I. Murray, “Masked autoregressive flow for density estimation”, 2018. https://arxiv.org/abs/1705.07057.

[50] C. Durkan et al., “Cubic-spline flows”, 2019. https://arxiv.org/abs/1906.02145.

[51] T. Mohr, “Performance studies with Normalizing Flow transformations”, 2023.

[52] I. Csiszár, “i-divergence geometry of probability distributions and minimization problems”, *Ann. Probab.* **3** (1975), no. 1, 146–158.

[53] S. V. Narayanan and G. Subramanian, “The geometry of planar linear flows”, 2022. https://arxiv.org/abs/2206.07626.

[54] D. P. Kingma and P. Dhariwal, “Glow: Generative flow with invertible 1x1 convolutions”, 2018. https://arxiv.org/abs/1807.03039.

[55] L. Dinh, D. Krueger, and Y. Bengio, “Nice: Non-linear independent components estimation”, 2015. https://arxiv.org/abs/1410.8516.

[56] C.-W. Huang et al., “Neural autoregressive flows”, 2018. https://arxiv.org/abs/1804.00779.

[57] D. P. Kingma et al., “Improving variational inference with inverse autoregressive flow”, 2017. https://arxiv.org/abs/1606.04934.

[58] H. M. Dolatabadi, S. Erfani, and C. Leckie, “Invertible generative modeling using linear rational splines”, 2020. https://arxiv.org/abs/2001.05168.

[59] C. Durkan et al., “Neural spline flows”, 2019. https://arxiv.org/abs/1906.04032.

[60] M. Germain et al., “Made: Masked autoencoder for distribution estimation”, 2015. https://arxiv.org/abs/1502.03509.

[61] M. Steffen, "A simple method for monotonic interpolation in one dimension.", *Astronomy and Astrophysics* **239** (1990) 443–450.

[62] J. F. Blinn, "How to solve a cubic equation, part 4: The 111 case", *IEEE Computer Graphics and Applications* **27** (2007), no. 1, 100–103, `doi:10.1109/MCG.2007.10`.

[63] L. V. Kantorovich, "Mathematical methods of organizing and planning production", *Management Science* **6** (1960), no. 4, 366–422.

[64] H. Goldstein, C. Poole, and J. Safko, "Classical Mechanics". Addison Wesley, 2002. ISBN 9780201657029.

[65] T.-P. Cheng and L.-F. Li, "Gauge Theory of Elementary Particle Physics". Oxford University Press, Oxford, UK, 1984. ISBN 978-0-19-851961-4, 978-0-19-851961-4.

[66] E. Noether, "Invariante variationsprobleme", *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-Physikalische Klasse* **1918** (1918) 235–257.

[67] I. Neutelings, "Standard model", 2025. accessed: 28-02-2025. `https://tikz.net/sm_particles/`.

[68] J. S. Schwinger, "On Quantum electrodynamics and the magnetic moment of the electron", *Phys. Rev.* **73** (1948) 416–417, `doi:10.1103/PhysRev.73.416`.

[69] J. B. Hartle, "The Space-time approach to quantum mechanics", *Vistas Astron.* **37** (1993) 569, `doi:10.1016/0083-6656(93)90097-4`, `arXiv:gr-qc/9210004`.

[70] P. A. M. Dirac, "Quantum theory of emission and absorption of radiation", *Proc. Roy. Soc. Lond. A* **114** (1927) 243, `doi:10.1098/rspa.1927.0039`.

[71] J. D. Jackson, "Classical Electrodynamics". Wiley, 1998. ISBN 978-0-471-30932-1.

[72] L. D. Landau and E. M. Lifschits, "The Classical Theory of Fields", volume 2 of *Course of Theoretical Physics.* Pergamon Press, Oxford, 1975. ISBN 978-0-08-018176-9.

[73] H. Fritzsch, M. Gell-Mann, and H. Leutwyler, "Advantages of the Color Octet Gluon Picture", *Phys. Lett. B* **47** (1973) 365–368, `doi:10.1016/0370-2693(73)90625-4`.

[74] D. J. Gross and F. Wilczek, "Ultraviolet Behavior of Nonabelian Gauge Theories", *Phys. Rev. Lett.* **30** (1973) 1343–1346, `doi:10.1103/PhysRevLett.30.1343`.

[75] H. D. Politzer, "Reliable Perturbative Results for Strong Interactions?", *Phys. Rev. Lett.* **30** (1973) 1346–1349, `doi:10.1103/PhysRevLett.30.1346`.

[76] M. Gell-Mann, "The Eightfold Way: A Theory of strong interaction symmetry", `doi:10.2172/4008239`.

[77] K. G. Wilson, "Confinement of Quarks", *Phys. Rev. D* **10** (1974) 2445–2459, doi:10.1103/PhysRevD.10.2445.

[78] Particle Data Group Collaboration, "Review of Particle Physics (RPP)", *Phys. Rev. D* **86** (2012) 010001, doi:10.1103/PhysRevD.86.010001.

[79] P. W. Higgs, "Broken Symmetries and the Masses of Gauge Bosons", *Phys. Rev. Lett.* **13** (1964) 508–509, doi:10.1103/PhysRevLett.13.508.

[80] F. Englert and R. Brout, "Broken Symmetry and the Mass of Gauge Vector Mesons", *Phys. Rev. Lett.* **13** (1964) 321–323, doi:10.1103/PhysRevLett.13.321.

[81] J. Ellis, "Higgs Physics", doi:10.5170/CERN-2015-004.117, arXiv:1312.5672.

[82] J. Goldstone, "Field Theories with Superconductor Solutions", *Nuovo Cim.* **19** (1961) 154–164, doi:10.1007/BF02812722.

[83] H. Yukawa, "On the interaction of elementary particles. i", *Progress of Theoretical Physics Supplement* **1** (01, 1955) 1–10, doi:10.1143/PTPS.1.1.

[84] R. Catherall et al., "The ISOLDE facility", *J. Phys. G* **44** (2017), no. 9, 094002, doi:10.1088/1361-6471/aa7eba.

[85] S. Baird et al., "The anti-proton decelerator: AD", *Nucl. Instrum. Meth. A* **391** (1997) 210–215, doi:10.1016/S0168-9002(97)00359-8.

[86] UA1 Collaboration, "Experimental Observation of Isolated Large Transverse Energy Electrons with Associated Missing Energy at $\sqrt{s} = 540$ GeV", *Phys. Lett. B* **122** (1983) 103–116, doi:10.1016/0370-2693(83)91177-2.

[87] UA1 Collaboration, "Experimental Observation of Lepton Pairs of Invariant Mass Around 95-GeV/c**2 at the CERN SPS Collider", *Phys. Lett. B* **126** (1983) 398–410, doi:10.1016/0370-2693(83)90188-0.

[88] UA2 Collaboration, "Evidence for $Z^0 \longrightarrow e^+e^-$ at the CERN $\bar{p}p$ Collider", *Phys. Lett. B* **129** (1983) 130–140, doi:10.1016/0370-2693(83)90744-X.

[89] UA2 Collaboration, "Observation of Single Isolated Electrons of High Transverse Momentum in Events with Missing Transverse Energy at the CERN anti-p p Collider", *Phys. Lett. B* **122** (1983) 476–485, doi:10.1016/0370-2693(83)91605-2.

[90] ATLAS Collaboration, "The ATLAS Experiment at the CERN Large Hadron Collider", *JINST* **3** (2008) S08003, doi:10.1088/1748-0221/3/08/S08003.

[91] CMS Collaboration, "The CMS Experiment at the CERN LHC", *JINST* **3** (2008) S08004, doi:10.1088/1748-0221/3/08/S08004.

[92] ALICE Collaboration, "The ALICE experiment at the CERN LHC", *JINST* **3** (2008) S08002, doi:10.1088/1748-0221/3/08/S08002.

[93] LHCb Collaboration, “The LHCb Detector at the LHC”, *JINST* **3** (2008) S08005, doi:10.1088/1748-0221/3/08/S08005.

[94] “High-Luminosity Large Hadron Collider (HL-LHC)”, technical report, Geneva, 12, 2015. doi:10.5170/CERN-2015-005.

[95] CMS Collaboration, “Recent upgrades and results from the cms experiment”, *J. Phys. Conf. Ser.* **912** (oct, 2017) 012005, doi:10.1088/1742-6596/912/1/012005.

[96] CMS Collaboration, “The CMS tracker system project”, technical report, Geneva, 1997.

[97] CMS Collaboration, “The CMS electromagnetic calorimeter project”, technical report, Geneva, 1997.

[98] CMS Collaboration, “The CMS hadron calorimeter project”, technical report, Geneva, 1997.

[99] CMS Collaboration, “The CMS muon project”, technical report, Geneva, 1997.

[100] CMS Collaboration, “CMS muon system towards LHC Run 2 and beyond”, *Nucl. Part. Phys. Proc.* **273-275** (2016) 1014–1022, doi:10.1016/j.nuclphysbps.2015.09.159.

[101] R. E. Kalman, “A New Approach to Linear Filtering and Prediction Problems”, *J. Fluids Eng.* **82** (1960), no. 1, 35–45, doi:10.1115/1.3662552.

[102] T. Speer et al., “Track reconstruction in the CMS tracker”, *Nucl. Instrum. Meth. A* **559** (2006) 143–147, doi:10.1016/j.nima.2005.11.207.

[103] CMS Collaboration, “Description and performance of track and primary-vertex reconstruction with the CMS tracker”, *JINST* **9** (2014), no. 10, P10009, doi:10.1088/1748-0221/9/10/P10009, arXiv:1405.6569.

[104] R. Fruhwirth, W. Waltenberger, and P. Vanlaer, “Adaptive vertex fitting”, *J. Phys. G* **34** (2007) N343, doi:10.1088/0954-3899/34/12/N01.

[105] CMS Collaboration, “Particle-flow reconstruction and global event description with the CMS detector”, *JINST* **12** (2017), no. 10, P10003, doi:10.1088/1748-0221/12/10/P10003, arXiv:1706.04965.

[106] CMS Collaboration, F. Beaudette, “The CMS Particle Flow Algorithm”, in *Int. Conf. Calorim. High Energy Front.*, pp. 295–304. 2013. arXiv:1401.8155.

[107] S. Catani et al., “New clustering algorithm for multi - jet cross-sections in e+ e- annihilation”, *Phys. Lett. B* **269** (1991) 432–438, doi:10.1016/0370-2693(91)90196-W.

[108] S. Catani, Y. L. Dokshitzer, and B. R. Webber, "The K^{-} perpendicular clustering algorithm for jets in deep inelastic scattering and hadron collisions", *Phys. Lett. B* **285** (1992) 291–299, doi:10.1016/0370-2693(92)91467-N.

[109] M. Cacciari, G. P. Salam, and G. Soyez, "The anti-k_t jet clustering algorithm", *JHEP* **04** (2008) 063, doi:10.1088/1126-6708/2008/04/063, arXiv:0802.1189.

[110] Y. L. Dokshitzer, G. D. Leder, S. Moretti, and B. R. Webber, "Better jet clustering algorithms", *JHEP* **08** (1997) 001, doi:10.1088/1126-6708/1997/08/001, arXiv:hep-ph/9707323.

[111] R. Atkin, "Introduction to Jet Reconstruction Algorithms". Online, 02, 2015. accessed: 2025-03-02. https://indico.cern.ch/event/367368/contributions/1783356/attachments/730376/1002150/HEPP2015_Presentation.pdf.

[112] G. P. Salam, "Towards Jetography", *Eur. Phys. J. C* **67** (2010) 637–686, doi:10.1140/epjc/s10052-010-1314-6, arXiv:0906.1833.

[113] CMS Collaboration, "Description and performance of track and primary-vertex reconstruction with the cms tracker", *JINST* **9** (oct, 2014) P10009, doi:10.1088/1748-0221/9/10/P10009.

[114] CMS Collaboration, "Jet Energy Scale and Resolution Measurements in CMS", *PoS* **ICHEP2022** (2022) 652, doi:10.22323/1.414.0652, arXiv:2301.02175.

[115] CMS Collaboration, "Identification of heavy-flavour jets with the CMS detector in pp collisions at 13 TeV", *JINST* **13** (2018), no. 05, P05011, doi:10.1088/1748-0221/13/05/P05011, arXiv:1712.07158.

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

[117] S. Alioli et al., "A general framework for implementing NLO calculations in shower Monte Carlo programs: the POWHEG BOX", *JHEP* **06** (2010) 043, doi:10.1007/JHEP06(2010)043, arXiv:1002.2581.

[118] P. Artoisenet et al., "Automatic spin-entangled decays of heavy resonances in Monte Carlo simulations", *JHEP* **03** (2013) 015, doi:10.1007/JHEP03(2013)015, arXiv:1212.3460.

[119] T. Sjöstrand et al., "An introduction to PYTHIA 8.2", *Comput. Phys. Commun.* **191** (2015) 159–177, doi:10.1016/j.cpc.2015.01.024, arXiv:1410.3012.

[120] GEANT4 Collaboration, "GEANT4 - A Simulation Toolkit", *Nucl. Instrum. Meth. A* **506** (2003) 250–303, doi:10.1016/S0168-9002(03)01368-8.

[121] DELPHES 3 Collaboration, "DELPHES 3, A modular framework for fast simulation of a generic collider experiment", *JHEP* **02** (2014) 057,

doi:10.1007/JHEP02(2014)057, arXiv:1307.6346.

[122] S. Wold, K. Esbensen, and P. Geladi, “Principal component analysis”, *Chemometrics and Intelligent Laboratory Systems* **2** (1987), no. 1, 37–52, doi:https://doi.org/10.1016/0169-7439(87)80084-9.

[123] J. Lei et al., “Distribution-free predictive inference for regression”, *Journal of the American Statistical Association* **113** (2018), no. 523, 1094–1111, doi:10.1080/01621459.2017.1307116, arXiv:https://doi.org/10.1080/01621459.2017.1307116.

[124] S. Lundberg and S.-I. Lee, “A unified approach to interpreting model predictions”, 2017. https://arxiv.org/abs/1705.07874.

[125] C. Molnar, “Interpretable Machine Learning”. Online, 03, 2025. accessed: 2025-04-08. https://christophm.github.io/interpretable-ml-book/.

[126] S. Wunsch, “Modern machine learning in the presence of systematic uncertainties for robust and optimized multivariate data analysis in high-energy particle physics”. PhD thesis, KIT, Karlsruhe, 2021.

[127] L. Sowa and A. Monsch, “lsowa/tayloranalysis: Initial release”, 03, 2025. doi:10.5281/zenodo.14998996, https://doi.org/10.5281/zenodo.14998996.

[128] G. Van Rossum and F. L. Drake, “Python 3 Reference Manual”. CreateSpace, Scotts Valley, CA, 2009. ISBN 1441412697.

[129] P. D. Group et al. *Progress of Theoretical and Experimental Physics* **2022** (08, 2022) 083C01, doi:10.1093/ptep/ptac097, arXiv:https://academic.oup.com/ptep/article-pdf/2022/8/083C01/49175539/ptac097.pdf.

[130] R. P. Feynman, “The theory of positrons”, *Phys. Rev.* **76** (Sep, 1949) 749–759, doi:10.1103/PhysRev.76.749.

[131] E. Pfeffer et al., “A Case Study of Sending Graph Neural Networks Back to the Test Bench for Applications in High-Energy Particle Physics”, *Comput. Softw. Big Sci.* **8** (2024), no. 1, 13, doi:10.1007/s41781-024-00122-3, arXiv:2402.17386.

[132] J. Alwall et al., “MadGraph 5 : Going Beyond”, *JHEP* **06** (2011) 128, doi:10.1007/JHEP06(2011)128, arXiv:1106.0522.

[133] M. Fey and J. E. Lenssen, “Fast graph representation learning with PyTorch Geometric”, in *ICLR Workshop on Representation Learning on Graphs and Manifolds*. 2019.

[134] CMS Collaboration, “A Deep Neural Network for Simultaneous Estimation of b Jet Energy and Resolution”, *Comput. Softw. Big Sci.* **4** (2020), no. 1, 10, doi:10.1007/s41781-020-00041-z, arXiv:1912.06046.

[135] CMS Collaboration, "Description and performance of track and primary-vertex reconstruction with the cms tracker", *JINST* **9** (oct, 2014) P10009, doi:10.1088/1748-0221/9/10/P10009.

[136] CMS Collaboration, "Description and performance of track and primary-vertex reconstruction with the cms tracker", *JINST* **9** (oct, 2014) P10009, doi:10.1088/1748-0221/9/10/P10009.

[137] P. J. Huber, "Robust Estimation of a Location Parameter", *Ann. Math. Stat.* **35** (1964), no. 1, 73 – 101, doi:10.1214/aoms/1177703732.

[138] R. W. Koenker and G. Bassett, "Regression quantiles", *Econometrica* **46** (1978), no. 1, 33–50.

[139] J. Vermorel, "Pinball loss function". Online, 2012. accessed: 2025-04–5. https://www.lokad.com/pinball-loss-function-definition/.

[140] L. Ardizzone et al., "Framework for easily invertible architectures (freia)", 2018-2022. accessed: 21-04-2025. https://github.com/vislearn/FrEIA.

[141] A. D. Bukin, "Fitting function for asymmetric peaks", 2007. https://arxiv.org/abs/0711.4449.

[142] F. James and M. Roos, "Minuit: A System for Function Minimization and Analysis of the Parameter Errors and Correlations", *Comput. Phys. Commun.* **10** (1975) 343–367, doi:10.1016/0010-4655(75)90039-9.

Danksagung

Zuallererst möchte ich mich herzlich bei meinem Doktorvater, Prof. Dr. Markus Klute, für seine Betreuung und Unterstützung während meiner Promotionszeit bedanken.

Mein besonderer Dank gilt meinem Betreuer Dr. Roger Wolf, der mich mit außergewöhnlicher Verfügbarkeit und Engagement durch meine gesamte Doktorarbeit begleitet hat. Seine Unterstützung war für mich von goßem Wert.

Ich danke dem Institut für Experimentelle Teilchenphysik (ETP) für die Möglichkeit und den damit verbundenen Aufwand, meine Promotion durchführen zu können, sowie allen Mitarbeitenden für ihren Beitrag.

Ein herzliches Dankeschön gilt meinen Kolleginnen und Kollegen des ETP für die kollegiale und stets angenehme Arbeitsatmosphäre. Der fachliche Austausch, gemeinsame Ausflüge und humorvolle Momente haben meine Zeit als Doktorand besonders gemacht.

Mein weiterer Dank gilt Artur Monsch für die bereichernde Zusammenarbeit an unseren gemeinsamen Projekten. Ebenso danke ich Torben Mohr, Simon Daigler und Max Venus für die produktive und bereichernde Zusammenarbeit.

Hervorheben möchte ich auch meine Bürokollegen Nils Faltermann und Jost von den Driesch – für ihre freundschaftliche Begleitung, die stets angenehme Stimmung im Büro und viele gemeinsame Stunden, die diese Zeit besonders gemacht haben.

Schließlich danke ich meiner Familie, insbesondere meinen Eltern, für ihre bedingungslose Unterstützung und ihren beständigen Rückhalt auf meinem gesamten Weg.