VRIJE
UNIVERSITEIT
BRUSSEL

Thesis submitted in fulfilment of the requirements for the award of the degree of Doctor of Science

DEEP LEARNING FOR JET ALGORITHM

Alexandre De Moor

December 5, 2024

Promotor: Prof. Dr. Jorgen D'Hondt (VUB)

Jury: Prof. Dr. Michael Tytgat (VUB), chairman
Prof. Dr. Alberto Mariotti (VUB), secretary
Prof. Dr. Sophie de Buyl (VUB)
Prof. Dr. Dominique Maes (VUB)
Prof. Dr. Loukas Gouskos (Brown University)
Prof. Dr. Pascal Vanlaer (ULB)

Faculty of Sciences and Bioengineering Sciences
Department of Physics and Astrophysics

The earth teaches us more about ourselves than all the books in the world, because it is resistant to us. Self-discovery comes when man measures himself against an obstacle.

***Wind, Sand and Stars**, Antoine de Saint-Exupéry (1939)*

Acknowledgement

First of all, I would like to thank my supervisor, Jorgen D'Hondt, for trusting me and advising and guiding me so well during the years of my doctorate. I can only acknowledge how fortunate I was to have enjoyed such freedom in the direction of my research and the responsibilities entrusted to me. My fulfilment in this environment would not have been the same without such confidence and encouragement in our relationship.

I would also like to thank Denise, Emil, and Gerrit, whose experience and knowledge sharing have allowed me to learn many skills necessary for the successful completion of my projects. Your patience and help are invaluable, as they have enabled me to progress steadily in this thesis.

I would also like to thank my colleagues at the CMS experiment, Sebastian, Spandan, Raffaele, Stephane, Huilin, Congqiao, Sitian, and Ming-Yan, whose rich collaborations and discussions were an endless source of ideas during my doctoral research. Having had the opportunity to work and exchange with such brilliant people has allowed me to learn so much and surpass what I thought were my limits. I would also like to thank and acknowledge all the collaborators of the BTV group for the extraordinary amount of work accomplished and whose unwavering resilience is an actual model of team spirit. The working atmosphere in this group was always exceptional thanks to the collaborators, in both good times and bad.

I also extend my thanks to all my collaborators at IIHE, Nordin, Felix, Juhee, Pavlo, Jas, Soumya, Martin, Senne, A.R., Max, Pierre-Alexandre, Eduardo, Kunal, Golnaz, and Arjun. I could not have

dreamed of a better working environment thanks to you and the incredible memories created within these walls.

Finally, I would like to thank my family, especially my parents and my partner, from the bottom of my heart. None of the work I have accomplished today would have been possible without you. You are the solid foundation I can rely on to build my future and the compass that helps me find my way in this world.

Love you all,
Alexandre

Abstract

The development and improvement of deep learning techniques over the past decades have created new opportunities for algorithmic methods in high-energy physics. Particularly, deep learning has led to significant advances in the performance achieved of algorithms for the flavour identification of jets, the structures formed by the fragmentation of a quark or a gluon when produced in a collider such as the CERN Large Hadron Collider.

In this doctoral thesis, we focus on deep learning methods to enhance the performance of jet flavour identification algorithms at the CMS experiment. We aim to extend their capabilities by improving model robustness against changes that may be applied to the variables used by the algorithms. Additionally, by extending their initial tasks, we enable new opportunities for future research. First, we explore the Transformer architectures in the context of creating deep neural networks that preserve the structure of jets. We establish two models whose performance and computational cost set a new state-of-the-art in the field. Second, we introduce a data-agnostic training method based on adversarial attacks, improving the model's robustness against changes in the distribution of input variables. Enhancing robustness is necessary to improve our models' performance after calibration. Finally, we successfully extend the algorithms' tasks to identify hadronic taus and to estimate jet energy corrections and resolutions. Additionally, we introduce the identification of strange jets, a first for an experiment at the LHC.

Ultimately, this doctoral work results in the creation of a new class of models with improved architecture, training methods, and an expanded scope of what an artificial neural network may achieve.

The final model produced, named UParT, serves as the state-of-the-art in jet identification for the CMS experiment at the LHC. With the identification of jets originating from strange quarks being a first for the LHC, new analyses targeting final states containing this type of jet can now be pursued once the new model is calibrated.

Samenvatting

De ontwikkeling en verbetering van deep learning-technieken in de afgelopen decennia hebben nieuwe mogelijkheden gecreëerd voor algoritmische methoden in de hoge-energiefysica. In het bijzonder heeft deep learning geleid tot een aanzienlijke vooruitgang in de prestaties van algoritmen voor de smaakidentificatie van jets, de structuren die worden gevormd door de fragmentatie van een quark of gluon wanneer deze worden geproduceerd in een deeltjesversneller zoals de CERN Large Hadron Collider.

In dit doctoraat richten we ons op deep learning-methoden om de prestaties van algoritmen voor de smaakidentificatie van jets te verbeterenbij het CMS experiment. We streven ernaar hun mogelijkheden uit te breiden door de robuustheid van de modellen te vergroten tegen veranderingen toegepast op de variabelen die door de algoritmen worden gebruikt. Daarnaast creëren we door het uitbreiden van hun initiële taken nieuwe perspectieven voor toekomstig onderzoek. Eerst onderzoeken we de Transformer-architecturen in de context van de creatie van diepe neurale netwerken die de structuur van jets behouden. We ontwikkelen twee modellen waarvan de prestaties en computationele kost een nieuwe state-of-the-art in het veld vestigen. Vervolgens introduceren we een data-agnostische trainingsmethode gebaseerd op adversariële aanvallen, die de robuustheid van het model verbetert tegen veranderingen in de distributie van input variabelen. Het vergroten van de robuustheid is noodzakelijk om de prestaties van onze modellen na kalibratie te optimaliseren. Ten slotte breiden we de taken van de algoritmen met succes uit om hadronische tau leptonen te identificeren en om jet-energiecorrecties en resoluties te schatten. Ook introduceren we de identificatie van

strange jets, een primeur voor een experiment bij de LHC.

Het resultaat van dit doctoraat is de creatie van een nieuwe klasse van modellen met verbeterde architectuur, trainingsmethoden, en een uitgebreidere omvang van wat een kunstmatig neuraal netwerk kan bereiken. Het finale model, genaamd UParT, vertegenwoordigt de state-of-the-art in jet-identificatie voor het CMS experiment bij de LHC. Met de identificatie van jets afkomstig van strange quarks, een primeur voor de LHC, kunnen nu nieuwe analyses worden uitgevoerd die zich richten op eindtoestanden met dit type jet, zodra het nieuwe model is gekalibreerd.

List of abbreviations

Chapter 1

$CERN$	European Organization for Nuclear Research
LHC	Large Hadron Collider
CMS	Compact Muon Solenoid
$CMSSW$	CMS Software
$ECAL$	Electromagnetic Calorimeter
$HCAL$	Hadronic Calorimeter
PV	Primary Vertex
SV	Secondary Vertex
PF	Particle-Flow
AVF	Adaptive Vertex Fitter
IVF	Inclusive Vertex Finder
$AK4$ jet	Anti-kt jet with a radius of 0.4
$PUPPI$	Pileup Per Particle Identification
JEC	Jet Energy Correction
$JERC$	Jet Energy Resolution Correction

Chapter 2

AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
AUC	Area Under Curve
ROC	Receiver Operating Characteristic
MSE	Mean Squared Error
CE	Cross Entropy
CV	Computer Vision
NLP	Natural Language Processing
MLP	Multi-Layer Perceptron
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
$LSTM$	Long Short-Term Memory
SGD	Stochastic Gradient Descent
EMA	Exponential Moving Average

Chapter 3

CSV	Combined Secondary Vertex
DJT	DeepJet Transformer
$ParT$	Particle Transformer
GNN	Graph Neural Network
CPF	Charged Particle-Flow Candidate
NPF	Neutral Particle-Flow Candidate
$FP16$	16-bit Floating Point
$SDPA$	Scaled Dot-Product Attention
MHA	Multi-Head Attention
$P\text{-}MHA$	Particle Multi-Head Attention
$K\text{-}NN$	k-Nearest Neighbors

Chapter 4

SAM	Sharpness-Aware Minimization
$FGSM$	Fast Gradient Sign Method
PGD	Projected Gradient Descent
$JSMA$	Jacobian-based Saliency Map Attacks
CW	Carlini & Wagner Attack
NGM	Normed Gradient Method
$R\text{-}NGM$	Rectified Normed Gradient Method
GM	Gradient Masking
IG	Integrated Gradients
$TRADES$	TRadeoff-inspired Adversarial Defense
KL	Kullback-Leibler Divergence

Chapter 5

τ_h	Hadronically decaying tau lepton
$UParT$	Unified Particle Transformer
Qp	Charge Weighted Momentum
ΔE	Neutral particles difference in energy deposition of calorimeters

Contents

Chapter 1

Introduction

The discovery of the Higgs boson in 2012 by the CMS and ATLAS experiments [1,2] at the LHC marked a significant advancement in particle physics by completing the Standard Model of particle physics. This scalar boson arises from the mechanism explaining the origin of particle mass, predicted almost 50 years earlier [3, 4], for which François Englert and Peter Higgs were awarded the Nobel Prize in 2013. This discovery is the latest in a series of successes of the Standard Model in describing the fundamental interactions. Since then, the high-energy collisions at the LHC have been thoroughly examined, and the Standard Model has consistently demonstrated its robustness by accurately predicting observed phenomena. However, the Standard Model cannot account for all physical phenomena, as it fails to incorporate certain key properties such as the mass of neutrinos, nor does it provide a description of gravitation as a fundamental force. Consequently, research in particle physics continues, with the objective of further refining the precision of our measurements. In doing so, we increase our sensitivity to any potential deviations from the Standard Model's predictions, which, based on current measurements, are expected to be extremely small.

In this context, the development and application of algorithms that enhance the efficiency and robustness of our predictions is crucial. In this thesis, we will focus particularly on jets, the products of quark and gluon hadronization, and how the use of Deep Learning can improve the identification and energy prediction of these jets.

Deep Learning is a subfield of Machine Learning that focuses on artificial neural networks, with the first model developed in 1957 by Frank Rosenblatt [5]. This field has seen significant growth due to advances in training methods, notably by Geoffrey Hinton, whose work on machine learning [6] earned him numerous distinctions, including the Turing Award in 2018 and the Nobel Prize in Physics in 2024 along with John Hopfield [7]. The evolution of learning techniques, combined with recent discoveries in the structure of artificial networks and increased tensor computation capacity, has led to the adoption of Deep Learning in many academic and industrial fields since the 2010s. In this context, we will explore in this thesis some of the latest Deep Learning advancements in jet algorithms and their implications for the evolution of the tasks assigned to them.

1.1 The Standard Model

The Standard Model of particle physics [3, 4, 8–12] is the most advanced theory to date for describing the fundamental interactions between the elementary constituents of matter. Developed in the early 1970s, this theoretical framework unifies three of the four fundamental forces of nature: electromagnetism, the weak interaction, and the strong interaction, leaving gravity outside its scope. It is based on the classification of elementary particles into two main categories: the twelve fermions, with spin $\frac{1}{2}$, which are the building blocks of matter and the four gauge bosons, with spin 1, which mediate the fundamental interactions. Additionally, there is the famous Brout-Englert-Higgs boson, more commonly known as the Higgs boson, which relates to the mechanism of how elementary particles get their mass. This model has been remarkably successful in making experimental predictions. It allows for the measurement and prediction of interactions and physical properties of its fundamental elements, whose nature and physical properties, such as the mass, can vary greatly from one element to another.

Despite its success in predicting most of the properties of the observed fundamental particles, the Standard Model fails to explain

certain physical phenomena, such as neutrino oscillations and their masses, the existence of dark matter and dark energy, or the matter-antimatter asymmetry in the Universe.

The fermions

In the context of the Standard Model, fermions are elementary particles characterised by a half-integer spin. As solutions of the Dirac Equation (1.1), they obey Fermi-Dirac statistics and adhere to the Pauli exclusion principle. In quantum field theory (QFT), fermions are described by spinor fields, which are solutions to the Dirac equation [11]. Thus, each fermion has an antiparticle with identical mass but opposite charges related to the fundamental forces:

$$(i\gamma^{\mu}\partial_{\mu} - m)\psi = 0 \tag{1.1}$$

where the spinor field ψ represents the quantum state of the fermion, m is the mass of the fermion, ∂_{μ} is the partial derivative with respect to the coordinate x_{μ}, and γ^{μ} are the Dirac matrices.

Fermions can be divided into two subsets: quarks and leptons. For each of these subsets, the six constituents are subdivided into three generations, each consisting of two fermions, as illustrated in Figure 1.1.

Leptons are a subcategory of fermions that do not interact via the strong interaction. There are six leptons: the electron (e), the muon (μ), the tau (τ), and their associated neutrinos (ν_{e}), (ν_{μ}), (ν_{τ}). Each lepton has an associated lepton quantum number conserved in interactions. Neutrinos are electrically neutral particles that only interact via the weak interaction, and they interact so weakly that they are particularly difficult to detect. The detectors used in particle colliders, such as the one employed in this doctoral thesis, are incapable of directly detecting and reconstructing neutrinos from interactions. Charged leptons, such as the electron, interact via the electromagnetic or weak interaction. They carry an electric charge of $-q_{\mathrm{e}} = -1.602 \times 10^{-19}$ C.

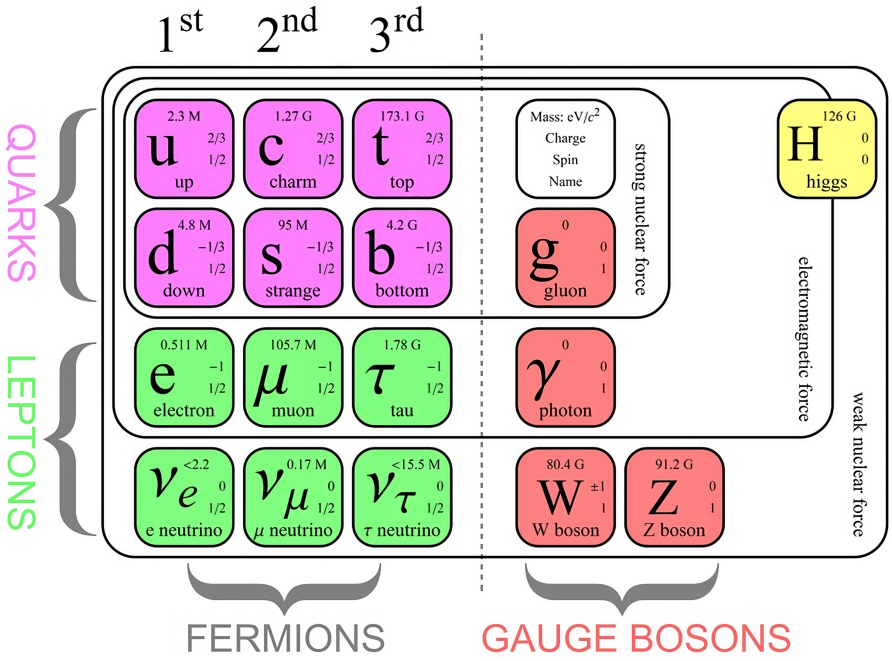

Figure 1.1: Summary table of the Standard Model of particle physics with the quarks (purple), the leptons (green), the gauge bosons (red) and the Higgs boson (yellow) [13].

Quarks are the fundamental constituents of hadronic matter. They are divided into three generations, coming with increasing masses: the up (u) and down (d) quarks form the first generation and are the main constituents of the stable hadronic matter of the Universe, such as the protons and neutrons in atomic nuclei. The charm (c) and strange (s) quarks form the second family, and the top (t) and bottom (b) quarks form the third. Quarks carry a colour charge, making them subject to the strong interaction, which is mediated by gluons. They also interact via the weak interaction and, due to their non-zero fractional electric charge, respectively $\frac{2}{3}q_e$ and $\frac{-1}{3}q_e$ for the up-like (u, c, t) and down-like (d, s, b) quarks, through the electromagnetic interaction. It should be noted that the masses of the quarks in the last generation are particularly high. The top quark has a measured mass of around ∼173 GeV [14–16], the largest of all fundamental particles, resulting in a very short lifetime of about 5×10^{-25} s. This short lifetime allows the top quark to decay before it hadronises, a phenomenon related to the strong interaction

we will describe later. The bottom quark, on the other hand, with a mass of approximately 4.18 GeV [14, 17, 18], remains significantly heavy compared to the other fermions, even though its mass is more than 40 times smaller than the top quark mass.

Gauge theory and fundamental interactions

While fermions describe the constituents of the Universe, gauge bosons are the mediators of the fundamental forces that allow these constituents to interact with each other [11, 12, 19]. The Standard Model describes three fundamental interactions: the electromagnetic interaction, the weak interaction, and the strong interaction.

Electric and magnetic phenomena are unified in the theory of electromagnetism which describes a force that that affects particles with an electric charge, and its gauge boson is the photon γ, a vector boson with zero electric charge and no mass. Being stable and massless, the photon allows electromagnetism to be a long-range. Electromagnetism is described by Quantum Electrodynamics (QED). The weak interaction affects all particles in the Standard Model. Its gauge bosons are the charged $W^{\pm}$ bosons and the neutral Z boson. Unlike the photon, these three gauge bosons are massive, with a mass of $\sim$80.38 GeV [14] and $\sim$91.19 GeV [14], respectively, and have a very short lifetime ($c\tau \sim 10^{-16}$ m), thus limiting the range of this interaction. This short distance can be compared to the charge radius of a proton or an atomic nucleus. The latter has a value of $\sim 1-10\times10^{-15}$ m, which is one to two orders of magnitude larger. This highlights the short-range nature of the weak interaction compared to the strong interaction, which holds nuclei together.

The strong interaction affects particles with a colour charge, with quarks being the only fermions carrying it. This force is mediated by gluons, the gauge bosons associated with the strong interaction. There are eight gluons, each carrying a double colour charge. Gluons are massless and have no electric charge. Due to their double colour charge, gluons can interact not only with quarks but also with themselves. The theory describing this interaction is Quantum Chromodynamics (QCD). QCD explains how quarks and gluons are

confined into hadrons. QCD is also characterised by asymptotic freedom of its interaction strength, where the coupling constant decreases as the energy increases. The coupling constant is too high in low-energy regimes, < 1 GeV, and therefore, in this regime, the QCD theory cannot be developed using a perturbative approach. Due to its high coupling and the gluon self-interaction, the strong force is constrained to be a short-range interaction within hadrons and the nuclei of atoms. Colour confinement prohibits the existence of isolated fermions with a non-zero colour charge. This forces quarks and gluons to combine into bound states with a neutral colour charge. Most of the colour-neutral states observed are hadrons, which are combinations of quarks. Hadrons can be divided into two groups: mesons, which are quark-antiquark bound states, such as pions, and baryons, which are three-quark states, such as protons and neutrons. Finally, it should be noted that QFT also allows for more exotic states such as tetraquarks, pentaquarks, or sexaquarks (four, five or six quarks bound together), as well as glueballs (no valence quarks, only gluons).

The properties of QCD applied to a final-state quark generate a very specific signature. First, the incoming or outgoing partons (quarks or gluons) in a pp collision are colour charged, and their evolution involves a cascade of QCD radiations. These radiations are also called initial state radiation (ISR) if originating from an initial colliding parton or final state radiation (FSR) if originating from an outgoing parton. The cascade process of the radiation, also called parton showering, occurs in the perturbative regime of QCD, above a certain Λ_{QCD} energy scale, about which the coupling strength of the strong interaction α_s is lower than 1, and QDC can be treated perturbatively. The value of Λ_{QCD} depends on theoretical assumptions and is of the order of magnitude of 100 MeV. The parton showering is defined by the Dokshitzer-Gribov-Lipatov-Altarelli-Parisi (DGLAP) equations [20–22] that depict the perturbative evolution of the cascade of QCD radiation. The DGLAP equations describe the probability of an initial parton, referred to as 'mother', to split into 'daughter' partons over time. The energy scale of the initial parton, Q^2, is then split into the 'daughter' partons. A 'daughter' parton will also

evolve perturbatively if its energy scale remains higher than Λ_{QCD} and obey the DGLAP equations. Therefore, the cascade of QCD radiations continues until reaching the non-perturbative energy scale Λ_{QCD}.

When reaching the non-perturbative energy scale, perturbative QCD cannot be used, and the description of the partons' behaviour at the scale depends on phenomenological models. In the context of this thesis, the phenomenological model used is the Lund string model [23–25]. After the parton showering described above, the resulting partons do not form a color-neutral state. However, the principle of colour confinement prevents this. The strong interaction potential can be characterised by its potential energy and visualised as a gluons' tube binding the quarks into a colour-neutral state. This potential energy is defined as $V(d) = kd$ where d is the distance between two partons and k the parameter characterising the strong force confinement, of order of magnitude 1 GeV/fm. As the quarks move apart, the tube lengthens and the gluon field acquires more energy. When the energy becomes large enough to form a new quark-antiquark pair, the tube can 'break' and create the pair, as illustrated in Figure 1.2. This process leads to the creation of new pairs and potential energy tubes binding them together. The phenomenon continues until the quarks' energy regime becomes low enough that the production of new pairs is no longer favoured, and the potential energy tubes eventually bind all quarks into colour-neutral states. This final stage is called hadronisation.

Therefore, a quark or gluon in the final state will never be observed alone and isolated. Instead, we will observe the final state of the production cascade caused by colour confinement, which results in a narrow cone of hadrons, possibly also containing leptons, that are detectable by our detector subsystems. This cone, originating from a quark or gluon, is called a jet and will be the primary subject of study discussed in this thesis, whether in terms of its reconstruction by the detector or, more importantly, how we can algorithmically identify certain essential properties, such as the quark or gluon of origin or its energy.

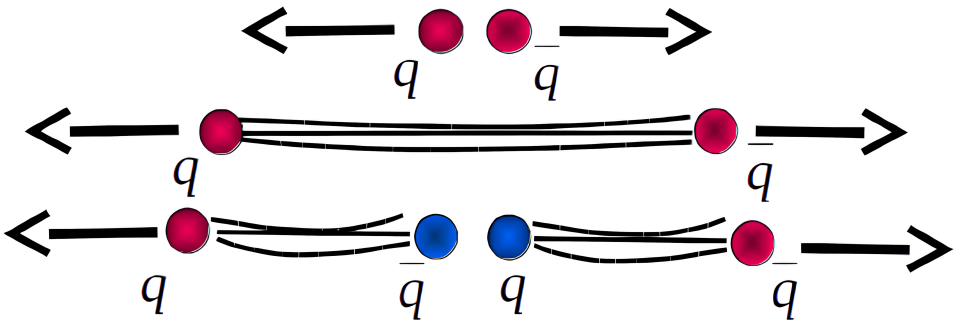

Figure 1.2: Illustration of the colour confinement. When quarks separate, the potential energy in the string increases linearly with the separation. When the stored energy exceeds $2 \times m_q$, a new $q\bar{q}$ pair can be created [26].

The formulation of the Standard Model is carried out using QFT. The formalism of QFT aims to represent the dynamics of particles and their interactions via their quantum fields, denoted $\phi(x)$, in the space-time coordinates x. The behaviour of these particles can be determined from the action S by applying the principle of least action:

$$S = \int \mathrm{d}^4x \; \mathcal{L}_{\mathrm{SM}}(\psi(x), \partial_\mu \psi(x)) \tag{1.2}$$

where the Lagrangian density of the Standard Model is represented by $\mathcal{L}_{\mathrm{SM}}$. Lagrangian density from here on will be mentioned as Lagrangian. To begin describing and formulating the Lagrangian of the Standard Model, let us begin by examining some of the fundamental properties governing QFT, starting with the Lagrangian of a free fermion defined by the spinor ϕ and mass m:

$$\mathcal{L} = \bar{\psi}(x)(i\gamma^\mu \partial_\mu - m)\psi(x) \tag{1.3}$$

with the Dirac adjoint $\bar{\psi}(x) = \psi^\dagger(x)\gamma^0$, where γ^μ are the Dirac matrices, and ∂_μ is the partial derivative with respect to the space-time coordinates.

One of the most important theorems in QFT is Noether's theorem on global symmetries [27]. This theorem states that if a La-

grangian is invariant under a global symmetry, then there exists an associated conserved current and charge. To illustrate this phenomenon, we can observe that the Lagrangian of a free fermion possesses a global $U(1)$ symmetry defined by an arbitrary phase θ of the form: $\psi(x) \to \psi'(x) = e^{-i\theta}\psi(x)$. In this case, the conserved current and charge will be the electromagnetic current and the electric charge, leading to the formulation of QED. To reconstruct the interaction term of QED while maintaining the global gauge symmetry, the system's symmetry must be extended to include a local symmetry. This extension arises from gauge theory principles and ensures the consistency of the theory under local transformations. Thus, we impose invariance under a local $U(1)$ gauge transformation, defined by $\psi(x) \to \psi'(x) = e^{-i\theta(x)}\psi(x)$. The Dirac adjoint also transforms as $\bar{\psi}(x) \to \bar{\psi}'(x) = e^{i\theta(x)}\bar{\psi}(x)$, allowing us to observe how the Lagrangian in Equation (1.3) changes under a local $U(1)$ symmetry:

$$\begin{aligned}\mathcal{L} &= e^{i\theta(x)}\bar{\psi}(x)(i\gamma^\mu\partial_\mu - m)e^{-i\theta(x)}\psi(x) \\ &= \bar{\psi}(x)i\gamma^\mu\partial_\mu\psi(x) + \bar{\psi}(x)(-(i)^2\gamma^\mu\partial_\mu\theta(x))\psi(x) - \bar{\psi}(x)m\psi(x) \\ &= \bar{\psi}(x)(i\gamma^\mu\partial_\mu + \gamma^\mu\partial_\mu\theta(x) - m)\psi(x)\end{aligned} \tag{1.4}$$

Under the effect of the local $U(1)$ symmetry, an additional term is introduced into our Lagrangian: $\bar{\psi}(x)(\gamma^\mu\partial_\mu\theta(x))\psi(x)$. We can now apply the gauge principle to introduce the gauge field associated with our symmetry, allowing us to cancel out this additional term for local transformations. In the context of the $U(1)$ symmetry, this additional gauge field $A_\mu(x)$ transforms under the symmetry as: $A_\mu(x) \to A'_\mu(x) = A_\mu(x) - \partial_\mu\theta(x)$. In the framework of QED, this gauge field corresponds to the gauge boson and mediator of the electromagnetic interaction, the photon. By adding this gauge field to our Lagrangian, we obtain the formulation that is invariant under the local $U(1)$ symmetry:

$$\begin{aligned}\mathcal{L} &= \bar{\psi}(x)(i\gamma^\mu\partial_\mu + \gamma^\mu A_\mu(x) - m)\psi(x) \\ &= \bar{\psi}(x)(i\gamma^\mu D_\mu - m)\psi(x)\end{aligned} \tag{1.5}$$

where we have introduced the covariant derivative $D_\mu = \partial_\mu - iA_\mu(x)$. Finally, we add the kinetic term of our gauge field. This term must also be gauge-invariant. In the context of the $U(1)$ symmetry, the kinetic term is constructed from the field strength tensor $F_{\mu\nu} = \partial_\mu A_\nu(x) - \partial_\nu A_\mu(x)$:

$$\mathcal{L} = \bar{\psi}(x)\left(i\gamma^\mu D_\mu - m\right)\psi(x) - \frac{1}{4}F_{\mu\nu}F^{\mu\nu} \tag{1.6}$$

With this final term, we have completed the Lagrangian of QED under the gauge theory applied to the $U(1)$ symmetry. All the gauge bosons of the Standard Model are obtained through the same principle. In the context of the Standard Model, the Lagrangian is derived from the group symmetry: $SU(3)_C \times SU(2)_L \times U(1)_\Upsilon$. The $SU(3)_C$ symmetry and its Lie algebra generate the strong interaction and its gauge bosons, the gluons. The electroweak interaction is described by the $SU(2)_L \times U(1)_\Upsilon$ symmetry. This symmetry will, at a low-energy regime, break to form the weak interaction with its gauge bosons $W^\pm$ and Z, as well as electromagnetism with its gauge boson, the photon.

The Brout-Englert-Higgs mechanism

The last component of the Standard Model Lagrangian is the Brout-Englert-Higgs mechanism, which explains the origin of the mass of the massive gauge boson and fermions as well as the origin of the Higgs boson. This scalar boson, whose theoretical existence was proposed in 1964 by Robert Brout, François Englert, and Peter Higgs [3,4], was the last component of the Standard Model to be discovered in 2012 at the LHC [1, 2]. With a mass of approximately 125 GeV, its existence was predicted to solve the problem of the mass term for gauge bosons and fermions in the Standard Model Lagrangian under the gauge invariance of the $SU(3)_C \times SU(2)_L \times U(1)_\Upsilon$ symmetry.

By introducing a scalar doublet ϕ endowed with a potential $V(\phi) = -\mu^2\phi^\dagger\phi + \frac{1}{2}\lambda(\phi^\dagger\phi)^2$, the potential is minimised under the relation $\phi^\dagger\phi = \frac{\mu^2}{\lambda}$. Thus, unlike other fields in the Standard Model, the value of the Higgs boson field ϕ that minimises its potential, illustrated in Figure 1.3, is not null and has a value of $v \approx 246$ GeV,

also known as the vacuum expectation value.

Through spontaneous symmetry breaking, three Nambu-Goldstone bosons are created from the three generators of the symmetry that were broken. These bosons, associating with the weak isospins $SU(2)_L$ gauge bosons W^1_μ, W^2_μ, W^3_μ, and the hypercharge $U(1)_\Upsilon$ gauge boson B_μ, allow us to obtain the gauge bosons of the weak interaction and electromagnetism. To do this, they must first be represented in their mass eigenstates using the mixing angle: the electroweak Weinberg angle $\theta_{\text{EW}} = \arctan\left(\frac{g_\Upsilon}{g_W}\right)$, where the weak isospin and hypercharge coupling strength are noted as g_W and g_Υ, respectively. We can then represent the gauge bosons in their mass eigenstates:

$$\begin{aligned} A_\mu &= \sin\theta_{\text{EW}} W^3_\mu + \cos\theta_{\text{EW}} B_\mu \\ Z_\mu &= \cos\theta_{\text{EW}} W^3_\mu - \sin\theta_{\text{EW}} B_\mu \\ W^\pm_\mu &= \sqrt{1/2}\left(W^1_\mu \mp i W^2_\mu\right) \end{aligned} \tag{1.7}$$

Thus, we can reconstruct the gauge bosons and their masses acquired through the Brout-Englert-Higgs mechanism. Furthermore, fermions acquire their masses through this mechanism via the Yukawa coupling: $-\lambda_\psi \bar{\psi}\phi\psi$, where the Yukawa coupling strength λ_ψ can be developed into $\lambda_\psi = \frac{\sqrt{2}}{v} M_\psi$ where M_ψ is the mass of the fermion ψ.

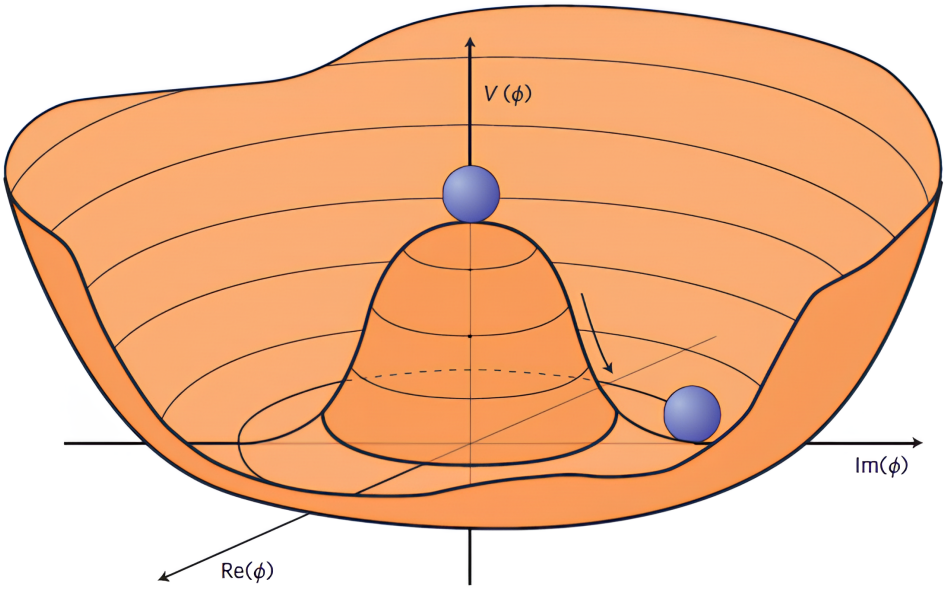

Figure 1.3: Illustration of the Higgs boson potential [28].

1.2 The Large Hadron Collider

The Large Hadron Collider (LHC) [29] is a particle collider located at the European Organization for Nuclear Research (CERN) in Geneva. It is a circular collider built underground, with a circumference of 26.7 km, housed in the same tunnel as its predecessor, the Large Electron Positron Collider (LEP) [30], making them the two largest colliders ever built. The LHC primarily operates proton-proton (pp) collisions but can also accelerate heavy ions for lead-proton or lead-lead collisions. In the context of this thesis, we will focus solely on the main type of collision operated by the LHC, namely pp collisions. The LHC has been operational since 2010, and its operations can be distinguished into three Runs, each separated by a long shutdown, during which upgrades to the collider and detectors are implemented. Run 1 of pp collisions occurred between 2010 and 2012, with a center-of-mass energy of 7 TeV for 2010-2011 and 8 TeV for 2012. Run 2 of pp collisions occurred between 2015 and 2018 with a center-of-mass energy of 13 TeV. Finally, Run 3 of pp collisions began in 2022 and will continue until the end of 2025, with a center-of-mass energy of

13.6 TeV.

The LHC benefits from its implementation within the complex of accelerators present at CERN to accelerate the proton beams that the LHC will use, as outlined in Figure 1.4. Protons are obtained from gaseous hydrogen subjected to an electromagnetic field, which separates the proton from its electron. The proton beam at the LHC consists of protons bunched together. This allows interactions between the two beams to occur at discrete intervals of approximately 25 ns. The resulting proton bunches are guided by an electric field to the first injector, LINAC4 [31], which replaced the previously used LINAC2. LINAC4 employs radiofrequency (RF) cavities to accelerate the protons by producing an oscillating magnetic field at 352.2 MHz. These RF cavities, synchronised with the incoming protons, accelerate them until they reach the desired energy of 160 MeV for LINAC4. By synchronization with the protons' arrival, the RF cavities ensure that protons at the correct energy no longer experience any further acceleration. After being accelerated by LINAC4, the proton bunches continue their injection into circular accelerators, first, the Proton Synchrotron Booster (PSB), where they reach an energy of 2 GeV. The Proton Synchrotron (PS) will then increase the proton beam energy to 26 GeV before injecting the beam in the Super Proton Synchrotron (SPS), which will accelerate the beam energy up to 450 GeV. After this series of injection accelerators, the proton beams are injected into the LHC.

Once injected into one of the two LHC trajectories on which the protons are accelerated in opposite directions, the particle beams continue to accelerate until they reach an energy of 6.8 TeV, or a collision energy of 13.6 TeV in the centre-of-mass when the beams intersect. To bend the particles on circular trajectories, dipole magnets are used to curve the trajectory in line with the curvature of the pipeline. To bend the highly energetic proton beams of the LHC, 1232 dipoles operating at a magnetic field of 8.3 T are positioned along the length of the accelerator and are maintained at a temperature of approximately 2 K to function via superconductivity. Additionally, 392 quadrupoles focus the proton bunches to ensure they

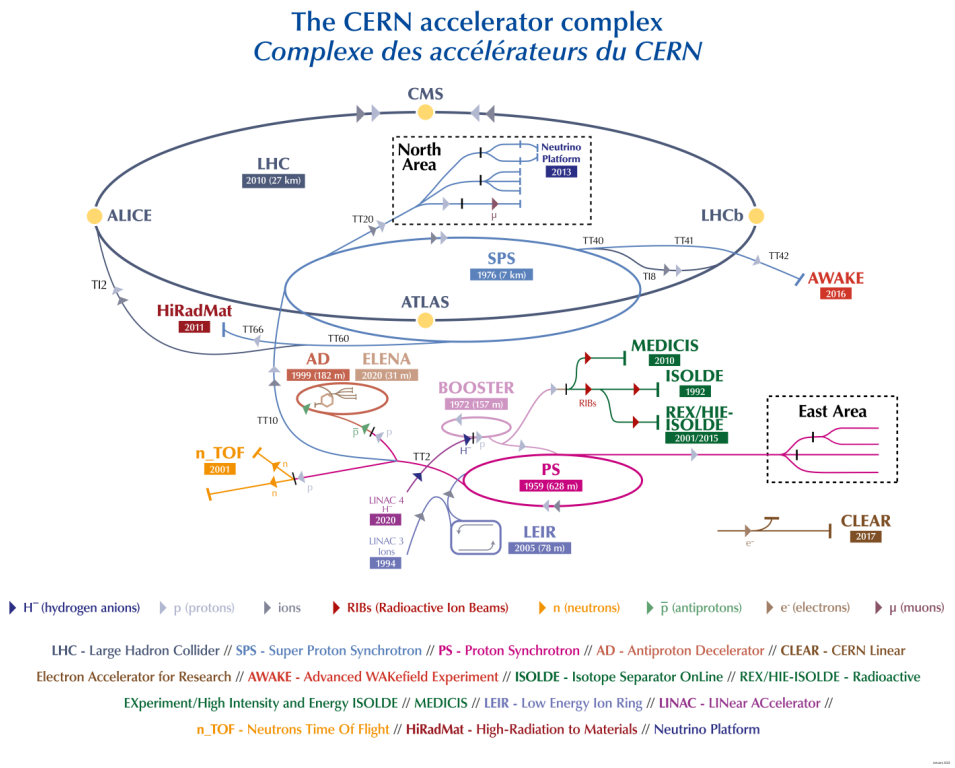

Figure 1.4: Schematics of the CERN accelerators complex illustrating the LHC and the injection accelerators [32].

are sufficiently compact at the crossing points where the two proton beams merge. Around these crossing points, of which there are four in the LHC, collisions occur and are recorded by four experiments: ALICE, LHCb, ATLAS, and CMS, each designed to measure the results of these collisions. To ensure that the experiments collect enough data, the LHC aims to optimise the number of potential collisions per unit of time and area, also known as luminosity. For pp collisions, the Equation (1.8) defines the instantaneous luminosity delivered by the LHC during operation.

$$L = \frac{N^2 n_{\mathrm{b}} f}{4\pi\sigma_x\sigma_y} \tag{1.8}$$

where N is the number of protons per bunch, approximately

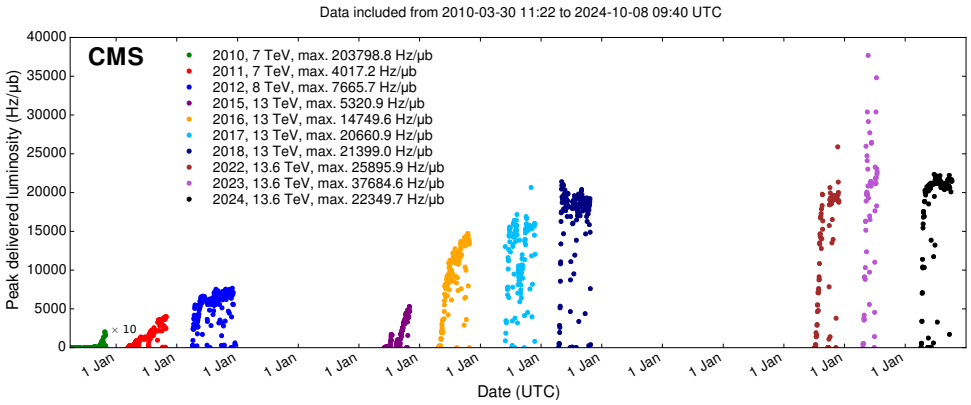

Figure 1.5: Peak luminosity versus day delivered to CMS during stable beams and for pp collisions [33]. The luminosity is show for Run 1, Run 2 and Run 3. The data-taking is separated per year: 2010 (green), 2011 (red), 2012 (blue), 2015 (purple), 2016 (orange), 2017 (light blue), 2018 (navy blue), 2022 (brown), and 2023 (light purple).

10^{11} in the LHC, and n_{b} is the number of bunches in each beam, around 2800 for the LHC. The factor f is the bunch crossing frequency at the interaction point, which is 40 MHz at the LHC, and $\sigma_x\sigma_y$ represents the bunch width in the plane transverse to the crossing axis. The LHC is capable of delivering an instantaneous luminosity superior to 2×10^{34} cm^{-2}s^{-1} since Run 3 as illustrated in Figure 1.5. The most commonly used unit for determining luminosity is the barn (b), which has a value of $1 \times 10^{-24}\,\mathrm{cm}^{-2}$. The instantaneous luminosity is defined in units of Hz/μb. The acquired luminosity is most often integrated over the data-taking time, as illustrated in Figure 1.6 for the CMS 2022-2024 pp data taking, resulting in a variable called the integrated luminosity, L_{int}:

$$L_{\mathrm{int}} = \int L \, \mathrm{d}t \tag{1.9}$$

The integrated luminosity recorded by the CMS detector during the entirety of Run 2 is 150.78 fb^{-1} [33]. The integrated luminosity recorded by the CMS detector during Run 3 up to October 8, 2024, amounts to 175.57 fb^{-1} [33].

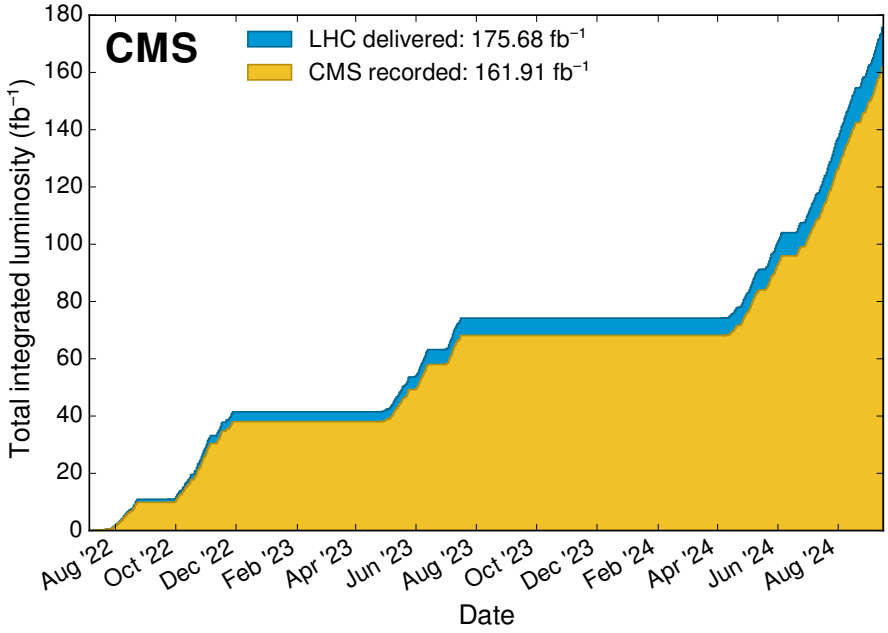

Figure 1.6: Cumulative delivered and recorded luminosity versus time for 2022-2024 [33] (pp data only). The integrated luminosity is illustrated in blue and the recorded.

1.3 The Compact Muon Solenoid experiment

Among the four experiments installed around the interaction points of the LHC is the Compact Muon Solenoid (CMS) experiment [34]. Located at the LHC point 5, in the commune of Cessy in France, the CMS experiment is one of the two general-purpose detectors, along with the ATLAS experiment [35]. The detector is 21 m long, 15 m of diameter, and weighs 14,000 tonnes.

One of the main components of the CMS detector is its solenoid magnet, which bends charged particles with its powerful 3.8 T magnetic field. The innermost part of the CMS experiment is the tracker, installed around the beam pipe, followed by the electromagnetic and hadronic calorimeters, respectively. The tracker and the electromagnetic and hadronic calorimeters of the CMS detector are embedded in the solenoid magnet, which is one of the main design differences

between the CMS and ATLAS experiments. This design difference allows the CMS experiment to be more compact. Another characteristic of the CMS detector are the muon chambers, located outside the solenoid magnet, which enable the detector to achieve excellent reconstruction and identification of produced muons.

The CMS coordinate system, illustrated in Figure 1.7, defines the z-axis along the beam pipe, the x-axis pointing towards the centre of the LHC ring, and the y-axis pointing towards the surface. The transverse plane is one of the key elements, where the azimuthal angle ϕ describes the opening in this plane from the x-axis. The polar angle θ defines the angle between the transverse plane and the z-axis, with the opening starting from the z-axis. However, the pseudorapidity η is more commonly used and is defined via θ or the momentum $p = (p_x, p_y, p_z)$:

$$\eta = -\ln\left(\tan\frac{\theta}{2}\right) = \frac{1}{2}\ln\left(\frac{|p| + p_z}{|p| - p_z}\right) \tag{1.10}$$

We will use η to describe the trajectory of a particle relative to the z-axis. If $\eta = 0$, the particle has a trajectory perpendicular to the beam axis and moves in the transverse plane. In contrast, as η increases, the particle approaches the beam axis, with $\eta = \infty$ corresponding to a trajectory parallel to the z-axis. Note that in the limit of ultra-relativistic particles, pseudorapidity is equivalent to rapidity $y = \frac{1}{2}\ln\left(\frac{E+p_z}{E-p_z}\right)$. Furthermore, unlike $\Delta\theta$ intervals, the Δy interval is invariant under boosts along the z-axis, and the $\Delta\eta$ is invariant under the same boosts for massless particles.

The z-axis of our reference frame is aligned with the beam axes, and due to momentum conservation, the sum of the momenta of the particles produced in the transverse plane should be zero if we are able to detect each one of them. Thus, another variable used in collider physics is the transverse momentum, denoted p_{T}:

$$p_{\mathrm{T}} = \sqrt{p_x^2 + p_y^2} = p\sin\theta \tag{1.11}$$

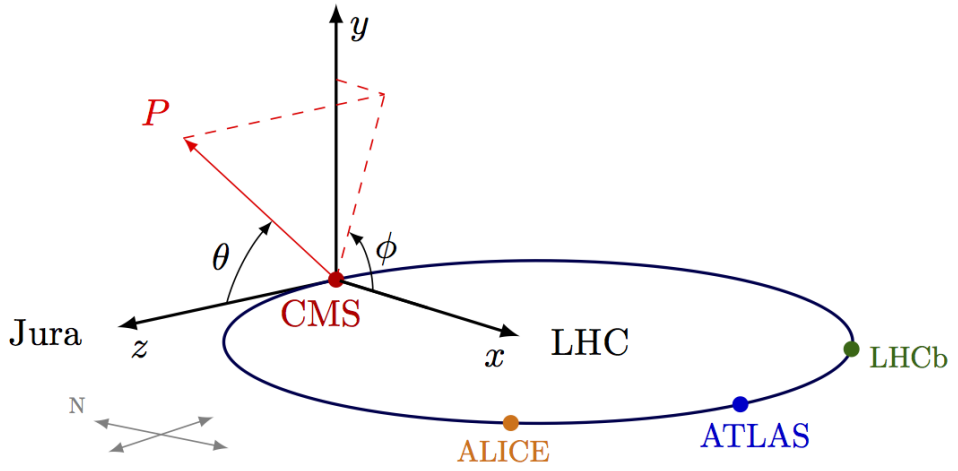

Figure 1.7: The CMS coordinate system [36].

1.3.1 The Silicon Tracker

The CMS silicon tracker [37–39] is the detector component closest to the beam pipe. It aims to track the paths of charged particles produced in collisions. A charged particle with charge q and momentum p in the magnetic field B generated by the CMS solenoid will follow a curved path with a radius of curvature $r = \frac{p}{Bq}$, allowing the measurement of the momentum via a measurement of the radius of the track. The particle trajectories also enable vertexing, which aims to identify the primary collision point. Beyond this primary vertexing, secondary vertexing can also be performed to reconstruct the vertex of certain particles whose lifetime leads to creating a secondary vertex displaced from the primary collision vertex. The silicon tracker illustrated schematically in Figure 1.8 consists of a barrel and two endcaps, providing coverage up to $|\eta| < 2.5$.

In its current configuration, the CMS silicon tracker has been upgraded with an upgraded pixel system [39, 41] installed during the year-end technical stop of the LHC in 2016/2017. This upgrade, also referred as the Phase-1 upgrade is illustrated in in Figure 1.9. Since then, the tracker barrel, with a radius of 1.2 m and a length of 5.6 m, is composed of four pixel layers, with the innermost layer located 29 mm from the beam pipe [39]. These pixel layers consist of silicon

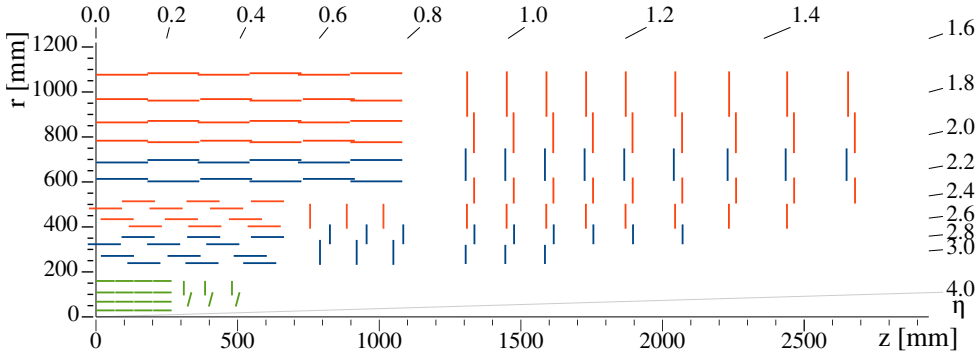

Figure 1.8: Schematic view of a quarter of the Phase-1 CMS tracking system [40]. The pixel modules are shown in green, while the strip modules are depicted in red and blue.

sensors, with pixel sizes of 150 × 100 μm, providing high granularity and ensuring a transverse hit resolution of 10 μm and a longitudinal resolution of 20-40 μm [39]. In addition to these four barrel layers, each endcap has two pixel layers. In total, the pixel detector consists of approximately 120 million pixels, covering an active tracking area of 1.2 m^2. Located beyond the pixel layers are the silicon strip sensor layers, which have lower granularity and offer a resolution of around 20-40 μm [42]. The barrel has ten layers of silicon strips, while the endcaps have twelve layers. Approximately 9.8 million strips make up the tracker, with the active area of the silicon strip detectors covering approximately 198 m^2, larger than the active area of the inner pixel detector. The CMS tracker is thus capable of measuring the momentum of charged particles with a resolution of about 1-2%, while primary vertices can be reconstructed with a resolution of about 10 μm in each of the three spatial dimensions [43].

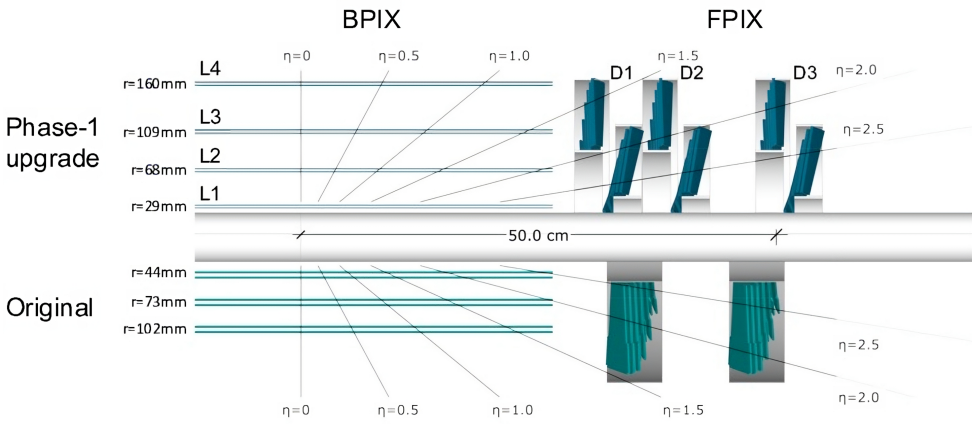

Figure 1.9: Schematic view of the pixel detector with the current Phase-1 upgrade configuration [39].

1.3.2 The Electromagnetic Calorimeter

Beyond the CMS silicon tracker, the particles cross the Electromagnetic Calorimeter (ECAL) of the CMS experiment [44, 45]. The ECAL is designed to measure the energy of particles that interact primarily electromagnetically with the medium, namely the electrons/positrons and photons produced in the collisions. The CMS ECAL is a homogeneous calorimeter composed of lead tungstate ($PbWO_4$) crystals with a radiation length of approximately 0.89 cm. Thus, the crystals require only a length of about $\mathcal{O}(20\text{ cm})$ to contain the entire energy deposition of a high-energy photon or electron, allowing for a compact design that fits within the solenoid magnet. The Molière radius of lead tungstate is 2.2 cm, leading to the trapezoidal shape of the crystal faces. The calorimeter is divided into a barrel covering $|\eta| < 1.479$ and two endcaps covering $1.479 < |\eta| < 3.0$, illustrated in Figure 1.10, and consists of 75,848 crystals in total. These crystals have a length of 22 to 23 cm, with a surface area varying from $2.2 \times 2.2\text{ cm}^2$ to $2.9 \times 2.9\text{ cm}^2$ for the side facing inward in the barrel and endcaps, respectively. The outer surface has dimensions of $2.6 \times 2.6\text{ cm}^2$ in the barrel and $3.0 \times 3.0\text{ cm}^2$ in the endcaps. Finally, preshower detectors are installed in the region $1.536 < |\eta| < 2.6$, in front of the previously described device to separate high-energy single

photons from photon pairs with small angular separation, primarily originating from the decay of a neutral pion π^0. The energy resolution of the CMS ECAL is about 2% in the barrel and 3-5% in the endcaps [46].

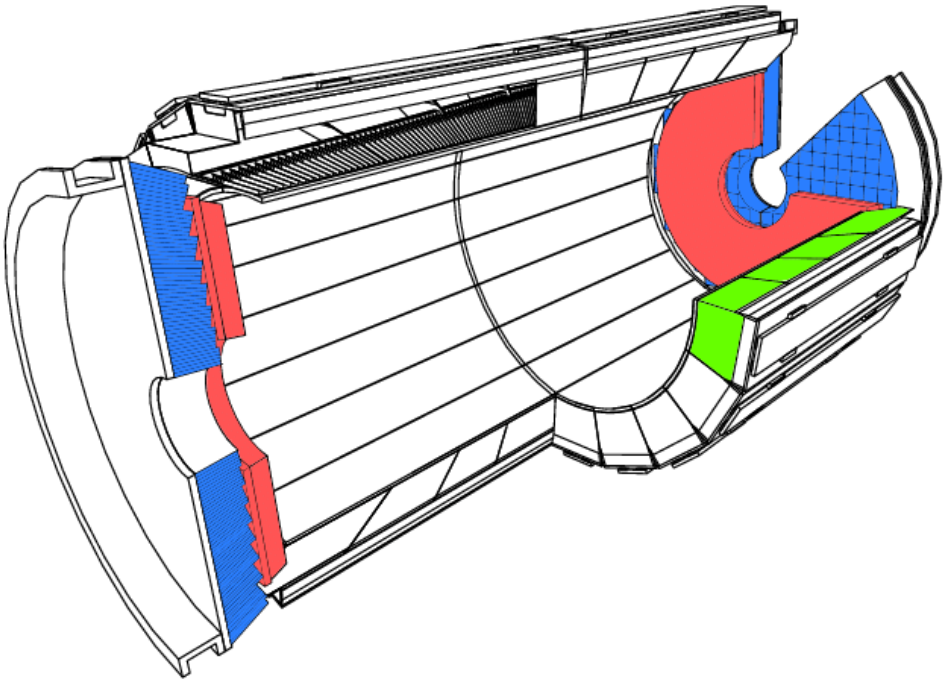

Figure 1.10: The schematics of the CMS ECAL [47]. An ECAL barrel module is highlighted in green, the two ECAL endcaps in blue and the two preshowers in red.

1.3.3 The Hadronic Calorimeter

The CMS Hadronic Calorimeter (HCAL) [48–50] is designed to measure the energy of neutral and charged hadrons produced during collisions. The HCAL is divided into several components: two endcaps (HE) and a barrel (HB) placed inside the magnet coil. Additionally, two further components, the forward calorimeter (HF) and an outer calorimeter (HO), complete the system. Unlike the ECAL, the CMS HCAL is a heterogeneous sampling calorimeter composed of alternating layers of absorbers and active media. The HB has a

coverage region of $|\eta| < 1.3$ and is made of 5 cm brass layers as absorbers. The active medium is composed of plastic scintillator tiles. In total, the HB consists of 17 layers for a total of 5.8 nuclear interaction lengths. The HE uses 8 cm brass layers to cover the region $1.3 < |\eta| < 3.0$. The HE consists of 19 layers, providing 10 nuclear interaction lengths. Together, the HB and HE contain 2304 modules. The HO employs the magnet coil as the absorber and plastic scintillator layers as the active medium. The final component, the HF scintillator, is located 11 m away, covering $2.9 < |\eta| < 5.0$ and is a Cherenkov calorimeter. It uses steel as the absorber material and quartz fibres to collect Cherenkov radiation signals. The energy resolution depends on pseudorapidity. As a reference, the energy resolution for a pion in the barrel is approximately $\frac{120\%}{\sqrt{E}}$ [50].

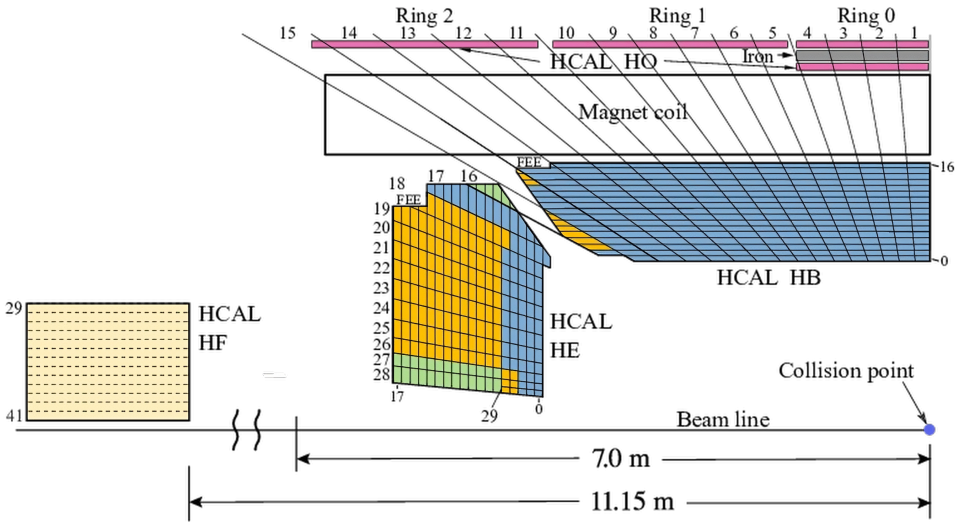

Figure 1.11: Quadrant schematics of the CMS HCAL [50]. The four components, HB, HE, HO, and HF HCAL modules, are labelled in the figure.

1.3.4 The Muon Chambers

Beyond the solenoid magnet of the CMS experiment lies the muon system [51, 52], which enables the identification and reconstruction of the produced muons. Only muons and neutrinos typically pass through the CMS solenoid magnet because they interact weakly with matter. However, neutrinos interact so weakly that they cannot be identified or reconstructed by the detector, leaving only muons to interact in the muon system. Figure 1.12 illustrates a schematic of the muon system.

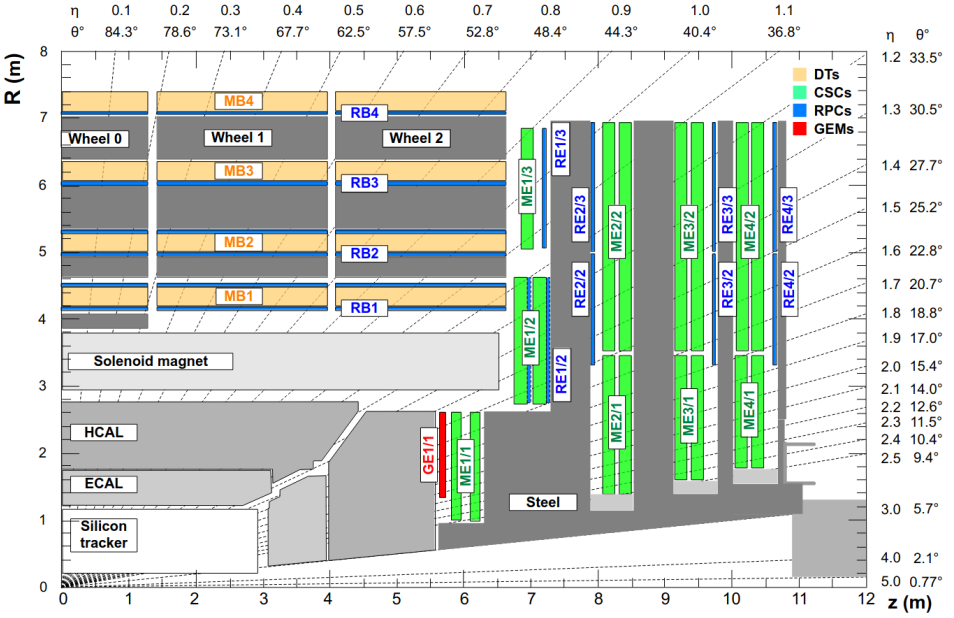

Figure 1.12: Quadrant of the CMS experiment highlighting in beige the DTs, in green the CSCs, in blue the RPCs and in red the GEMs [53].

This system is divided into two parts: the section covering the barrel, with pseudorapidity $|\eta| < 1.2$, and the two endcaps, which cover pseudorapidity values of $0.9 < |\eta| < 2.4$. Detection in the barrel uses drift tubes (DTs), while the endcaps use cathode strip chambers (CSCs), better suited to the radiation and data acquisition constraints in this region of the detector. In addition to these components, resistive plate chambers (RPCs) are placed throughout the barrel and endcaps to improve the temporal resolution of the muon

system. Moreover, gaseous electron multiplier (GEM) detectors were added at the end of Run 2 in the region closest to the centre, covering $1.6 < |\eta| < 2.2$ [53]. Thanks to its muon system, the CMS experiment has a muon reconstruction efficiency of approximately 95-99%, and the resolution on their transverse momentum varies between around 2% in the barrel and less than 10% in the endcaps.

1.3.5 The Trigger System

During data taking, the LHC produces bunch crossings at a frequency of 40 MHz, and in each crossing, several pp collision occur. For the 2023 Run 3 data-taking, the mean number of pp interactions per crossing was 52 [33]. However, the majority of interactions resulting from these collisions are inelastic, involving a small momentum exchange. In contrast, the primary interest of the CMS experiment lies in collisions with large momentum exchanges, enabling research into heavy physical objects such as the Higgs boson, Z boson, and top quark. Moreover, considering that the data produced per bunch crossing is around 1 MB and the immense number of pp collisions results in a data production rate of 40 MHz, we would need to process 40 TB/s of data at high granularity. This constraint is several orders of magnitude beyond the bandwidth and hardware capabilities that the data acquisition system can support, making it necessary to apply selections to filter the data. This collision selection/filtering system, which is the focus of this section, is called the trigger system and consists of two levels of selection, illustrated by the schematic view of the Run 2 trigger system in Figure 1.13. Note the Run 3 trigger system follows the same data-taking scenario with increased rates [54].

The first, called the level-1 trigger (L1) [55], consists of hardware systems such as field-programmable gate arrays (FPGAs), which use non-granular data from the calorimeters and muon chambers with a latency of around 3.8 μs and allow the selection of events at a rate of 110 kHz under Run 3 conditions [54] for the next stage. The second stage, called the high-level trigger (HLT) [56], consists of a hardware farm composed of heterogeneous elements,

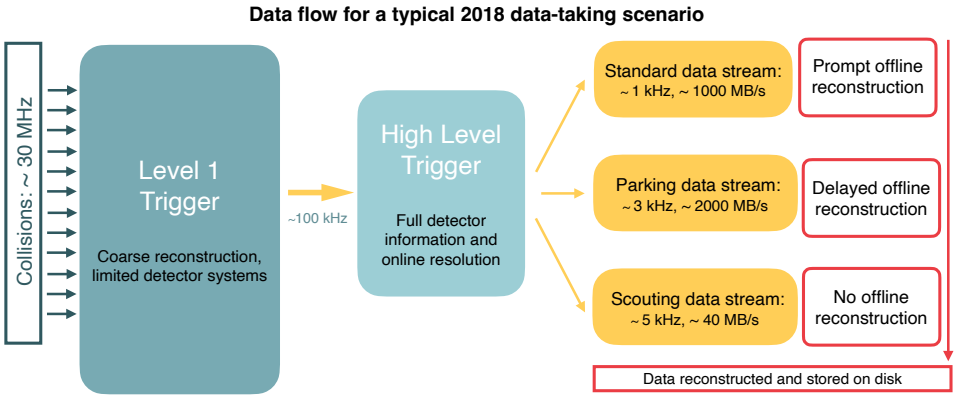

Figure 1.13: Schematic view of the Run 2 data flow for the 2018 data-taking scenario [54].

including CPUs and GPUs, for Run 3 [54]. At this stage, the full granularity of the event is used for processing, with a simplified and faster reconstruction compared to the final level, also known as offline reconstruction, allowing the selection of events at a rate of approximately 7 kHz. Of this rate, 2 kHz will be promptly reconstructed within 48 hours of acquisition, while 3 kHz will be reconstructed later towards the end of the data-taking year using a 'data parking' system [54].

1.4 The reconstruction of the physics objects

1.4.1 Tracking

The tracks left by charged particles in the silicon tracker are one of the primary signatures at the CMS experiment. Tracking in the CMS experiment is performed using a combinatorial method based on Kalman filters, updated for Run 3 to function optimally by exploiting parallelisation and vectorisation of multi-core CPU architectures [57]. This update allows for the same level of performance as the original Combinatorial Kalman Filter (CKF) algorithm [43,58] while improving computational efficiency.

Track fitting starts with initial seeds corresponding to a number

of compatible hits. The fitting algorithm then extrapolates the trajectory to the next layers from the initial seed tracks and attempts to find compatible hits in these layers with the seed tracks. Once a hit is found, the trajectory of the associated seed track is updated by incorporating the new hit. If multiple hits are compatible, the algorithm produces an updated track version for each possible association, and the fitting continues. After adjusting all tracks with all layers, a final fit is performed, accounting for the non-uniformity of the magnetic field B and a better description of the interaction with the tracker material. After fitting, the tracks are classified using a deep neural network developed to assess track quality and reject fakes that may be produced, as well as duplicates that share too many common hits [59]. Once the fitting and the track classification are completed, the algorithm extrapolates the momentum, direction, and origin of each track. The used hits are removed from the available hits, and a new fit with different seeding conditions can be performed. These steps are repeated with various seeding conditions to accurately fit tracks under different conditions. Twelve iterations are performed, starting with pixel quadruplets to fit high p_T tracks, and the final two procedures take into account the muon system to specifically reconstruct this type of particle.

1.4.2 Vertexing at CMS

The number of pp collisions per bunch crossing, in the Run 3 condition, is around 52 [33], leading to numerous low-energy collisions that are not part of the physics and interactions we wish to study. These secondary collisions, also known as pileup, contaminate the reconstruction of our event and require several treatments. The first step is reconstructing the interaction point of each pp collision, also known as the primary vertex (PV). To reconstruct these, PV candidates are determined via a clustering algorithm, the deterministic annealing algorithm, using the longitudinal coordinate of the point of closest approach of each track to the beam spot [60]. After identifying the potential PVs, they are fitted using the Adaptive Vertex Fitter (AVF) algorithm [60].

In addition to vertexing PVs, CMS employs an algorithm to reconstruct the vertices from the decay of particles with a sufficiently long lifetime to decay after travelling a certain distance. This step allows for the reconstruction of what we call Secondary Vertices (SVs), a critical component of heavy-flavour jets, as these contain heavy hadrons that will decay after travelling a distance of approximately ~ 0.5 mm, sufficient to reconstruct a secondary vertex. The algorithm used for secondary vertex reconstruction at CMS is the Inclusive Vertex Finder (IVF) algorithm [61]. The IVF algorithm employs all the tracks to find significantly displaced seed tracks. Other tracks are associated with the seeds based on their angular difference and distance of closest approach extrapolated by their trajectory, with certain quality cuts applied. During this association, the IVF algorithm also ensures that tracks associated with the potential SV are closer to the SV candidate than any PV candidate. Finally, the tracks associated with each SV seed are fitted to determine the vertex position, its distance from the PV, and the kinematic properties of the reconstructed vertex.

1.4.3 The Particle-Flow algorithm

In the CMS experiment, the Particle-Flow (PF) algorithm is a particle reconstruction method that combines information from different subdetectors to identify and measure the individual particles produced during collisions. Depending on the nature of the particles, their interactions with the various subdetectors differ, as illustrated in Figure 1.14. Photons, being neutral, leave no trace in the tracker but deposit their energy in the ECAL, where they are detected. Electrons/positrons, on the other hand, are charged and leave signatures in the tracker and the ECAL. Hadrons, originating from the hadronisation of quarks, can also be classified as neutral or charged. Neutral hadrons leave no trace in the tracker and deposit most of their energy in the HCAL. Charged hadrons, in contrast, will leave a trace in the tracker before undergoing the same energy deposition process in the HCAL, with possibly some energy initially deposited in the ECAL. Finally, being charged and weakly interacting, muons leave

tracks in the tracker and the muon system. Thus, we observe that the nature of the particles defines distinct signatures, which the PF algorithm leverages to reconstruct particles more efficiently than the reconstruction provided by each individual subdetector.

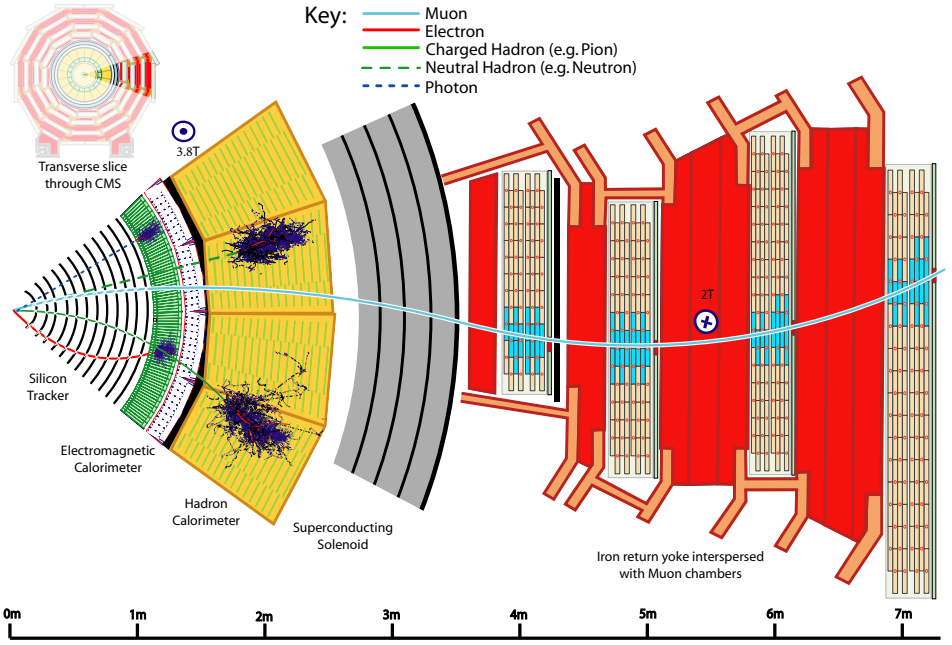

Figure 1.14: Schematic view of the Run 2 data flow for the 2018 data-taking scenario [54].

From the reconstructed tracks and calorimeter clusters of the CMS detector, also called PF elements within the PF algorithm, particle candidates can be reconstructed by linking information from different subdetectors. The algorithm associates the calorimeters and the tracker by extrapolating the track trajectory from the outermost hit while accounting for interactions with the traversed material. If an ECAL or HCAL cluster is compatible with the uncertainties in the η, ϕ plane, they are considered linked. If multiple links exist between several tracks and the same clusters, or vice versa, the separation in η, ϕ is used to decide which final association will be considered. An additional link with potential clusters produced by bremsstrahlung radiation is made for electrons. Similarly, the link between ECAL and HCAL clusters is also established based on their positions in the

η, ϕ plane. As before, if multiple links are made, an arbitration is performed, and the link with the smallest distance in the η, ϕ plane is associated. The linked PF elements thus form PF blocks, which are then processed to reconstruct the final particles. The algorithm starts by reconstructing muons, which are the easiest to identify due to the muon system. Then, isolated electrons and photons are processed. Finally, charged hadrons are processed, followed by neutral and non-isolated photons. It should be noted that at each step, the PF elements and PF blocks used for identifying and reconstructing particles of a certain type are removed from the collection used for the next step.

The PF algorithm performs the reconstruction and identification of muons using PF blocks reconstructed from tracks originating from one of the previously mentioned track muon seeding iterations. For a well-isolated muon, such as one originating from the decay of a Z or W boson, identification is done by checking the total transverse momentum contained in the tracks and the energy deposited in the calorimeter clusters within a radius of $\Delta R = 0.3$ [62]. A correction factor accounting for pileup is also added. If the measured quantity is less than 10% of that of the isolated muon candidate, corresponding to a reconstruction efficiency of 98%, the PF block will be considered a muon. For non-isolated muons, which may originate from the leptonic decay of heavy hadrons in a jet from a b or c quark, such isolation criteria cannot be applied because other particles will surround the low transverse momentum muon. Further procedures based on the granularity and resolution of the tracker, muon chambers, and calorimeters [62] exist.

Electrons and isolated photons are processed simultaneously. Indeed, electrons radiate photons through bremsstrahlung, and photons can convert into an e^+e^- pair, making their identification similar. First, the reconstruction of tracks from electrons is performed using a different tracking algorithm from the Kalman filter, called the Gaussian-sum filter (GSF) [63], which accounts for the energy loss of the electron in the tracker. PF blocks containing a GSF track are labelled as electron candidates. The energy of these electron candidates is calculated from the momentum of the GSF track and the

associated ECAL cluster. Similarly, a PF block with no tracks but only a deposit in the ECAL is considered a photon candidate. A check is then performed in the HCAL to ensure no significant associated deposit is present, as photons and electrons are expected to deposit all or most of their energy in the ECAL. Finally, a final selection using various quality criteria is applied to the electron candidates, using a boosted decision tree algorithm, to improve the efficiency of their selection.

The final reconstruction step involves the reconstruction of charged and neutral hadrons, which constitute most of the particles produced in a pp collision, as well as non-isolated photons. A PF block associated with an HCAL cluster without a track is labelled as a neutral hadron, and its energy is reconstructed from the deposit of the HCAL cluster. PF block containing a track and an HCAL cluster is considered a charged hadron. If an additional deposit is found in the ECAL, a fit between the track's momentum and the energy deposited in the ECAL and HCAL is performed to ensure the correct association of PF elements, and a fit of the energy from the clusters and the track's momentum is then used to obtain a more precise estimate of the charged hadron's energy. If this is not the case, the problematic PF element is removed and reclassified as a neutral hadron or a photon. A PF block consisting of an ECAL cluster without a track is considered a photon, and its energy is determined from the deposit of the respective cluster. If any PF blocks consisting solely of tracks remain, they are reconstructed as charged hadrons. It should be noted that the PF algorithm does not identify the type of charged hadron, and all are reconstructed under the assumption of being a charged pion. Therefore, it is not possible, for example, to differentiate pions $\pi^{\pm}$ from kaons $K^{\pm}$. Finally, outside the tracker acceptance of $|\eta| < 2.5$, no tracks can be reconstructed. Hence, the PF algorithm does not distinguish between neutral and charged hadrons in this region.

Next, all the objects reconstructed by the PF algorithm are gathered into a specific pfCandidate collection within the CMS Software (CMSSW) [64]. A second collection corresponding to the tracks not associated by the PF algorithm, which are stored in the so-called

lostTracks collection if their p_{T} is above 0.95 GeV or they are associated with a secondary vertex or a K_s or Λ^0 reconstructed decay.

1.4.4 Jet reconstruction

As mentioned earlier, quarks produced during collisions will hadronise into a set of colour-neutral particles in a cascade known as a jet, whose classification and the reconstruction of certain variables we will explore in this thesis using Deep Learning algorithms. Thus, jets are the key objects in this thesis.

Among the particles comprising a jet, we find neutral and charged hadrons produced during hadronisation, as well as photons, electrons, and muons, which can, for example, be the product of the decay of certain hadrons such as neutral pions, whose lifetime causes them to decay almost instantly at the PV ($c\tau \sim 2.4 \times 10^{-5}$ mm), primarily into a pair of photons. Figure 1.15 illustrates the formation of a jet during a pp collision and what the CMS detector will be able to observe in comparison with the initial quark alone. During pp collisions, multiple jets are often formed, making it necessary to design an algorithm capable of clustering the reconstructed particles into a jet. Once this is done, corrections are necessary to obtain the best estimate of its properties, such as its energy. These corrections include, for instance, the subtraction of particles associated with the reconstructed jet originating from pileup, as well as corrections accounting for detection inefficiencies and uncertainties.

Jet clustering is handled by the anti-k_t algorithm [66]. It is a jet recombination method that is both infrared and collinear safe (IRC safe). This means that if a low-energy radiation, known as soft radiation, or a collinear splitting from the original quark or gluon, also called a parton, occurs, the jet clustering remains unaffected. The algorithm requires the definition of a radius R, a parameter that defines the jet cone in the (y, ϕ) plane. The choice of radius is an important element that needs to be adjusted to the experimental and pileup conditions. It should also be noted that a larger radius can be chosen to reconstruct more complex jet structures, such as the decay of a boosted particle into two quarks, $X \to q\bar{q}'$. In the context

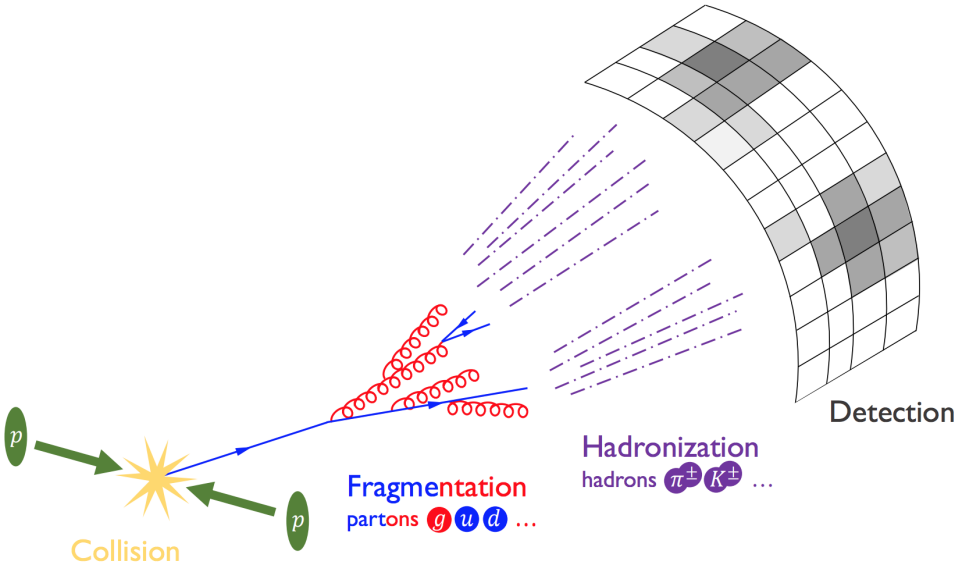

Figure 1.15: Sketch of a jet formation in a pp collision. After the hard process, the quark or gluon produced undergoes fragmentation before hadronising [65]. Note the hadronization product can decay later and induce new particles.

of the CMS experiment, the radius R is typically set to 0.4 for most reconstructed jets. These reconstructed jets are thus referred to as AK4 jets or resolved jets. Based on the choice of radius R, the anti-k_t algorithm defines the following variables for each pair (i, j) of PF candidates:

$$\begin{aligned} d_{ij} &= \min\left(k_{t,i}^{-2}, k_{t,j}^{-2}\right) \frac{\Delta^2}{R^2} \\ d_{iB} &= k_{t,i}^{-2} \end{aligned} \tag{1.12}$$

where $k_{t,i}$ represents the transverse momentum, and Δ is the angular separation given by $\Delta y_{ij}^2 + \Delta\phi_{ij}^2$.

For each pair, if the calculated distance d_{ij} is smaller than d_{iB} and d_{jB}, then the two PF candidates are associated, forming a pseudo-jet, which replaces the two PF candidates in the algorithm, resulting in the final reconstructed jets as illustrated in Figure 1.16. The recombination algorithm continues until no further associations are possible. After obtaining our jets, we still need to manage pileup using a mitigation method. The one employed by the CMS experi-

ment in Run 3 is the pileup per particle identification (PUPPI) algorithm [67]. This technique assigns a weight to each particle in the event based on its likely origin, such as the primary interaction or pileup. The weight indicates how much each particle can contribute to the jet, for example, in measuring its transverse momentum. Therefore, particles originating from pileup are expected to contribute less to the jet properties.

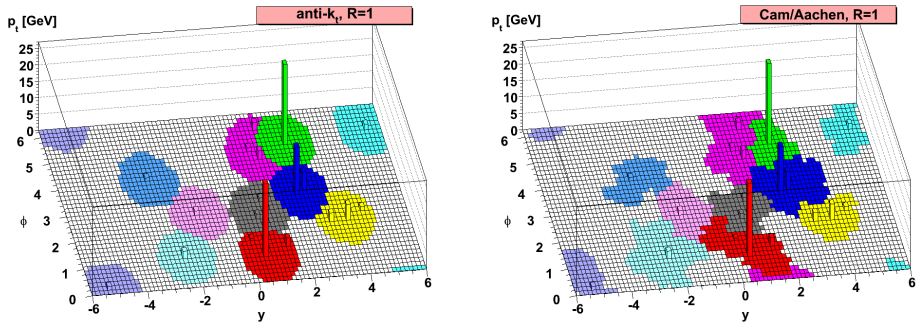

Figure 1.16: Illustration of the anti-k_t clustering algorithm (left) and Cambridge/Aachen clustering algorithm [68,69] (right) using sample parton-level event with 'soft ghosts' [66]. We can see that the anti-k_t algorithm clearly defines conical jet contours, and the shape and boundaries of the jets are not significantly affected by 'soft ghosts,' unlike in the case of the Cambridge/Aachen algorithm.

Jet energy corrections

Once our jet is reconstructed and pileup mitigation applied, several corrections or predictions can be made regarding its properties. The first is the correction of its momentum relative to the generated quark. Multiple factors contribute to the discrepancy between the momentum of the reconstructed jet and the original quark, such as detector response, mismodelling of simulations compared to data, or residual pileup contributions not eliminated by the PUPPI algorithm. The applied corrections, called jet energy corrections (JEC), are provided in several steps [70,71]. The first one aims to correct the remaining pileup contribution after mitigation. This involves deriving the correction from simulated dijet events, with or without background contribution. The next two corrections aim to adjust the de-

tector response by comparing reconstructed jets with generator-level jets matched via ΔR. Generator-level physics objects are described by the generator only. They do not account for the detector interaction and efficiencies. In the context of a generator-level jet, the jet is reconstructed with the anti-k_t algorithm. It consist of the generator-level information of the jet constituents, excluding the pileup contribution. These first three corrections are usually called level 1, level 2, and level 3 corrections (L1, L2, and L3). Finally, corrections are derived between simulations and data using dijet, Z+jets, or γ+jets processes, accounting for differences between data and simulations. This final correction is commonly referred to as residual correction.

Chapter 2

Deep Learning

This chapter is the introductory chapter on the elements of machine learning and mathematics for the thesis. We will discuss the theoretical elements of machine learning and deep learning that allow us to deepen our understanding of the algorithms we will use later. This chapter aims to enable the reader to understand and become familiar with the key elements of deep learning and their origins. Specifically, we will first establish the fundamentals of ML and statistics in Section 2.1. This section gives an overview of the machine learning landscape and the mathematical elements used to evaluate the performance of an algorithm. Then, we will detail the fundamental elements structuring our deep learning algorithms in Section 2.2. We will approach chronologically the different elements of deep learning, which will be addressed in this thesis, as well as their structures and fundamental properties. Finally, we will detail the fundamental principles of artificial neural network training in Section 2.3. We aim to describe how an artificial neural network optimises itself for a given task. More particularly, we will discuss the evolution of training methods from the first primitive neural networks to modern techniques used to train the algorithms mentioned in this thesis. The list of methods described here is not exhaustive of the modern landscape of training methods but covers the elements mainly used in deep learning applied to high-energy physics.

2.1 Machine Learning and fundamentals

Machine Learning (ML) is a subfield of Artificial Intelligence (AI) that encompasses algorithms and statistical methods enabling a system to optimise itself to perform tasks based on data without human intervention during the learning process. More specifically, once the algorithmic logic, objectives, and rules for learning and performance evaluation are established, the machine learning algorithm can autonomously perform statistical learning. This involves predicting values based on the received data, with the goal of minimising the average predictive error as much as possible.

We categorise Machine Learning algorithms according to the training paradigm used. Supervised learning employs labelled data, where each input sample is associated with the values we wish to predict. The objective is to learn a function that maps inputs to outputs, allowing the prediction of labels for new data after the learning phase. For example, image classification is a typical task of supervised learning. Unsupervised learning, on the other hand, uses unlabelled data. The goal is to uncover hidden structures or patterns within the data, such as clustering or dimensionality reduction. A classic example is the k-means clustering algorithm [72]. Semi-supervised learning is an intermediate paradigm: it exploits a small set of labelled data and a large set of unlabelled data to build more robust models. This paradigm aims to enhance the model's performance by leveraging the abundance of unlabelled data while utilising the limited but valuable information from the labelled data to obtain an adequate sample for learning how to identify labels [73]. This approach is particularly useful when data labelling is costly or labour-intensive.

In recent years, ML has effectively solved complex tasks such as computer vision (CV) [74, 75], natural language processing [76–78], autonomous driving [79], and even the discovery of new antibiotics [80]. In high-energy physics, machine learning is employed for numerous tasks such as quark-gluon (QG) jet tagging [81–83], using a variety of ML techniques ranging from likelihood ratios to artificial neural networks. Similarly, Boosted Decision Trees (BDTs), another family of machine learning algorithms, have been used for particle

identification [84], and efficient versions of graph neural networks have enabled the creation of an ML version of the particle-flow algorithm [85]. Machine Learning has contributed not only to the improvement of our tools for describing physical objects but also to numerous analyses, such as enhancing signal sensitivity in searches for Higgs boson production associated with a top quark pair at the CMS experiment [84,86], and providing evidence for single top quark production at the Tevatron [87].

Define and evaluate a Machine Learning algorithm

Many ML problems are formulated to minimise a loss function over a set of training examples. Loss functions quantify the non-precision of the predictions, denoted by $\hat{y}$, of the model being trained and the actual values the algorithm wants to predict, denoted by y. One of the most commonly used cost functions is the Mean Squared Error (MSE) for regression problems (predicting a continuous value), defined by Equation (2.1). For classification problems (predicting a class for the object), a commonly used cost function is the cross-entropy (CE). The CE for binary classification is defined by Equation (2.2).

$$\mathrm{MSE} = \frac{1}{2}(\hat{y} - y)^2 \tag{2.1}$$

$$\mathrm{CE} = -(y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})) \tag{2.2}$$

From these cost functions, we can define a training strategy. In the context of Deep Learning, this will be covered in Section 2.3. To train a machine learning algorithm effectively and improve its generalization capability, it's customary to use three distinct datasets: the training set, the validation set, and the test set [88,89]. The training set is used to adjust the algorithm's parameters. The ML algorithm learns from this set by adjusting its parameters to minimise errors in this data. The validation set is used to evaluate the model during training and to fine-tune its hyperparameters. This set helps prevent overfitting by providing an intermediate evaluation that the model

has not seen during training. Finally, the test set is used to estimate the model's generalisation error after complete training. This set is never used during training or validation, providing an unbiased estimate of the model's generalization ability.

Splitting the data in this way is crucial for several reasons. Firstly, it helps prevent overfitting. By using a validation set, we can detect overfitting and choose the model snapshot where it performs best. This prevents us from keeping only the last step of the model, which may be overly optimised only towards the training set and fail to generalise to new data. Lastly, the test set provides an unbiased evaluation of the model's performance on unseen and independent identically distributed (iid) data. This gives us an indication of the model's real-world performance. It's possible to have multiple test sets to evaluate the model's performance in different scenarios and distinguish its performance in each individual configuration.

After training the algorithm, we evaluate its performance to quantify its effectiveness. In the case of a classification task, the concepts of true positive (TP) and true negative (TN) define the correct outcomes predicted by the model. A true positive occurs when the model correctly predicts a class (e.g., the nature of a jet) when that class is indeed present. A true negative (TN) is an instance where the model correctly predicts the absence of that class. Conversely, a false positive (FP) occurs when the model predicts the presence of the class when it is absent (Type I error), while a false negative occurs when the model predicts the absence of the class when it is present (Type II error). Type I (false positive) and Type II (false negative) errors are critical as they impact the reliability of the model's predictions. A Type I error represents contamination in predicting the class, whereas a Type II error represents a failure to detect an existing class.

From these concepts, we can derive several metrics to evaluate the quality of the algorithm's classification. Precision is defined as the ratio of TP to the total predicted positives such as in Equation (2.3). However, precision alone may not be sufficient when adjusting the precision rate based on a prediction threshold. To further

refine our understanding of our algorithms, we use Receiver Operating Characteristic (ROC) curves. ROC curves plot the TP rate (1 - Type II error) against the FP rate (Type I error) for different decision thresholds, allowing visualization of the model's performance at various thresholds. The area under the ROC curve (ROC AUC) quantifies this performance by calculating the integral under the ROC curve, with 1 indicating perfect performance and 0.5 indicating random guessing. Conventionally and for visibility in jet tagging, we invert the axes compared to classical ROC curves. The power of rejection at a given efficiency is the inverse of the false positive rate measured at a TP threshold. This metric provides insight into the model's ability to reject contaminating classes at a typically considered rate correctly. Figure 2.1 illustrates the ROC curve and the related metrics.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.3}$$

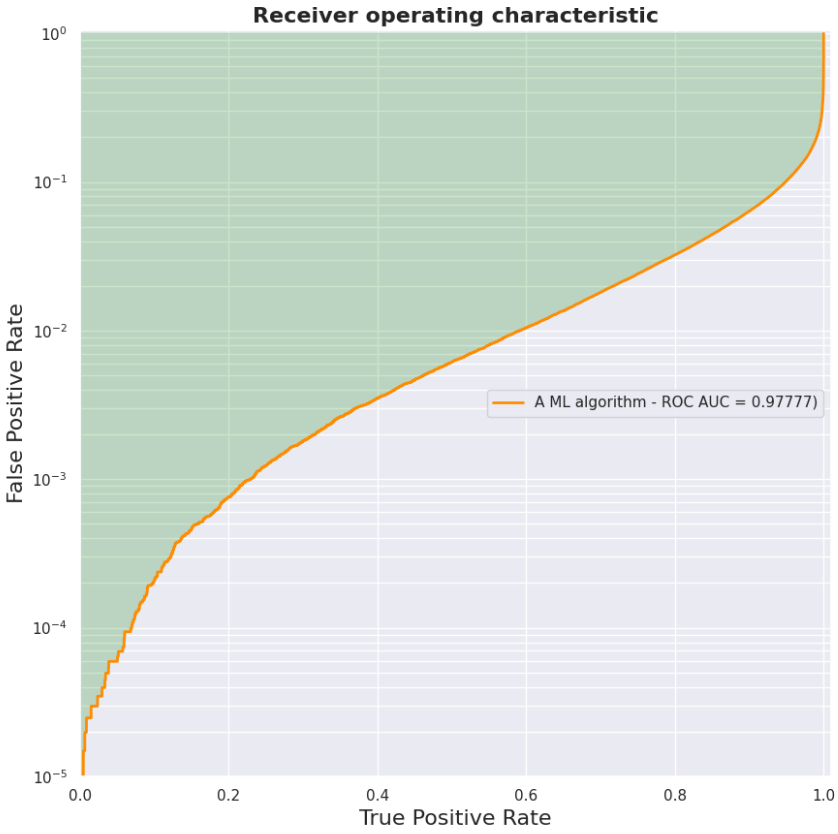

Figure 2.1: Illustration of the ROC curve for jet tagging. The ROC AUC score is indicated in the label, and the area is highlighted in green.

2.2 Neural Network structures

This Section discusses the fundamental components of the neural networks that we will use to develop the algorithms employed later. We will detail the fundamental principles in chronological order to highlight the advances that Deep Learning has experienced over time, which have led to recent developments in flavour tagging.

2.2.1 The Perceptron and Multi-Layer Perceptron

The origin of artificial neural networks dates back to the 1950s when neuroscientists and mathematicians endeavoured to create a mathematical and algorithmic foundation that mimicked the fundamental principles of biological neurons as conceptualised at the time. The first machine learning algorithm considered an artificial neural network is the Perceptron [5], initially simulated and then described by Rosenblatt in 1957 and 1958, respectively. The perceptron algorithm performs binary classification using a set of n input variables $\bar{x} = (x_1, ..., x_n)$ and an output variable o. These variables are combined linearly with n weights $\bar{W} = (W_1, ..., W_n)$ (also called synaptic coefficients) and a bias (also called threshold) b according to the following equation:

$$o(\bar{x}, \bar{W}, b) = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} W_i x_i + b \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.4}$$

The parameters W_i and b are the trainable variables of the algorithm that allow the perceptron to be adjusted in order to optimise problem-solving. The output value of the perceptron o, whose structure is illustrated in Figure 2.2, corresponds to the Heaviside function and enables the algorithm to predict the class, either 1 or 0. This output value can then be compared to the actual class y. By extending this example to a labelled data sample comprising M elements $D = \{(\bar{x}^1, y^1), ..., (\bar{x}^M, y^M)\}$, we can derive metrics to measure the quality of our prediction such as the precision or the ROC AUC.

Like any machine learning algorithm, we can derive a learning

rule to adjust the model parameters on our data sample. In the context of the perceptron, Rosenblatt starts his perceptron with all weights having an initial value of 0. Then, for each labelled data point, we can update the value of our weights relative to the cost function $\mathcal{L} = \frac{1}{2}(y^m - o(\bar{x}^m, \bar{W}))$, via Equation (2.5).

$$W_i = W_i + \lambda x_i^m (y^m - o(\bar{x}^m, W_i)) \tag{2.5}$$

Where x^m and y^m represent the input variables and the actual class of the m-th element of sample D. Additionally, we have added the bias to the list of weights as the last element $W_{n+1} = b$ and included a term for the bias in the input variables $x_{n+1} = 1$ to match the initial Equation (2.4). The value λ corresponds to the learning rate, a parameter that controls the magnitude of the weight changes with respect to the additional term. The term $(y^m - o(\bar{x}^m, W_i))$ indicates the quality of the prediction; if we predict the correct class, the algorithm does not need to adjust its weights, and the value of this term is zero.

In the decade following its creation, several mathematical properties were derived from the perceptron and its training method. Notably, the proof of convergence for the algorithm in Equation (2.5) was published in 1962 [90]. However, in 1969, Minsky and Papert derived the limitations of simple perceptrons [91]. These, constrained by their linear structure, are indeed incapable of solving non-linearly separable problems. In other words, if a hyperplane separates our two classes, then Novikoff's convergence theorem [90] guarantees the perceptron's convergence. Beyond the limitations of the simple perceptron, Minsky and Papert also contributed by demonstrating that classification tasks require non-linear transformations, which can be achieved by extending the perceptron's definition. Indeed, by changing the perceptron's output function from the simple Heaviside function to non-linear functions and extending the perceptron from a single layer to multiple layers, it was shown that we can overcome the difficulties encountered in certain non-linear problems. This revelation of what we call deep neural networks (DNNs), and more specifically, the multi-layer perceptron (MLP), established the fundamental

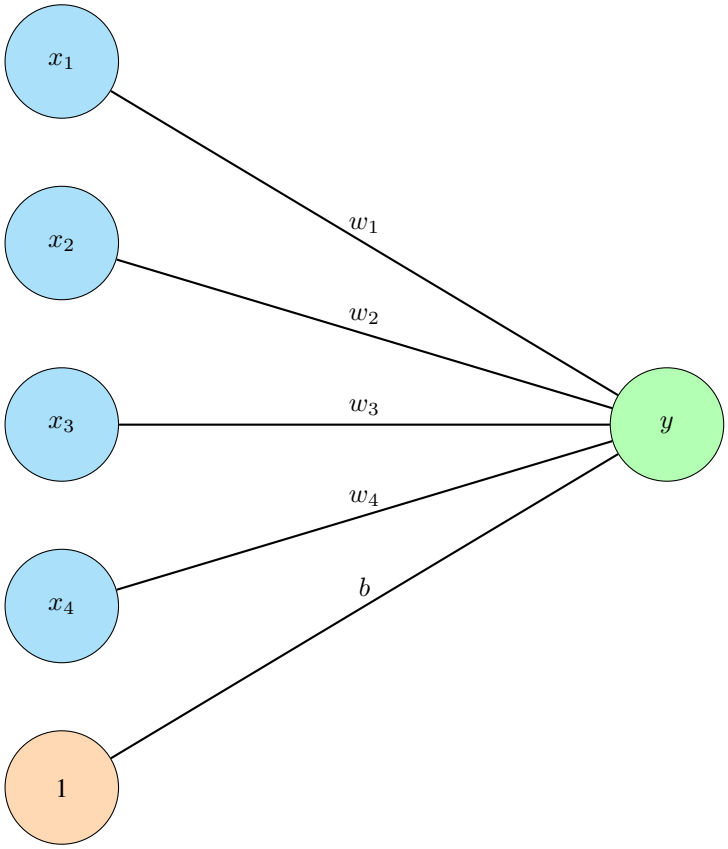

Figure 2.2: Schematic structure of the perceptron algorithm with 4 input features x_i in blue, a bias in orange and one output y in green.

principles that led to the improvement of neural networks and their learning techniques in the 1980s and 1990s, forming the foundations of contemporary deep learning techniques. A multi-layer perceptron consisting of two hidden layers is illustrated in Figure 2.3 and corresponds to Equation (2.6).

$$\begin{aligned} o_j^1(\bar{x}, \bar{W}^1) &= a^1(\sum_i x_i W_{ij}^1 + b_j^1) \\ o_j^2(\bar{o}^1, \bar{W}^2) &= a^2(\sum_i o_i^1 W_{ij}^2 + b_j^2) \\ o_j^3(\bar{o}^2, \bar{W}^3) &= a^3(\sum_i o_i^2 W_{ij}^3 + b_j^3) \end{aligned} \tag{2.6}$$

In a MLP, each layer of neurons performs a linear transformation followed by a non-linear activation. In our example, each layer j receives as input the outputs of the previous layer i. For the first layer, the inputs $\bar{x}$ are transformed into outputs $\bar{o}^1$ according to the first of Equation (2.6). Here, x_i represents the inputs, W_{ij}^1 are the weights of the first layer, b_j^1 is the associated bias, and a^1 is the activation function. The second layer takes the outputs of the first layer, $\bar{o}^1$, and transforms them into outputs according to the second equation. Similarly, the third layer transforms the outputs of the second layer, $\bar{o}^2$, into final outputs $\bar{o}^3$ according to the last equation. Each linear transformation is followed by a non-linear activation, enabling the MLP to model non-linear relationships within the system.

The sigmoid and softmax functions, defined in Equation (2.7), are frequently used to characterise probability distributions. The softmax function is particularly employed in multi-class classification tasks to transform the N output discriminants into a probability distribution over N choices. Among the commonly used activation functions, we can mention the Rectified Linear Units (ReLU) [92] and its derivatives GELU [93] and SiLU [94], whose Equation (2.8) illustrate the functions.

$$\begin{aligned} \sigma(x) &= \frac{1}{1+e^{-x}} \\ SoftMax(x) &= \frac{e^{x_i}}{\sum_j e^{x_j}} \end{aligned} \tag{2.7}$$

$$\begin{aligned} ReLU(x) &= \max(0, x) \\ GELU(x) &= x \cdot \Phi(x), \quad \Phi(x) = \frac{1}{2} \cdot (1 + (\text{erf}(\frac{x}{\sqrt{2}}))) \\ SiLU(x) &= x \cdot \sigma(x) \end{aligned} \tag{2.8}$$

The evolution of the perceptron and MLP over the years has also led to the advent of new training methods, such as the pocket algorithm [95], allowing the perceptron to be trained on non-linearly separable data with a tolerated error rate, and the Least-Mean-Square (LMS) algorithm, which minimises the MSE using the gradi-

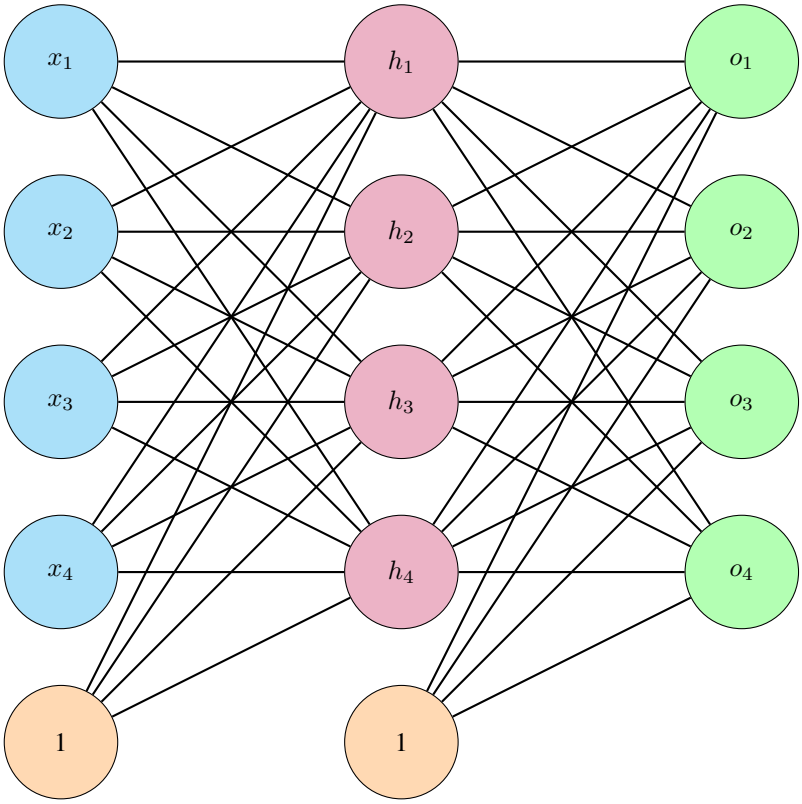

Figure 2.3: Schematic structure of the multilayer perceptron algorithm with 4 input features x_i in blue, one hidden layer consisting of 4 neurons h_i in purple and an output layer consisting of 4 neurons o_1 in green. Bias are included in the MLP and are illustrated via the orange cells.

ent descent method [96, 97]. However, the methods employed were computationally expensive, and despite the development of certain algorithms, training MLPs was complicated, with intermediate layers posing a challenge for parameter adjustment. It was not until the introduction of backpropagation in 1986 [6] that an efficient learning method was available. Backpropagation is a supervised learning method to train MLPs. It involves adjusting the weights $\bar{W}$ of the connections between neurons to minimise the error between the network's predicted output $\bar{o}$ and the desired actual output. The process occurs in two phases: the forward pass and the backward pass. In the forward pass, the input $\bar{x}$ is propagated through the network layer by layer to the output $\bar{o}$. The error is then calculated using a cost

function, such as the mean squared error. In the backward pass, this error is backpropagated through the network using the derivative of the cost function with respect to the weights calculated via the chain rule. The weights are updated using the gradient descent algorithm, which allows the error to be gradually reduced with each training iteration. More formally, for each layer l, the error δ^l is calculated, and the weights $\bar{W}^l$ are updated according to the rule $\Delta\bar{W}^l = -\eta\frac{\partial\mathcal{L}}{\partial\bar{W}^l}$, where η is the learning rate and $\mathcal{L}$ is the cost function. Backpropagation, formalised by Rumelhart, Hinton, and Williams in 1986 [6], revolutionised deep learning by enabling the efficient training of deep neural networks.

2.2.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a class of deep neural networks primarily used for image processing and visual recognition tasks. Their usage was popularised in the 1990s [98] with the advent of backpropagation, which allowed CNNs to update weights through gradient-based methods directly from the images used for training.

$$Y(i,j) = (X \cdot F)(i,j) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} X(i+m, j+n) \cdot F(m,n) \tag{2.9}$$

CNNs exploit the local spatial structures of input data through convolutional layers. Convolutional layers are the fundamental building blocks of CNNs. A convolutional layer applies filters F (or kernels) to the input to produce feature maps. Each filter performs a convolution operation on the input, detecting local patterns such as edges, textures, or simple objects. For instance, in image recognition, a two-dimensional kernel K characterised by a height and width dimension (M, N) represents a window capable of scanning the input in two dimensions. The 2-dimensional discrete convolution operation for an input X and a filter F is defined by Equation (2.9), where i, j represent the 2D coordinate of the output obtained by the convolution of the image with the kernel F and m, n represent the kernel

steps characterised by the chosen window size M, N of our kernel filter F. Beyond the equation, we can also adjust the window sliding, also called stride. Returning to our image example, when we move our window pixel by pixel to observe, we perform a stride of 1. However, we could also move the window by a larger number of pixels, thereby increasing the stride value. A larger stride reduces the number of scans performed by the kernel and, thus, the computational complexity of our neuron.

In the context of jet tagging, despite some approaches using the 2D convolution paradigm [99, 100], most of the application requires a 1D convolution. The discrete 1-dimensional convolution operation, which reflects the type of operation one could apply on a jet's constituents tensor, is then defined as follows:

$$Y(i) = (X \cdot F)(i) = \sum_{m=0}^{M-1} X(i+m)F(m) \tag{2.10}$$

where i is the 1D coordinate of the image output, m the kernel step of the 1D kernel F of dimension M.

The advantages of convolution are numerous. Firstly, it allows the kernel size to capture local dependencies between neighbouring elements. Additionally, the number of parameters is significantly reduced, as illustrated by Figure 2.4, decreasing the model's size and complexity. Typically, to process 300×300 pixels of an image with 3 RGB channels using a hidden layer of an MLP composed of 100 outputs would require 27.27 million parameters, whereas a convolutional neural layer would require only 7600 parameters, accounting for an added bias for each output neuron. CNNs quickly demonstrated their superior capability over MLPs for image processing [101, 102]. Finally, note that an input image of size (H, W), to which we apply a 2-dimensional convolutional neuron with a kernel size of (M, N) and a stride S, will produce an output of dimension $((H - M)/S + 1, (W - N)/S + 1)$. Therefore, stride values greater than one will reduce the image resolution and act like downsampling.

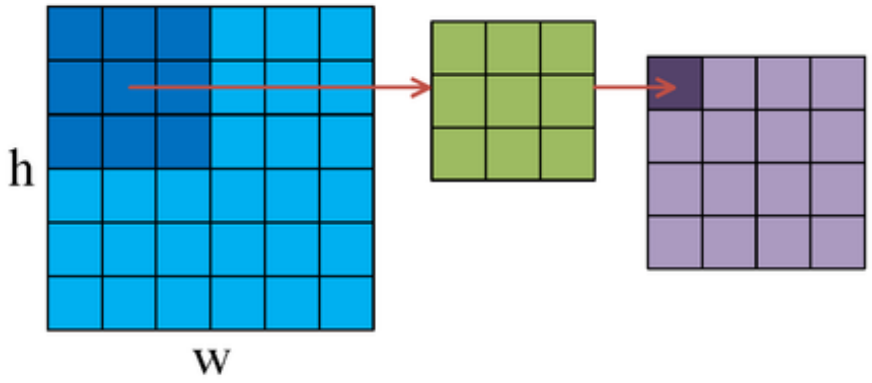

Figure 2.4: Schematic of a 2D CNN [103] where the input image in blue, of dimension $h \times w$ is convoluted with the kernel F in green of dimension 3×3 for obtaining the output Y in purple. No striding is applied, and therefore, Y is of dimension $h - 2 \times w - 2$.

2.2.3 Recurrent Neural Networks

A Recurrent Neural Network (RNN) is an artificial neural network that processes sequential data. Unlike traditional neural networks, where connections between units are unidirectional, RNNs have loops that allow information to persist from one sequence element to the next. By reintroducing the output of neurons into the construction of the variables for the next element of the sequence, this architecture retains a memory of previous states, making them particularly suitable for tasks such as speech recognition [104] and machine translation [105]. Most modern properties of RNNs were theorised by the Lenz-Ising model in 1925 [106] and were adapted for neural learning through the work on backpropagation [6].

Among the early recurrent neural networks, we can cite the Elman network as an example [107]. Often referred to as a simple recurrent network, this network has a simple structure, which is as follows:

$$\begin{aligned} h_t &= \sigma_h(W_h x_t + U_h h_{t-1} + b_h) \\ y_t &= \sigma_y(W_y h_t + b_y) \end{aligned} \tag{2.11}$$

Where x_t, h_t, and y_t represent the input vector, hidden state, and output vector for the t-th element of the sequence. σ_h and σ_y are the activation functions of our neural network, and the tensors W_h, U_h, W_y represent the weights, while b_h and b_y are the biases if we choose to incorporate them. We clearly observe the recurrence relation introduced in Equation (2.11) where our hidden state h_t is constructed via the input vector x_t but also the previous hidden state in the sequence h_{t-1}. Thus, the recurrent neural network can build relationships between different sequence elements with a causality relationship, allowing information from the previous sequence to persist.

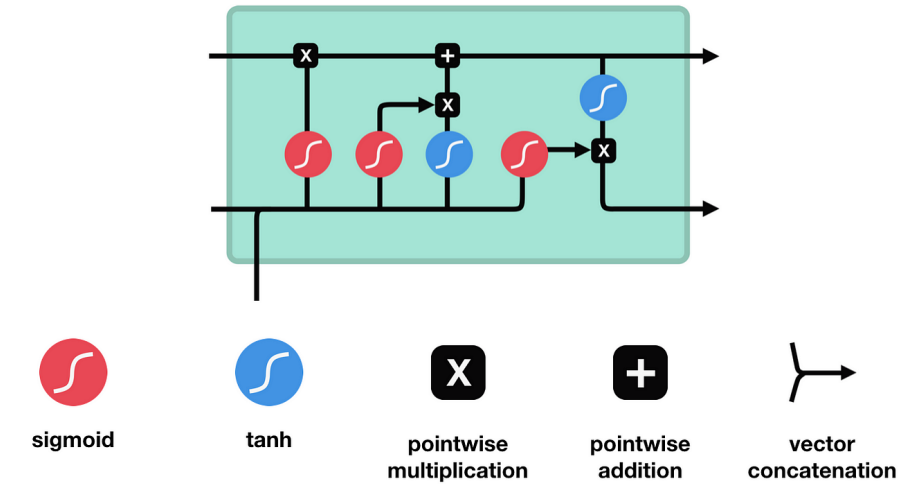

Figure 2.5: Schematic of a LSTM from [108]

Among the several types of recurrent neural networks with distinct structures and recurrence mechanisms, we will focus on a specific RNN structure, the Long Short-Term Memory (LSTM) architecture [109]. LSTMs are common and highly efficient RNNs that have

been used in applications in HEP [110]. The Long Short-Term Memory (LSTM) is a specific architecture of recurrent neural networks (RNN) designed to overcome the limitations of traditional RNNs, notably the vanishing or exploding gradient problems [109]. The vanishing gradient problem occurs when the gradients of certain layers in a neural network, usually the earlier ones, become very small during backpropagation. This phenomenon can make it difficult to learn these layers due to negligible updates incapable of converging to the desired optimum. On the contrary, gradient explosion happens when gradients become excessively large, leading to unstable weight updates and often resulting in infinite values propagating throughout the model, rendering it unusable. LSTMs introduce memory cells and gating mechanisms via a system of input gate, output gate, and forget gate to regulate the flow of information. These gates, illustrated in Figure 2.5, allow the network to retain or forget information over long periods, making LSTMs particularly effective for tasks involving long-term temporal dependencies compared to previous RNNs. The LSTM operates recurrently, and we can define its operation for each element of the sequence as follows:

$$\begin{aligned} f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\ i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\ o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\ \tilde{c}_t &= \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\ c_t &= f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t \\ h_t &= o_t \cdot \sigma_h(c_t) \end{aligned} \tag{2.12}$$

Where x_t is the input variable vector at step t of the sequence, h_{t-1} is the hidden state, also called the output variable, at step $t-1$, and c_{t-1} is the memory cell state at step $t-1$. The variables f_t, i_t, and o_t are the outputs of the forget gate, input gate, and output gate at step t, while $\tilde{c}_t$ is the activation gate. σ_g, σ_c, and σ_h represent the activation functions for the gates, memory cells, and output variables, respectively, the sigmoid function for the first and hyperbolic tangent for the latter two. The weight matrices for the inputs W_k, for the hidden states U_k, and the bias b_k are used in the different gates and

memory cells and represent the learning parameters of the LSTM.

2.2.4 Attention Neural Networks

An attention neural network, often called an 'attention mechanism,' is a component used in various neural networks, particularly in natural language processing (NLP). The primary purpose of an attention mechanism is to dynamically focus on different parts of the input data when producing an output. This is particularly useful in tasks where the relevance of different input parts varies depending on the context. Unlike previous architectures, specifically recurrent models that use fixed-size windows or recurrent connections, the attention mechanism dynamically assigns weights to individual elements based on their relevance, capturing complex dependencies across the entire sequence structure. Furthermore, many modern architectures such as Transformer models [111] offer a structure that is not sequential but parallel, allowing the outputs of each element of the sequence to be computed simultaneously, thus facilitating and accelerating both training and inference compared to RNNs.

Attention mechanisms assign different weights to various elements of the input data, hence determining the influence of each element on the output. Focusing on the relevant parts of the input data based on the current context, attention mechanisms help the model to better capture dependencies and relationships within the data. There are various types of attention mechanisms, the most common being additive attention [112], multiplicative attention [113], and scaled-dot product attention [111]. The latter is the key component of Transformer models, which have since demonstrated state-of-the-art performance across a wide range of tasks such as NLP [76, 77] and CV [74, 75]. Transformer models can be defined as a class of deep neural networks characterized by the usage of a specific attention mechanism, the scaled-dot product attention, first introduced by Vaswani et al. in 2017 [111].

The scaled-dot product attention (SDPA) mechanism uses three inputs: a query matrix Q, a key matrix K, and a value matrix V. The query matrix represents the elements for which attention weights are

calculated, while the key and value matrices represent all sequence elements. After being injected into linear layers, the query tensor Q of dimension (B, N, d_k), the key tensor K of dimension (B, L, d_k), and the value tensor V of dimension (B, L, d'_k) are introduced into the scaled-dot product attention through Equation (2.13). The input tensors Q, K, and V are obtained via linear layers, facilitating the transformation and projection of the input tensors into the attention space.

$$\text{Attention}(Q, K, V) = \text{SoftMax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V. \tag{2.13}$$

The attention mechanism in this work is employed in a specific configuration where the query, key, and value tensors are identical. This particular case is commonly referred to as self-attention. Self-attention allows each element of the sequence to focus on other elements of the same sequence, which is particularly useful for capturing long-range dependencies. It should also be noted that the term $\sqrt{d_k}$ used in the denominator of the matrix product QK^T helps control the range of values before applying the softmax function. This allows us to stabilise training and improve model convergence [111].

The SDPA, depicted on the left in Figure 2.6, is extended to enhance the model's discriminative power by allowing it to focus on multiple attention subspaces in parallel. This extension, called Multi-Head Attention (MHA), facilitates the capture of diverse and complementary high-level features from the input by independently projecting the query, key, and value matrices for each of the h attention heads. Each attention head performs an SDPA operation, producing distinct representations. These head representations are then concatenated and passed through a linear layer to integrate the information across the heads, as illustrated on the right in Figure 2.6. The MHA layer can be mathematically represented by the following equations:

$$\begin{aligned} \text{MHA}(Q, K, V) &= \text{Concat}(h_1, ..., h_n)W^O, \\ h_i &= \text{Attention}(QW^{Q,i}, KW^{K,i}, VW^{V,i}) \end{aligned} \tag{2.14}$$

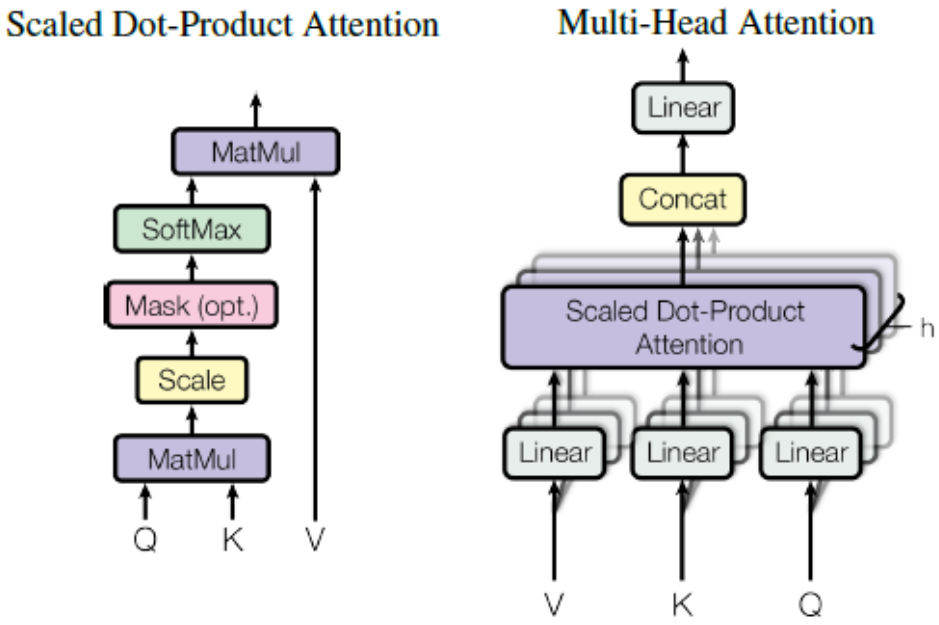

Figure 2.6: Schematics of the SDPA (left) and MHA (right) from [111]

were for n attention head, each head h_i is built through the SDPA mechanism. The output tensor produced by each head is concatenated, and the information is mixed across all the features of the head through a shared linear layer defined by the weights W^O.

Compared to traditional recurrent models such as RNNs (LSTMs in particular), the attention mechanism and Transformers allow for more efficient parallelization. They are less prone to vanishing or exploding gradient problems, making them particularly suitable for tasks requiring contextual modelling and long sequences.

2.2.5 Normalization layers and internal covariate shift

Normalization layers were introduced to improve the stability and speed of convergence of deep neural networks [114, 115]. Training deep networks can be sensitive to the distribution of activations in each layer, which can vary from one element to another during learning. This phenomenon, also known as internal covariate shift [114],

slows down training because each layer must continuously adapt to changes in the distribution of its inputs. To counteract this, normalization techniques aim to fix the distribution of activations during training, allowing networks to converge faster and more stably, as illustrated by Figure 2.7. Normalization techniques have also proven effective in improving the robustness of neural networks to weight initialization and can act as a regularization term [114]. In the context of our algorithms, the normalizations applied all start with the same formulation:

$$y = \frac{x - E(x)}{\sqrt{Var(x) + \epsilon}} \cdot \gamma + \beta \tag{2.15}$$

where γ and β are trainable variables for creating an afine transformation of the normalised values and ϵ is a small value for numerical stability.

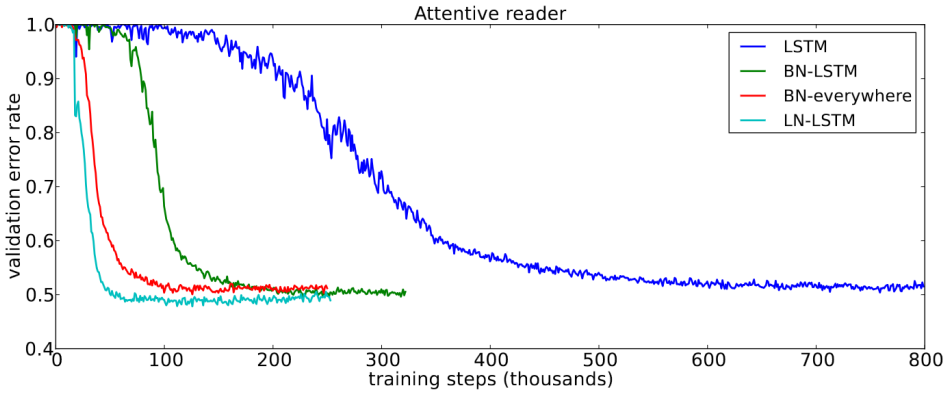

Figure 2.7: Comparison of the validation error over the training step for training of LSTM models using different normalization approaches [115]. We can see that the convergence of the training of the LSTM models on the CNN corpus dataset is faster with normalization than with the bare version of LSTM.

Among the most frequently used normalization methods, batch normalization [114] normalises the activations of a layer using the statistics (mean and variance) calculated on each mini-batch during training. In the context of batch normalization, the parameters γ and β are vectors of dimension d representing the number of features describing each constituent of our jet. The means $E(x)$ and variance

$Var(x)$ are measured for each feature across all constituents in the considered minibatch.

In contrast, layer normalization [115] proposes a different approach based on the same equation. Instead of normalising the activations based on the mini-batch, it normalises the activations of each neuron individually using the statistics of all activations in the same layer. This method is particularly useful in architectures such as recurrent or attention networks where calculating statistics on entire mini-batches can be problematic.

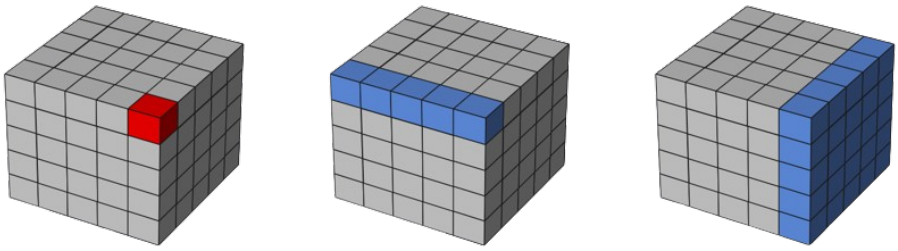

Figure 2.8: Illustration of layer normalization and batch normalization where the values B, N, and C represent the batch axis (or jets), the jet constituents axis, and the variables axis, respectively. On the left, in red, is the variable for which normalization is applied; in the center are the variables considered for measuring the mean and variance for layer normalization; and on the right are those considered for batch normalization.

2.2.6 Dropout

Dropout [116] is a regularization method used in DL to prevent overfitting. During each training step, a subset of neurons passing through dropout is temporarily deactivated by being set to a default value of zero during the forward pass. These neurons no longer contribute to the prediction for the given step and do not contribute to learning during backpropagation, as illustrated in Figure 2.9. The elements are deactivated based on a Bernoulli distribution with a predefined rate, also called dropout rate, and usually ranging between 10% and 50%. During model evaluation, dropout is turned off, allowing the neural network to utilise all information to make the most

accurate prediction possible.

This procedure forces the network to learn more robust and general data representations, avoiding excessive dependence on specific neurons where overfitting effects might occur. During backpropagation, neurons can form co-adaptations that work well on the training dataset but perform poorly on evaluation data and in later use [116]. Dropout breaks these co-adaptations, thus requiring each neuron to participate more evenly in the decision-making process, improving the model's ability to generalise to unseen data. Dropout has proven to be an effective regularization method and generalizable across various domains, being state-of-the-art in many applications such as computer vision, document classification, and speech recognition [116].

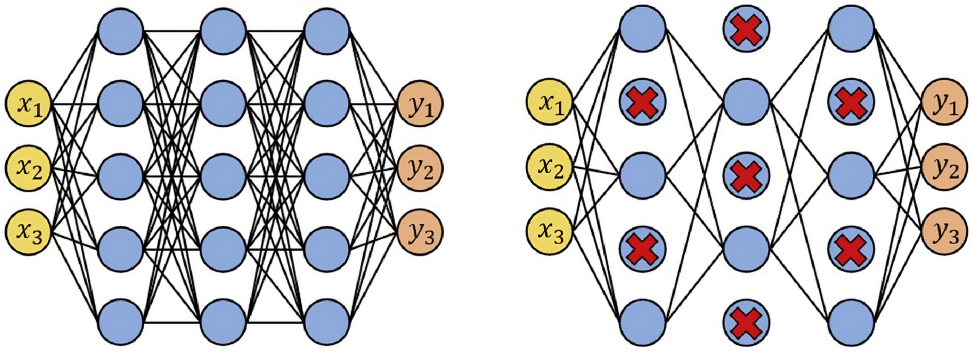

Figure 2.9: Illustration of the dropout mechanism [117]. During training, the dropout will randomly select some neurons to turn off with a fixed dropout rate.

2.3 Optimizers for Neural Networks training

This Section is dedicated to the description of the training methods used in modern Deep Learning. We will begin with its foundational elements, namely backpropagation and stochastic approximations, in Section 2.3.1, before outlining step by step the evolution of learning algorithms, also known as optimizers, in the following Sections.

2.3.1 Stochastic Approximations and Backpropagation

Modern learning methods for artificial neural networks all inherit from stochastic approximations. Stochastic approximations are a family of iterative approximation methods used to solve problems where the exact solution is difficult to obtain, often due to random variables. Introduced by Herbert Robbins and Sutton Monro in 1951 [118], these techniques work by gradually adjusting an initial estimate using randomly sampled data, following an iterative process that converges to an optimal solution or an acceptable approximation. After initial attempts at learning via stochastic approximation in the 1950s [5, 118, 119], the development of backpropagation in 1986 [6] demonstrated the effectiveness of this learning method for MLP neural networks using the stochastic gradient descent (SGD) method.

The preliminary step of backpropagation, also known as the forward pass, is achieved after computing the output of our neural network and evaluating the quality of our prediction with our cost function $\mathcal{L}$. This function $\mathcal{L}$ must be differentiable and continuous to ensure the proper functioning of backpropagation. Finally, the cost function must be defined such that its minimum corresponds to a perfect prediction by the model. In other words, it must penalise prediction errors so that reducing its value improves the accuracy of the model's prediction.

Backpropagation is based on the chain rule and allows us to find, for each layer and trainable parameter of the neural network, a partial derivative with respect to the cost function $\mathcal{L}$. These partial derivatives will then be used to establish a training method to update the weights and biases of the neural network to find the minimum of

$\mathcal{L}$. To illustrate the functioning of backpropagation, we will consider an MLP containing N layers, with the last layer containing a single neuron that produces the prediction $\hat{y} = o^n$. Once the cost function $\mathcal{L}$ is calculated, we can compute the gradient for the last layer n via Equation (2.16), where $\hat{x}^n$ represents the neuron's response before the activation function, in other words: $\hat{x}^n_j = \sum_i o^{n-1}_i W^n_{ij} + b^n_j$. After obtaining this first partial derivative, we can reuse the chain rule to obtain the partial derivative with respect to the output neuron's weights as illustrated by Equation (2.17).

$$\frac{\partial \mathcal{L}}{\partial \hat{x}^n} = \frac{\partial \mathcal{L}}{\partial o^n} \cdot \frac{\partial o^n}{\partial \hat{x}^n} \tag{2.16}$$

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial W^n_{ij}} &= \frac{\partial \mathcal{L}}{\partial \hat{x}^n} \cdot \frac{\partial \hat{x}^n}{\partial W^n_{ij}} \\ &= \frac{\partial \mathcal{L}}{\partial \hat{x}^n} \cdot \frac{\partial}{\partial W^n_{ij}} (\sum_i o^{n-1}_i W^n_{ij} + b^n_j) \\ &= \frac{\partial \mathcal{L}}{\partial \hat{x}^n} \cdot o^{n-1}_i \end{aligned} \tag{2.17}$$

After obtaining the partial derivatives of the last layer and its weights, we can now generalise the rule for any layer by applying the chain rule to the partial derivatives. By using the recurrence relation between each layer 2.18, we can obtain the partial derivatives for the $n-l$-th layer in the backpropagation pass, that is, the output layer o^{n-l}, defined by the system of Equations (2.19).

$$\begin{aligned} o^k &= a^k(\hat{x}^k) \\ \hat{x}^k_j &= \sum_i o^k_i W^k_{ij} + b^k_j \end{aligned} \tag{2.18}$$

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \hat{x}^{n-l}} &= \frac{\partial \mathcal{L}}{\partial o^{n-l}} \cdot \frac{\partial o^{n-l}}{\partial \hat{x}^{n-l}} \\ &= \frac{\partial \mathcal{L}}{\partial o^n} \cdot \frac{\partial o^n}{\partial \hat{x}^n} \cdot \frac{\partial \hat{x}^n}{\partial o^{n-1}} \cdots \frac{\partial o^{n-l+1}}{\partial \hat{x}^{n-l+1}} \cdot \frac{\partial \hat{x}^{n-l+1}}{\partial o^{n-1}} \cdot \frac{\partial o^{n-l}}{\partial \hat{x}^{n-l}} \\ \frac{\partial \mathcal{L}}{\partial \hat{W}^{n-l}} &= \frac{\partial \mathcal{L}}{\partial \hat{x}^{n-l}} \cdot \frac{\partial \hat{x}^{n-l}}{\partial W^{n-l}} \end{aligned} \tag{2.19}$$

From Equation (2.18), we can obtain the values of the partial derivatives, $\frac{\partial o^k}{\hat{x}^k} = \dot{a}^k(\hat{x}^k)$ and $\frac{\partial x^k}{o^{k-1}} = W^k$, where $\dot{a}^k$ represents the first derivative of the activation function a^k. By combining this result with Equation (2.19), we obtain the following relationships:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \hat{x}^{n-l}} &= \frac{\partial \mathcal{L}}{\partial o^n} \cdot \dot{a}^n(\hat{x^n}) \cdot W^n ... \dot{a}^{n-l+1}(\hat{x}^{n-l+1}) \cdot W^{n-l+1} \cdot \dot{a}^{n-l}(\hat{x}^{n-l}) \\ \frac{\partial \mathcal{L}}{\partial \hat{W}^{n-l}} &= \frac{\partial \mathcal{L}}{\partial \hat{x}^{n-l}} \cdot \frac{\partial \hat{x}^{n-l}}{\partial W^{n-l}} \\ &= \frac{\partial \mathcal{L}}{\partial \hat{x}^{n-l}} \cdot o^{n-l}\end{aligned} \tag{2.20}$$

2.3.2 Stochastic gradient descent

By applying backpropagation throughout the entire artificial neural network, we can establish a relationship between our cost function and all the weights and biases. This allows us to define a learning rule. The first method popularised with the advent of backpropagation was Stochastic Gradient Descent, whose version adapted for machine learning was first introduced in 1952 [119]. The version adapted for backpropagation, aiming to minimise the cost function $\mathcal{L}$, is as follows:

$$W_t = W_{t-1} - \lambda \cdot \frac{\partial \mathcal{L}}{\partial W_{t-1}} \tag{2.21}$$

Where λ is the learning rate and W_t and W_{t-1} represent the parameters of our model at training steps t and $t-1$, respectively.

We prefer the minibatch SGD method over its original version, which iterates through each element of the training dataset one by one due to its efficiency and speed. This approach improves convergence and is more suitable for large datasets, making model training faster and more stable. Minibatch SGD updates the parameters using a subset of the data, allowing control over the variance in

parameter updates. This compromise maintains computational efficiency compared to the full gradient descent method, which requires computing the gradient for the entire dataset, incurring a significant computational cost. Mathematically, the minibatch SGD algorithm for updating the parameters W_t can be expressed as:

$$W_t = W_{t-1} - \frac{\lambda}{n} \cdot \sum_{i=1}^{n} \frac{\partial \mathcal{L}(x_i, W_{t-1})}{\partial W_{t-1}} \tag{2.22}$$

2.3.3 Momentum optimizers

The success of backpropagation and SGD has led to numerous advances in the training of deep neural networks, leading to the development of the first CV algorithms [101, 120, 121] and language representation models [122], also known as word embeddings. However, simple SGD can converge slowly, especially in cost function valleys where gradients can have small values. To accelerate convergence, SGD optimization with momentum is often used [123, 124]. Momentum is a technique that accelerates the convergence of gradient descent by accumulating a portion of the gradient from previous iterations, allowing the parameter updates of the neural network to maintain their 'inertia' and avoid stagnation in regions of low gradient. The parameter update with momentum is given by the following equations:

$$\begin{aligned} v_t &= \beta v_{t-1} + (1-\beta) \cdot \frac{\partial \mathcal{L}}{\partial W_{t-1}} \\ W_t &= W_{t-1} - \lambda \cdot v_t \end{aligned} \tag{2.23}$$

In Equation (2.23), v_t represents the update velocity of the parameters at step t. The momentum coefficient, denoted as β, ranges between 0 and 1 and controls the fraction of momentum retained as well as the fraction updated via the gradient at step t.

A variant, based on Nesterov momentum [123], anticipates parameter updates by calculating the gradient not at the current position of the parameters but at the expected position. This anticipation

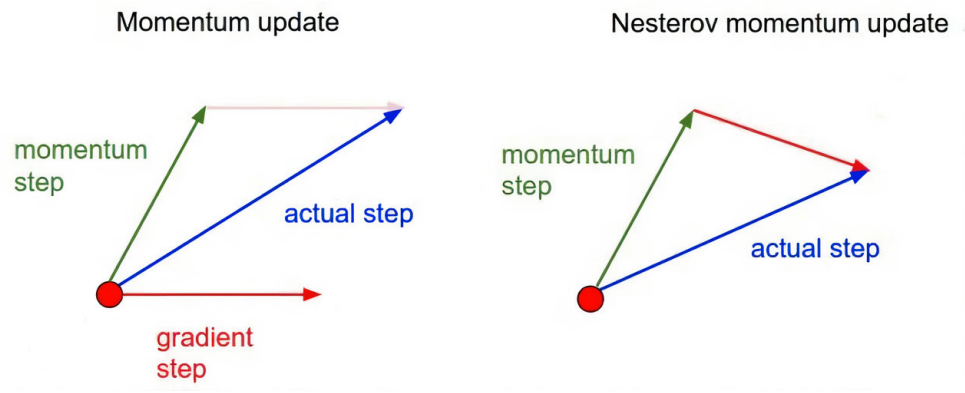

Figure 2.10: SGD momentum (left) and SGD Nesterov momentum (right) from [125]. The red dot represents the current weight value. The Nesterov lookahead gradient is evaluated at the shifted weights

allows the optimizer to correct its velocity if it is heading in a suboptimal direction. Nesterov momentum, illustrated in comparison with classical momentum in Figure 2.10, enables finer and often more effective adjustments, leading to faster and more stable convergence. The updated equations for Nesterov momentum are:

$$\begin{aligned} \tilde{W}_{t-1} &= W_{t-1} - \beta \cdot v_{t-1} \\ v_t &= \beta v_{t-1} + (1-\beta) \cdot \frac{\partial \mathcal{L}}{\partial \tilde{W}_{t-1}} \\ W_t &= W_{t-1} - \lambda \cdot v_t \end{aligned} \tag{2.24}$$

SGD with momentum and Nesterov momentum are powerful techniques for improving the convergence speed of gradient descent optimization. By anticipating updates, Nesterov momentum can offer superior performance in certain machine-learning scenarios such as CV [124].

2.3.4 Modern technique of learning

Following the success of SGD with momentum, several optimizer variations were created to improve the convergence of neural networks facing increasingly complex problems and requiring adjustments to optimise training convergence for non-convex problems [126, 127]. In

particular, the Adam optimizer (Adaptive Moment Estimation) [128] is a widely used stochastic optimization method for training neural networks due to its ability to adapt learning rates dynamically to converge quickly. This simple and computationally efficient algorithm allows the training of neural networks on large datasets with many training parameters. This method has demonstrated its robustness in training models for numerous complex problems such as CV [75], NLP [76] and jet algorithm [129].

The Adam optimizer computes the moving averages of the first and second moments of the gradients. The parameter update equations are given by:

$$\begin{aligned} m_t &= \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot \frac{\partial \mathcal{L}}{\partial W_{t-1}} \\ v_t &= \beta_2 \cdot v_{t-1} + (1-\beta_2) \cdot \left(\frac{\partial \mathcal{L}}{\partial W_{t-1}}\right)^2 \\ \hat{m}_t &= \frac{m_t}{1-\beta_1^t} \\ \hat{v}_t &= \frac{v_t}{1-\beta_2^t} \\ W_t &= W_{t-1} - \lambda \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \end{aligned} \tag{2.25}$$

The terms m_t and v_t are the exponential moving averages (EMA) of the first and second moments of the gradients, respectively. The coefficients β_1 and β_2 are the exponential decay rates for the moments, which the authors suggest setting to 0.9 and 0.999. The terms $\hat{m}_t$ and $\hat{v}_t$ are the bias-corrected moments, correcting the initialization of EMA moment to vectors of zeros. ϵ is a regularization term to prevent division by zero, generally set to 10^{-8} as suggested by the authors. The parameter $\frac{1}{\sqrt{\hat{v}_t}+\epsilon}$ represents the adaptive learning rate of the Adam algorithm.

Despite its success, the Adam optimizer faces difficulties with certain complex problems. In particular, it has been shown that it requires a warm-up period with a low learning rate to achieve satisfactory convergence and generalization of neural network predictions

[76, 111]. Studies have shown that the Adam optimizer suffers from high variance in adaptive learning rates during the initial iterations of training [130], primarily due to the term $\hat{v}_t$ whose variance $\mathrm{Var}[\frac{1}{\sqrt{v_t}}]$ can be undesirably large or even diverge under certain conditions. Additionally, given that $\mathrm{Var}[\lambda \cdot \frac{1}{\sqrt{v_t}}] = \lambda^2 \cdot \mathrm{Var}[\frac{1}{\sqrt{v_t}}]$, we can control the impact of this high variance by adjusting the learning rate λ to lower values during the initial iterations. This confirms the observations from empirical training.

To address this issue in the learning algorithm, we can use the Rectified Adam (RAdam) optimizer, which introduces a variance rectification. The training equations incorporating the rectification parameters are defined as follows:

$$\begin{aligned}
\rho_\infty &= \frac{2}{1-\beta_2} - 1 \\
m_t &= \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot \frac{\partial \mathcal{L}}{\partial W_{t-1}} \\
v_t &= \beta_2 \cdot v_{t-1} + (1-\beta_2) \cdot \left(\frac{\partial \mathcal{L}}{\partial W_{t-1}} \right)^2 \\
\hat{m}_t &= \frac{m_t}{1-\beta_1^t} \\
\rho_t &= \rho_\infty - \frac{2t\beta_2^t}{1-\beta_2^t}
\end{aligned} \tag{2.26}$$

In addition to the standard parameters of Adam, we introduce ρ_t, the 'length' or the number of degrees of freedom of the adaptive learning rate variance under the approximation of the EMA by a simple moving average (SMA) [130], as well as the maximum 'length' ρ_∞ obtained under the same approximation.

Using this length, we can obtain the following parameter update rule:

$$
\begin{aligned}
&\textbf{if } \rho_t > 4: \\
&\quad l_t = \frac{\sqrt{1-\beta_2^t}}{\sqrt{v_t}+\epsilon} \\
&\quad r_t = \sqrt{\frac{(\rho_t-4)(\rho_t-2)\rho_\infty}{(\rho_\infty-4)(\rho_\infty-2)\rho_t}} \\
&\quad W_t = W_{t-1} - \lambda \cdot \hat{m}_t \cdot r_t \cdot l_t - \lambda \cdot \alpha \cdot W_{t-1} \\
&\textbf{else }: \\
&\quad W_t = W_{t-1} - \lambda \cdot \hat{m}_t - \lambda \cdot \alpha \cdot W_{t-1}
\end{aligned}
\tag{2.27}
$$

where the parameter ρ_t sets a condition to use the adaptive learning rate λ_t, which is corrected by the variance rectification term r_t. For values of ρ_t less than 4, which occurs in the initial iterations of training, the variance of the system becomes too large, and we update the algorithm without the rectified adaptive learning rate. Finally, we also use a decoupled weight decay term [131] with a decay parameter α to add a regularization parameter on the weights of our neural network, hence improving its generalization.

One final optimization method we will use is the Lookahead optimizer [132]. Lookahead is an optimization method that enhances the performance of other optimizers by combining them with a specific update strategy. The main idea behind Lookahead is to separate the optimization into two distinct processes: we have two sets of parameters, the slow weights ϕ and the fast weights θ. At each training iteration, we use a standard optimizer such as SGD, Adam, or RAdam to update the weights θ. Then, after k internal steps of our standard optimizer, we update the slow weights ϕ by performing a linear interpolation between the weight spaces of ϕ and θ. The fast weights θ are then reset to the values of the slow weights ϕ.

The process of updating the slow weights is achieved through an exponential moving average of the fast weights from the last iteration k of the fast weights' inner loop as follows:

$$\begin{aligned}\phi_t &= \phi_{t-1} + \alpha \cdot (\theta_{t-1,k} - \phi_{t-1}) \\ &= \alpha \cdot (\theta_{t-1,k} + (1-\alpha) \cdot \theta_{t-2,k} + ... + (1-\alpha)^{t-1} \cdot \theta_{0,k}) \\ &\quad + (1-\alpha)^t \cdot \phi_0\end{aligned} \tag{2.28}$$

The combination of slow and fast weights allows the Lookahead optimizer to improve learning in regions with high curvature, as illustrated in Figure 2.11, while reducing variance to enable faster convergence of the neural network [132].

In this thesis, we will often use the combination of Lookahead with RAdam, commonly referred to as Ranger [133]. This approach leverages the improvements of RAdam over Adam in terms of robustness, particularly at the beginning of training, to avoid erratic behaviours. Lookahead ensures stability and performance in model convergence. By merging these two approaches, Ranger offers faster and more stable convergence, reducing the risk of overfitting and improving model generalization.

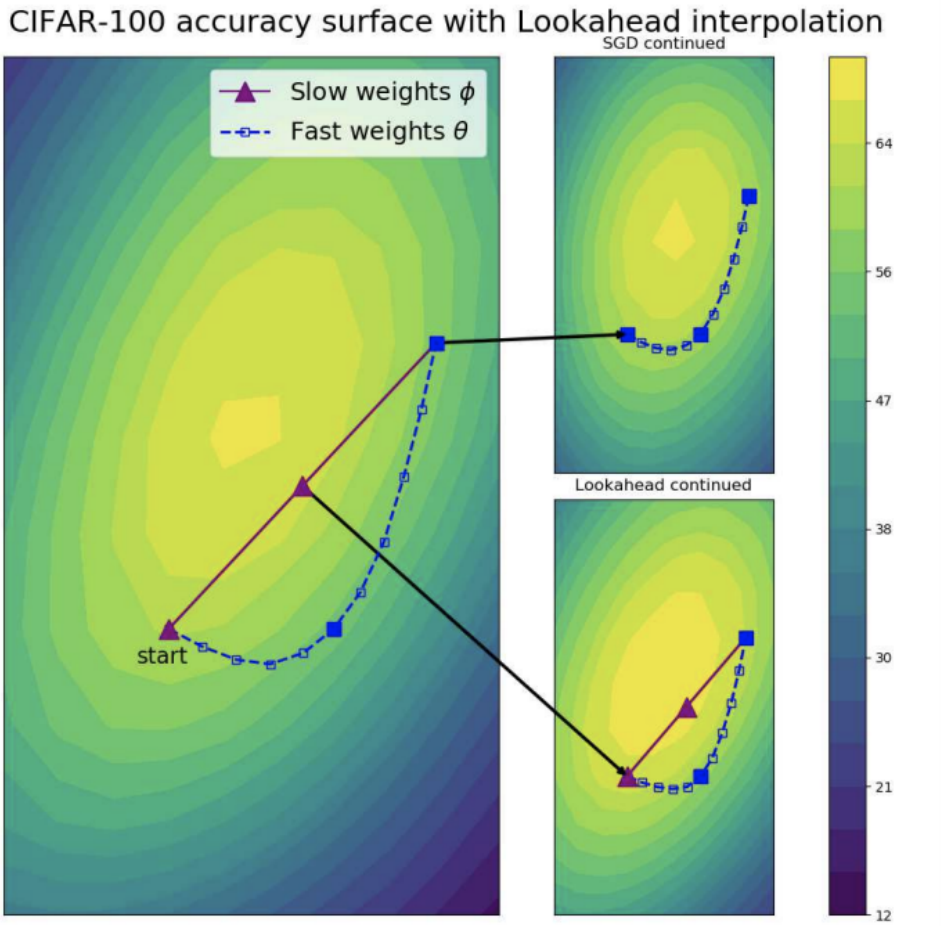

Figure 2.11: Visualization of the lookahead optimizer [132] method (left) with $k = 10$. We can observe after 20 training steps that the slow weights of the lookahead method (right bottom) have been able to interpolate a region of higher accuracy, while SGD alone has not yet explored this region (right top).

Chapter 3

Heavy-flavour tagging with Deep Learning

In this chapter, we will describe the application of deep learning to b and c jet tagging, also known as heavy-flavour tagging. Specifically, we will start by elaborating on the concept of jet representation as a Particle Cloud in Section 3.1. We will discuss essential properties and concepts to understand how modern jet tagging algorithms were developed, focusing on the concepts of sets and graphs to handle the observables we consider. Next, in Section 3.2, we will discuss the architectures of neural networks used in the CMS experiment in the context of flavour tagging. First, we will describe early algorithms treating variables as vectors, inheriting from multivariate analysis (MVA), sequence models recurrently processing the jet constituents and the first algorithms considering the jet as a graph. We will then establish the necessary connections between the Particle Cloud representation and the most efficient modern model structures based on the Transformer architecture. Here, we will describe DeepJet Transformer, the first Transformer-based model for jet tagging, and Particle Transformer, an advanced version introducing new edge variables. In the context of my thesis, we will focus on resolved jets at the CMS experiment. Thus, in Section 3.3, we will describe the jets used to train our algorithms and how they are simulated. We will also describe the dataset used, the labelling procedure as well as the input

features considered. We will then describe the training procedure and environment used for performing the training of the models considered. Finally, in Section 3.4, we will discuss performance results and conclude on the evolution of deep learning algorithms. We will explore both the performance of the algorithm for identification and the impact of the structure on computational complexity, as well as the consequences of this on the inference of the models.

3.1 The particle cloud representation of a jet

Historically, the first models for jet tagging employed a simple representation at the jet level. The jet was then perceived as a vector of information, where each variable represented a global variable or a specific signature of a constituent [134]. This simplest possible representation was often fed into MLP models that processed the information without extracting any structure from the jet and its constituents. Other models, based on recurrent neural networks [110, 135], recognised a structure in the jet based on the principle of sequence. This principle requires an order among the elements or selecting those meeting quality criteria [135] so that these can be iterated in a sequence in a similar way to the evolution of a time series. The sequential representation of the jet is, therefore, hierarchical and considers that a natural order emerges from the jet's structure, allowing elements to be processed one by one, analogous to a time series or the order of words in the syntax of a sentence. Finally, a simple set representation based on the principle of DeepSets [136] was also developed [137], which, with simple linear layers and sum pooling, produces a network competitive with recurrent networks.

Since the introduction of ParticleNet [129], the concept of the Particle Cloud has become the state-of-the-art representation of jet structure. A Particle Cloud represents the jet as an unordered set of jet constituents. Elements of differing nature, such as charged particles, neutral particles, or SVs, appear in different quantities for each jet. Therefore, the jet can be considered as a set S, following a partition between the elements of each nature. For example,

$S = \{X_1, X_2, X_3\}$ represents our jet such that every element considered is associated with one of the partition subsets X_i. Under this representation, we want to preserve the key property of our unordered set of particles: the conservation of the unordered structure via invariance under the permutation of the jet constituents.

In the context of a tensor representation of the jet constituents consisting of N constituents and feature dimension d, let P denote an $N \times N$ permutation matrix on the constituents and $\mathbb{P}_N$ the set of all $N \times N$ permutations. To discuss permutation invariance, we will need the following definitions:

Definition 3.1.1 (Permutation invariance) *A function $f : \mathbb{R}^{N,d} \to \mathbb{R}^d$, is permutation-invariant if:*

$$f(PX) = f(X),\ \forall\ X \in \mathbb{R}^{N,d}, P \in \mathbb{P}_N$$

Definition 3.1.2 (Permutation equivariance) *A function $g : R^{N,d} \to \mathbb{R}^{N,k}$, is permutation-equivariant if:*

$$g(PX) = Pg(X),\ \forall\ X \in \mathbb{R}^{N,d}, P \in \mathbb{P}_N$$

In our situation, given that we have a set composed of a partition of different types of components, we will require our neural networks to perform an initial embedding step, which consists of a trivial feature engineering aiming to project the input tensor to the feature dimension of the model, that is permutation-equivariant for each partition of the set before aggregating the representations of each partition into a single tensor. This allows us to create a representation of the same dimension between constituents of different natures and to associate them to obtain a tensor representing the complete set of constituents of our jet. This tensor will then be introduced into the core of the neural network. This embedding task is often accomplished by simple layers such as MLPs [138,139] whose equivariance is trivial and well-known [136,140,141].

The scaled dot-product attention mechanism [111] is the key component of the neural network we will develop in this doctoral

research. Being composed of linear transformations and matrix multiplication, this mechanism preserves the equivariance of the representation of our set and is, therefore, a candidate operation to replace the linear layers of ParticleNet. The detailed demonstration of this property for all the blocks composing our two models studied in this thesis is shown in Appendix A. This representation concept was used to build the aforementioned models and relies on constructing a permutation-invariant model with respect to the set of jet constituents taken as inputs. This essential property is contrary to most Transformer models established around the principle of causality. Furthermore, using positional embedding [76, 111] will also be avoided as it introduces an obvious permutation non-invariant function.

We can also draw an analogy between the Particle Cloud and graph theory. We can consider our jet as a graph $G = (N, E)$ where N represents the set of nodes or constituents of the graph, and $E \subseteq N \times N$ represents the edges or properties defining the interconnection between two nodes. In our case, N is equivalent to our set S, and the edges E depend on the model considered. These edges, representing pairwise variables, can be designed based on the architecture and variables considered by the model. Typically, edges are dynamically constructed with k nearest neighbours [129] as illustrated by Figure 3.1, based on the attention score of the model [138, 139], or using inherent kinematic properties [139, 142].

Deep neural networks applied to graphs, also known as Graph Neural Networks (GNNs), share key properties with the Particle Cloud, including permutation invariance over the group or partitions [143]. These common properties, along with the extension of the jet representation to the relations between the constituents, denoted as A, lead to the final construction of the representation considered here. A jet is thus an unordered structure of constituent elements mathematically represented as a graph $G = (N, E)$ where N, the nodes, represent our set S potentially divided into a partition X_i each representing constituents of the same type such as particles carrying electric charge or SVs. Additionally, we can consider additional variables as edges E, capturing pairwise relationships among elements in

S, such as the invariant mass of two jet constituents or their angular sepparation. Our graph can be complete if E contains an edge between every pair of elements, also known as a fully connected graph or incomplete. The extreme case is $E \subseteq \emptyset$, containing no edges as input variables, which is also associated with GNNs constructing edges dynamically from node features N.

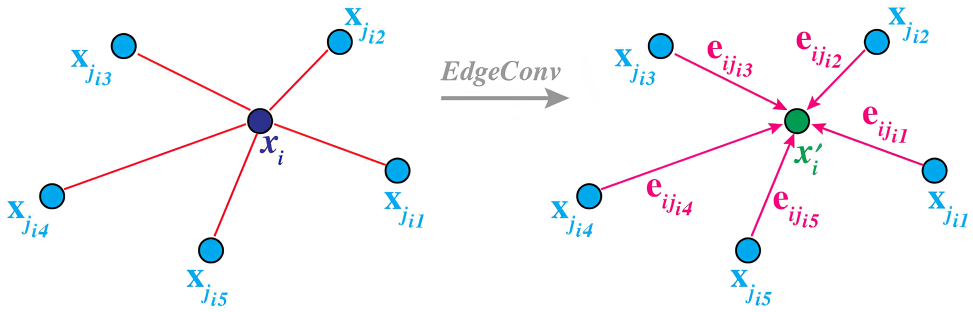

Figure 3.1: Illustration of a Graph Network for the ParticleNet/DGCNN architecture [144]. The output of the EdgeConv layer noted x'_i is obtained by aggregating all the edges, e_{ij}, from each connected node x_i.

3.2 Neural Network architectures for jet algorithms

In this part of the chapter, we will start with briefly introducing earlier algorithms used in CMS in Section 3.2.1 followed by a discussion on newer Transformer models in Section 3.2.2, demonstrating this new family of models preserves the structure representation of the jet while introducing a feature engineering mechanism helping to overcome the limitation of the previous state-of-the-art models.

3.2.1 Related work

The CSVv2 and DeepCSV algorithms

CSVv2 and DeepCSV are enhanced versions of the Combined Secondary Vertex (CSV) algorithm used in heavy-flavour tagging at the CMS experiment [61, 134]. In the updated version CSVv2 [61], the CSV algorithm combines multiple discriminant variables related to SVs and displaced tracks associated with the jet. These variables include information such as the mass and significance of the flight distance in the transverse plane of SVs reconstructed by the inclusive vertex finding algorithm [61], as well as features of significantly displaced tracks like the significance of transverse displacement. Finally, CSVv2 is also the first algorithm to employ class rebalancing via a 2D reweighting of the jet p_{T} and η, allowing the training to be performed on a dataset where classes have the same kinematic distributions and thereby reducing training bias from the data used to train the flavour tagging algorithm.

The CSVv2 algorithm categorises jets into three categories based on available input variables: *RecoVertex* jets containing at least one associated SV, *PseudoVertex* jets lacking SVs but having at least two tracks with significant transverse displacement greater than two and an invariant mass of at least 50 MeV, and *NoVertex* jets not belonging to either of the other two categories. For each category, the available variables vary, necessitating separate training of the CSVv2 algorithm for each category. CSVv2 is the first flavour

tagging algorithm using artificial neural networks at the CMS experiment. It consists of a simple MLP containing one hidden layer with dimensions equal to twice the number of input variables and an output classification layer as illustrated in Figure 3.2. The training of CSVv2 does not perform a single multiple classifications. Instead, the training in each jet category is divided into two stages: one performing binary classification of b jets vs c jets and another of b jets vs light jets (udsg), respectively. A weighting of 1:3 for the discriminants is then applied to obtain the final discriminant.

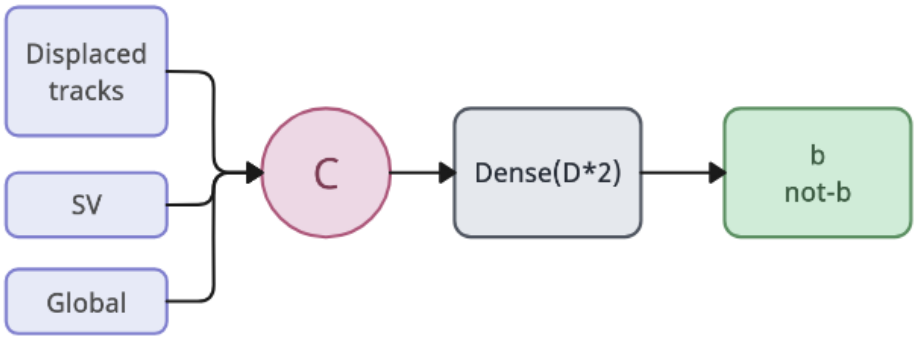

Figure 3.2: Illustration of the CSVv2 algorithm, all inputs are concatenated and fed into a single dense hidden layer before the binary classification layer.

DeepCSV is the evolution of CSVv2 using deep learning. It consists of a MLP neural network with four hidden layers, each containing 100 neurons before the classification layer as illustrated in Figure 3.3. The ReLU activation function [92] is applied between the hidden layers and the softmax function is used in the prediction layer to convert the output values into a probability distribution for each considered class. DeepCSV was originally designed and trained using the Keras/TensorFlow libraries [145, 146]. Unlike CSVv2, DeepCSV is trained across all previously considered jet types, employing a single multi-class classification training to identify jets according to the following definitions: the b class containing a single b hadron, the bb class containing two or more b hadrons, the c class containing a single c hadron and no b hadrons, the cc class containing two or more c hadrons and no b hadrons, and finally the light class for jets not belonging to any of the previous classes.

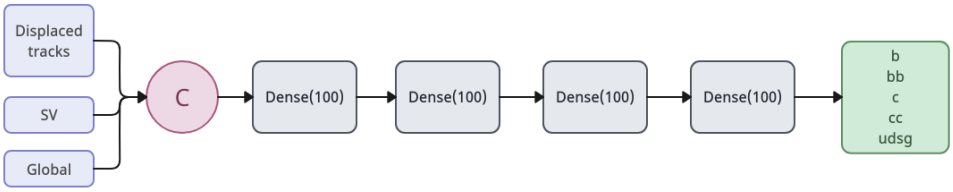

Figure 3.3: Illustration of the DeepCSV algorithm, all inputs are concatenated before being fed into a series of dense hidden layers. The multi-label classification layer then processes the outputs of the last layer.

DeepJet

The DeepJet algorithm [110] represents an advancement over the CSV algorithms. DeepJet utilises a more sophisticated deep network structure, illustrated in Figure 3.4, which includes fully connected layers, convolutional layers, and recurrent layers (LSTM) [110]. This combination allows the model to effectively capture local and sequential dependencies in the data, thus providing better discrimination. Convolutional layers are particularly useful for extracting relevant features from individual constituents, while LSTMs effectively model sequential dependencies between these features for each component category.

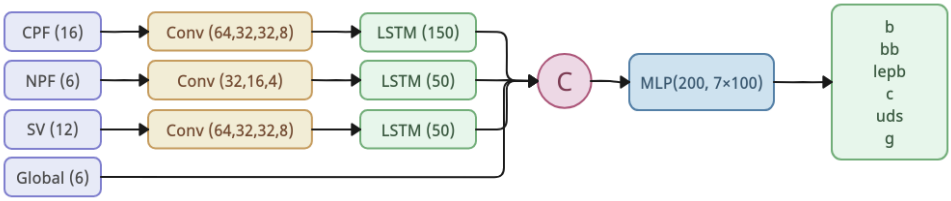

Figure 3.4: Illustration of the DeepJet algorithm, each constituents categories are processed via convolution layers compressing the information before engineering an output vector via LSTM layers independently. The outputs of the LSTM layers, combined with global jet features, are concatenated and fed into MLP layers before classification.

In terms of input variables, DeepJet also advances by focusing primarily on jet constituents, extending the variables to an even broader range compared to DeepCSV [110]. Input variables include detailed information on charged particle-flow candidates (such as track quality, angular distance to the nearest SV, transverse dis-

placement), neutral particle-flow candidates (such as the fraction of jet transverse momentum carried, a neutral photon/hadron discriminator), SVs (such as mass and displacement relative to the primary vertex), as well as global jet features (such as total jet energy and momentum). By considering jet constituents, DeepJet has a wider array of variables to characterise the jet and, in conjunction with a more advanced and deep structure, achieves better performance in jet identification tasks. DeepJet is trained in a multi-classification way to identify jets according to the following definitions: the b class containing a single b hadron, the bb class containing two or more b hadrons, the lepb class containing one b hadron decaying leptonically, the c class containing one or more c hadron and no b hadrons, the uds class for jets containing no associated hadron but an u-d-s parton associated and finally the g class for jets associated with a gluon.

By extending the jet representation to focus on constituents, DeepJet has made a major advancement in jet flavour tagging in the CMS experiment. However, DeepJet independently processes tensors of each constituent type and shares information only after processing each sequence, making it impossible for the network to extract interdependencies between constituents of different types. Moreover, treating variables as sequences processed by LSTMs does not adhere to the Particle Cloud representation and imposes an arbitrary hierarchy among elements. Finally, it is also noted that convolutional layers produce a compressed representation, limiting the neural network's ability to extract maximum discrimination potential on the nature of the jet. Thus, despite significant advances, DeepJet remains a model that does not align with modern deep learning representations and structure for jet algorithms.

ParticleNet

ParticleNet is a graph neural network based on the Dynamic Graph Convolutional Neural Network (DGCNN) structure [144]. It is the original model designed for jet tagging with the Particle Cloud representation [129]. As a graph neural network, ParticleNet employs

edges dynamically created by k nearest neighbours (k-nn) around each constituent. Graph neural networks can be constructed with permutation-equivariant layers applied on a graph. A generic concept of GNN layers defined by Equation (3.1):

$$x'_i = g(x_i, \square_{j=1}^{k} h(x_i, x_{i_j}, A_{i,i_j})), \tag{3.1}$$

where the output of a node i, is labelled as x'_i. g and h represent the permutation-equivariant neural network layers and x_i, x_{i_j} and A_{i,i_j} represent the i^{th} node, the j^{th} neighbour of the i^{th} node and the edge features between them respectively. $\square_{j=1}^{k}$ represents a permutation-invariant aggregation function, also named a pooling function, allowing the network to lower the dimensionality back to the tensor representation of the jet constituents.

In the context of the ParticleNet architecture, after embedding features of different natures with basic MLPs for obtaining tensors with adequate feature dimension, the model concatenates all the tensors to create a single tensor representing the jet as a Particle Cloud. Then, the neighbours are selected via pairwise coordinates. For the first block, η_{ref} and ϕ_{ref}, respectively, the relative pseudorapidity and the relative polar angle to the jet axis are employed. For the next layers, the engineered node features are also used as coordinates for the k-nn. With these coordinates, the network selects the k nearest neighbours for each constituent i, noted $(i_1, ..., i_k)$ and creates a local feature engineering via dynamic edge convolution layer (EdgeConv) [144] as follows:

$$x'_i = \square_{j=1}^{k} h(x_i, x_{i_j}, \theta) = \square_{j=1}^{k} f(x_i, x_{i_j} - x_i, \theta), \tag{3.2}$$

where the output of a node i, labelled as x'_i is obtained by pooling the edge function h, a specific subclass of GNN layers, with an operation $\square_{j=1}^{k}$, [143]. This type of function allows each node of a graph to collect and assimilate information from its neighbours to obtain a new, more refined representation of the node, which has benefited from learning complex relationships between the graph nodes. For the EdgeConv layer, shown in Figure 3.5, the pooling function is

the mean among the k nearest neighbours considered, and the GNN layer is a series of convolution layer, batch normalization layer and the ReLU activation, denoted as the function f in Equation (3.2). The EdgeConv layer combined the information of the input features of node x_i, the difference between the input features of the node and its neighbour $x_{i_j} - x_i$, processed with weight and bias of the convolution kernels θ. To preserve the permutation equivariance of the EdgeConv layer, the kernel size is set to one. The network then undergoes a global average pooling among all the constituents before being fed into a dense hidden layer and a final classification layer.

The global average pooling, taking the mean response over each constituent's features, is trivially permutation-invariant, making ParticleNet the first model to preserve the Particle Cloud representation. It is also the first network using a graph approach, taking the set of constituents as nodes and building k edges for each via the dynamic edge convolution mechanism. For AK4 jet tagging, the ParticleNet architecture was first trained during Run 2 with an MLP embedding layer of dimension 64, three EdgeConv blocks, each having three 1D-convolution layers and considering $k = 8$ nearest neighbours and with a feature size of 64-64-64, 96-96-96, 128-128-128 for respectively the first, second, and last block.

The ParticleNet algorithm identifies the same truth classes as DeepJet, except for the lepb class, which is not considered in ParticleNet and also makes the distinction between the c class containing one c hadron and no b hadrons and the cc class containing two or more c hadrons and no b hadrons.

Extended ParticleNet

For the Run 3 conditions, ParticleNet has been retrained and extended to perform combined heavy-flavour and hadronic τ_h tagging by including the tau truth classes for each decay mode and tau charge, allowing charge assignment and a flavour-aware jet p_{T} correction. The samples utilised have also been extended to include additional sources of τ_h jets as well as improving the p_{T} and η distribution, more particularly enriching the high p_{T} (> 300 GeV) jet population

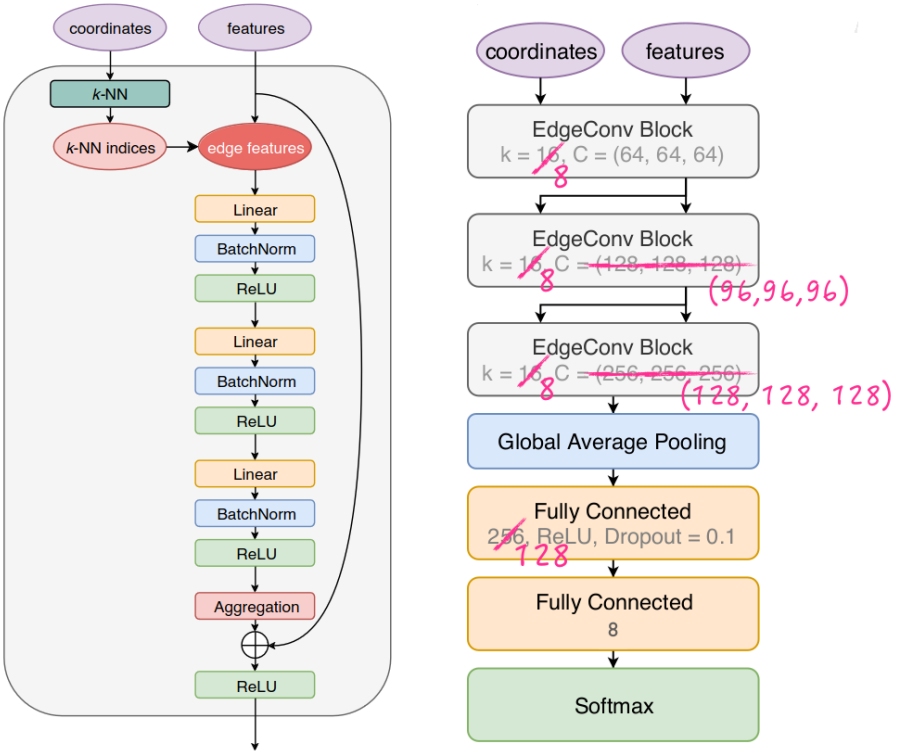

Figure 3.5: Schematic of the EdgeConv block (left) and the Run 2 ParticleNet architecture (right) [129].

to improve the tagging performance in this region. The inclusive loss function is defined as:

$$
\begin{aligned}
L = & CE(y, y_{truth}) + \lambda_1 \cdot \log(\cosh(z - z_{truth})) \\
& + \lambda_2 \cdot (\rho_{16}(z_{16} - z_{truth}) + \rho_{84}(z_{84} - z_{truth})) \\
& with \\
\rho_q(x) = & \begin{cases} qx & \text{if z} > 0 \\ (q-1)x & \text{else} \end{cases}
\end{aligned}
\tag{3.3}
$$

where y_{truth} and y are the truth class and the truth class prediction, and z_{truth} and z are, respectively, the ratio between the generator-level jet p_{T} and the raw jet reco p_{T} before applying JEC

and the prediction of this p_{T} correction ratio. In addition to the regression of the jet energy response, a resolution estimation is provided in the form of quantile regression. Quantile regression consists of estimating the relationship between the regressed quantity and specific quantiles of the regression, providing a more comprehensive view of the conditional distribution than mean regression. The quantile regression of the q quantile is noted as ρ_q, and λ_1 and λ_2 are hyperparameters giving the importance contribution of each loss to the global target. In addition to the usual classification prediction, this new version introduces a regressed correction ratio z serving as a flavour-aware JEC before the JERC and jet energy resolution estimation via the quantile regression terms z_{84} and z_{16}. This approach is undergoing validation, promising better correction, considering the flavour and constituent composition of each jet [147].

The extended model consists of three EdgeConv blocks of larger dimensions (224, 192, 160 each) to perform multiple tasks with extended k-nn considering 16-12-8 nearest neighbours to improve understanding at both a closer and wider neighbourhood extend. The variables generated by each block are concatenated together before using a global average pooling to aggregate the maximum knowledge possible. The obtained context vector is then fed into a series of MLPs of dimensions 192-160-128-96-64 before the final prediction layer of dimension $N_{class} + 3$, performing the classification and the jet energy regression and the two quantile estimations.

3.2.2 Transformer models for jet tagging

Motivation

To develop a new structure, we build upon existing concepts to mathematically constrain the problems we aim to solve and establish a list of existing mechanisms that can serve as a foundation mechanism. The first principle we aim to preserve is the Particle Cloud representation of the jet, which provides the adequate constraints to establish a simple structure adapted to the jet and has successfully provided the jet representation of the current state-of-the-art. Therefore, we seek

models applicable to graphs and invariant under permutation. We require a structure inheriting from Equation (3.1). Secondly, despite its success, we can highlight certain limitations affecting ParticleNet. This model establishes the neighbourhood of each constituent based on the angular coordinate differences or the Euclidean distance of hidden states for deep layers. This *k-nn* calculation is computationally expensive and fixes the number of interactions considered, demanding a balance between the number of neighbours considered in the system and the computational cost we allow. Moreover, as the aggregation function is the mean function, this model does not effectively highlight the importance of certain interrelations, such as those between constituents from the same SV, compared to others. Thus, ParticleNet does not effectively emphasise the interactions between constituents and using a module to construct the neighbourhood can be costly.

We can eliminate recurrent models among existing structures based on the first criterion. Despite their recent resurgence of interest due to the advent of the state-space-models family [148], such as the well-known Mamba [149, 150] and other similar efforts [151, 152], which show promising performances in the field of Large Language Models (LLMs), their recurrence does not respect the representation we desire. Due to the mathematical construction of state spaces models that represent a dynamic system in terms of observed and latent (state) variables, allowing the modelling of a process's 'temporal' evolution using state and observation equations, it would be difficult to create a trivial and efficient permutation-invariant version without having to revisit the entire structure. Similarly, we cannot extend our search to convolutional models [153], as these are not permutation-invariant when considering a kernel size greater than one, resembling a simple MLP on a set, where ParticleNet has already been shown to outperform such models [129]. Finally, we also exclude other GNNs based on convolutions or similar structures, such as Graph Convolutional Networks (GCNs) [154] or Graph Attention Networks (GATs) [155] due to their similarity to the ParticleNet structure and their inability to address its limitations.

In recent years, the development of deep learning has demon-

strated significant success with Transformer models across various tasks ranging from natural language processing [76, 111] to computer vision [75] or fraud detection [156]. Despite their computational complexity, Transformer models, particularly their key mechanism, the self-attention mechanism detailed in Section 2.2.4, benefit greatly from their design, allowing the parallelism of self-attention, unlike their predecessors such as RNNs [109]. Additionally, the advent of increasingly powerful GPUs equipped with Tensor Cores, which accelerate tensor product calculations, has made the Transformer models the most performant algorithms without sacrificing speed. Thus, this family of neural networks represents an ideal field of study that has not been extensively explored in particle physics applications, and the potential for performance gains, combined with the rise of increasingly large and cheap suitable computing power in the form of GPUs [157], could lead to a new class of state-of-the-art deep learning models for jets.

Transformers can be fully connected graph networks

To demonstrate the effectiveness of Transformer models for jets, let us begin by developing the scaled-dot-product attention mechanism and compare it with the GNN layer from Equation (3.1). Starting from the initial MHA Equation (2.14) and isolating the attention score A, we can perform the following development:

$$
\begin{aligned}
Q, K, V &= W^Q x_i, W^K x_i, W^V x_i \\
A &= \text{SoftMax}\left(\frac{QK^T}{\sqrt{d_k}}\right) \\
h_{i,att} = \square_{j=1}^{N} h_i(x_i, x_{i_j}, A_{i,i_j}) &= \text{Attention}(A, V) = AV \\
h_{att} &= \text{Concat}(h_{1,att}, ..., h_{n,att}) \\
g_{att} &= h_{att} W^O \\
x'_i = g_{att}(h_{att}(x_i, x_{i_j}, A_{i,i_j})), g_{att} \quad &= h_{att} W^O
\end{aligned}
\tag{3.4}
$$

The SDPA mechanism can be defined as a permutation-equivariant function $h_{i,att}$, connecting each constituent to all other constituents. This type of function, therefore, considers the jet as a complete graph. The attention score A acts as a tensor on the edges, dynamically constructed by the neural network, and simultaneously contributes as an aggregation function via its matrix product with the tensor V. This allows the neural network to optimise each connection's contribution, hence weighting each pair's contribution in constructing the output variables, highlighting the importance of interrelations crucial for understanding the task, and reducing the contribution of non-relevant pairs. Thus, Transformer models are fully connected graph neural networks. Due to their efficiency on GPUs and the effectiveness of the selection applied by the attention mechanism, Transformer models are excellent candidates to represent a new class of jet algorithm models.

The DeepJet Transformer architecture

The first algorithm based on the Transformer architecture applied to jet tagging is the DeepJet Transformer algorithm [138]. DeepJet Transformer is an evolution of the DeepJet algorithm, abandoning convolutional and recurrent layers in favour of using the SDPA mechanism to construct a permutation-invariant representation of the jet constituents.

The core component of the DeepJet Transformer algorithm, the heavy-flavour Transformer block (HFT), is designed as follows: Initially, the inputs are processed through a basic MLP layer, succeeded by a ReLU activation function. The output of the MLP layer is then fed into a MHA layer, followed by a residual connection and layer normalization. In addition to the MHA layer, a fully connected feed-forward layer, akin to the original Transformer implementation, is incorporated, followed by a final residual connection and layer normalization. To preserve the particle cloud representation of the jet, no positional encoding or causal masking, two usual techniques used in Transformer models [76, 111], are employed.

Usual MLPs are used as embedding layers to represent the same

input feature's dimension for each nature of constituents. The pooling operation of the model is performed via a single-head attention pooling mechanism, designed by the Equation (3.5) between the constituent tensor, noted X and a learnable projection of the constituent tensor via a linear layer into a vector of dimension equal to the number of constituents, noted Z.

$$\text{Attention Pooling}(Z, X) = \text{SoftMax}\left(Z^T\right) X \tag{3.5}$$

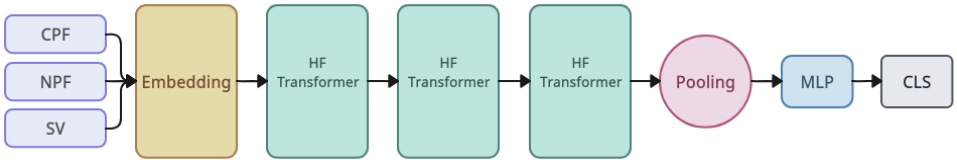

Figure 3.6: Illustration of the DeepJet Transformer algorithm. The algorithm considers charged particle-flow candidates, neutral particle-flow candidates, and SVs as inputs. After the embedding layer, the jet-constituent tensor is fed into the three HFT layers before the attention pooling. A series of MLPs processes the aggregated information before classifying the jet flavour.

The DeepJet Transformer architecture, depicted in Figure 3.6, is designed in the context of AK4 jet classification with the following structure. Initially, features from the jet constituents pass through an embedding layer. This embedding layer comprises an MLP with three linear layers, ReLU activation functions, batch normalization, and dropout. The hidden dimensions of these layers are 64, 128, and 128, respectively, and the dropout rate is set at 0.1. The output tensors from the embedding are then concatenated into a single tensor representing our jet and containing all the constituents. This jet global tensor is subsequently processed through three HFT blocks of dimension 128, each using eight attention heads for processing the attention and using dropout. The representation engineered by our HTF blocks is then aggregated to a jet-level representation through attention pooling and fed to an additional MLP, similar to the embedding block, of feature dimensions 128-128-128 before feeding it into the classification layer. Unlike the FCC-ee version of DeepJet Transformer [138], we chose not to employ jet-level input features

such as kinematic properties of the jet or jet-level features related to the tracks. The ParticleNet architecture employs only jet-constituent variables, and by excluding jet-level inputs, we ensure the model comparison remains unbiased by additional information in favour of the DeepJet Transformer architecture. The DeepJet Transformer algorithm uses the same truth flavour targets as DeepJet for heavy-flavour tagging.

The Particle Transformer architecture

The Particle Transformer [139] architecture (ParT) can be seen as an evolution of the DeepJet Transformer, mainly improving the performance via additional kinematic pairwise features [139, 142, 158]:

$$\begin{aligned} \Delta &= \sqrt{(y_i - y_j)^2 - (\phi_i - \phi_j)^2} \\ k_{\mathrm{T}} &= \min(p_{\mathrm{T},i}, p_{\mathrm{T},j})\Delta \\ z &= \frac{\min(p_{\mathrm{T},i}, p_{\mathrm{T},j})}{p_{\mathrm{T},i}, p_{\mathrm{T},j}} \\ m^2 &= (E_i + E_j)^2 - |\overrightarrow{p_i} + \overrightarrow{p_j}|^2 \end{aligned} \tag{3.6}$$

where the pairwise features between two constituents of subscript i and j are built via their four-momentum $p = (E, p_x, p_y, p_z)$, their momentum $\overrightarrow{p}$, rapidity y, and azimuthal angle ϕ. These new pairwise features, introduced by Equation (3.6), are preprocessed via a logarithm rescaling ($\ln\Delta$, $\ln k_{\mathrm{T}}$, $\ln z$, $\ln m^2$) and then fed into MLP layers producing a new edge tensor U which will be used in the new attention mechanism named Particle Multi-Head Attention ($P\text{-}MHA$), defined by Equation (3.7).

$$P\text{-}MHA(Q, K, V, U) = \mathrm{SoftMax}\left(\frac{QK^T}{\sqrt{d_k}} + U\right)V \tag{3.7}$$

The $P\text{-}MHA$ can be seen as a modification of the original attention mechanism used in Transformer models, including an additional edge tensor U, based on kinematic pairwise features. This new mechanism, aware of additional information about the interrelation inside

all pairs of constituents, allows the model to engineer better attention scores. Similar approaches were also studied in general GNN research leading to the Graph Transformer subclass [159].

The embedding of ParT is nearly identical to the previous embedding, with only the activation function now being the GELU [93] function. Similarly, the Transformer block of ParT, also named the Particle Attention block, is similar to the HFT block, only dropping the initial MLP and including the new edge tensor U as an input for the P-MHA layer. For aggregation, ParT uses a class token vector x_{cls} similar to the CaiT approach [160] along with the usual MHA as follows:

$$\text{Attention}(Q_{CLS}, K, V) = \text{SoftMax}\left(\frac{Q_{CLS}K^T}{\sqrt{d_k}}\right)V \tag{3.8}$$

where Q_{CLS} represents the vector class token used as a query and K, V are the usual key and value built from the jet constituent tensor concatenated with the class token. This class attention mechanism is embed into a Class Attention block in a similar way to the Particle Attention block, as illustrated in Figure 3.7.

For AK4 jet classification, the ParT architecture consists of an embedding block consisting of three layers of feature dimensions 128-128-128, three Particle Attention blocks of feature dimension 128 and eight attention heads and one Class Attention block of the same feature dimension and attention head. We have thus described all the architectures discussed and developed during this thesis. The demonstration of the permutation invariance of DeepJet Transformer and ParT is available in Appendix A.

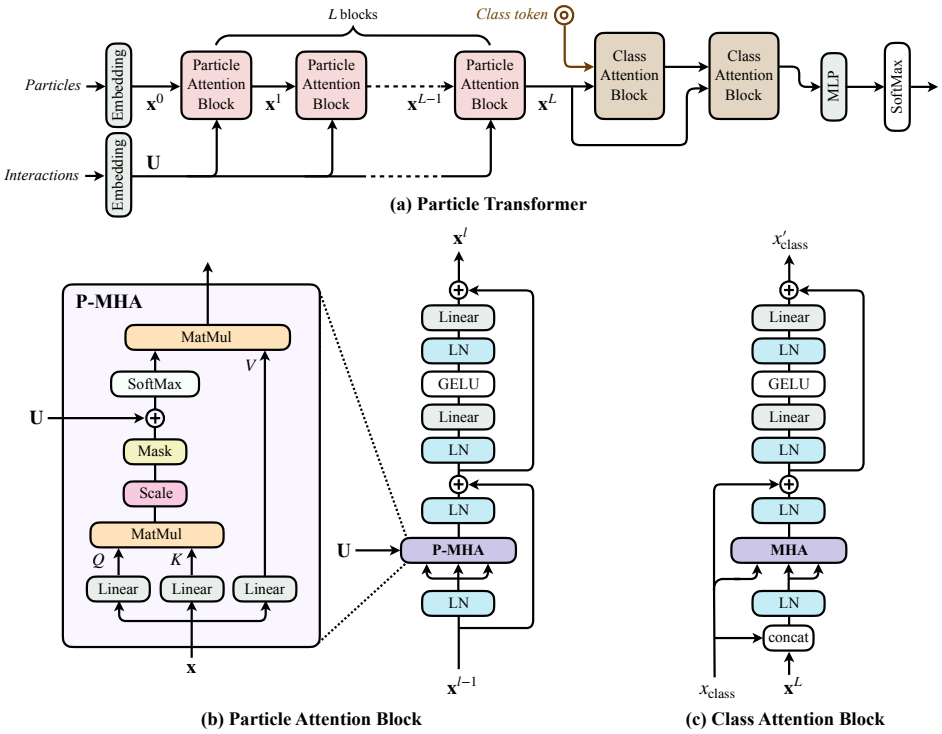

Figure 3.7: Illustration of the ParT model (a), the P-MHA and Particle attention block (b) and the class attention block (c) [139]

3.3 Training samples for jet tagging

This passage outlines the necessary elements for training our algorithms. In Section 3.3.1, we will detail the dataset used, how we obtain the jet labels, and the tools employed for training, from preprocessing ROOT files to performance evaluation. Section 3.3.2 explains how we preprocess our dataset and the input features used. Finally, Section 3.3.3 describes the training procedure employed and the exact models trained in the context of this chapter.

3.3.1 Datasets and labelling

The training, validation and test datasets consist of simulated anti-k_T jets [66], with $R = 0.4$. The simulated jet sources are binned QCD multijet and semileptonic $t\bar{t}$ events produced with PYTHIA8 [25] and

POWHEGv2 [161] respectively. A gluon reduction of 50% is applied on the QCD multijet events to avoid class imbalance. PYTHIA8 processes hadronization and showering of our samples. GEANT4 [162] is used to simulate the response of the CMS detector [163, 164]. The PUPPI algorithm handles pileup mitigation [67]. The jet constituents are reconstructed using the particle-flow algorithm [165]. To prevent any risk of overfitting, our samples for training and validation are also processed via a merging scheme, producing merged files containing 400,000 jets each. A detailed table of the samples used is available in Appendix C under the label 'Training 2023'.

Deep Learning algorithms have been introduced at different times and by several groups. They have never been trained under the same conditions. This has prevented a fair comparison of the performance of each algorithm for AK4 jet flavour identification. Therefore, we propose an identical training for all algorithms based on jet constituents to enable this comparison and to measure the impact of different architectures on performance and a benchmark of their inference and training cost. We exclude DeepCSV and CSVv2 from this comparison as their maintenance has been discarded for the CMS Run 3 data taking, and they are simple MLP models based on jet level inputs and, for DeepCSV, a limited amount of tracks, known from a while to be underperforming [61, 110].

The samples used are processed using the updated versions of DeepNTuples [166] as well as the b-hive framework [167]. DeepNTuples is a ntupler framework that produces flat jet ROOT [168] files, containing the variables and labels necessary for training as well as analysing the performance of our algorithm, by using MiniAOD files [169] as input. The b-hive framework, released in September 2023 by the CMS experiment, is a modern framework for deep learning applications in particle physics. It particularly has been designed for jet identification tasks at the CMS experiment in the context of the research detailed in this chapter and the following ones. Unlike its predecessor, the DeepJet/DeepJetCore framework [170, 171], b-hive is a polyvalent tool built around workflow tasks via the law library [172] and state-of-the-art Python libraries for processing any

flat ROOT files such as DeepNTuples or NanoAOD [173] files. We use the integrated data workflows to preprocess and build our dataset. Designed around the PyTorch library [174, 175], we use this framework for its modern dataloader and state-of-the-art training environment. Beyond its increased versatility, b-hive has also demonstrated its usefulness thanks to its robustness and a significant improvement in training speed, up to a +243% speed-up for Particle Transformer, as illustrated in Figure 3.8.

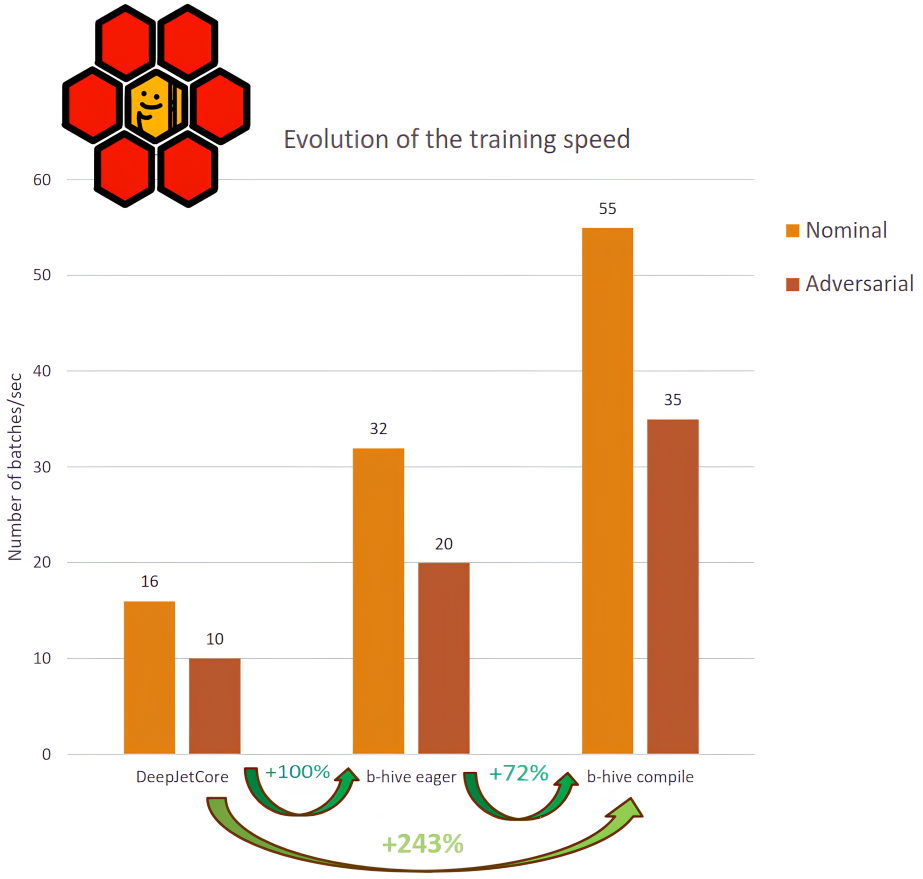

Figure 3.8: Training speed comparison of the ParT architecture with a batch size value of 512. The b-hive framework already shows a 100% training speed improvement compared to DeepJet/DeepJetCore [170, 171] thanks to the new data structure and fast I/O compression algorithm. When using the new PyTorch GPU compilation [175], another 72% training speed improvement is obtained, leading to a total 243% speed-up compared to the previous framework.

Labelling

Jet labels are obtained via ghost association [61, 176]. The last iterations of each b and c hadron before decaying have their momentum scaled to a small value of 10^{-18} GeV, to transfer only the directional information of these heavy-flavour hadrons during the clustering of the generator-level jet (gen jet). These scaled-generated hadrons, also called *ghost* hadrons, are included among the particles clustered by the jet clustering algorithm in the reconstructed jets of our simulated data, allowing us to include information about heavy-quark physics without modifying the kinematic properties of the gen jets. We similarly treat the partons' information to obtain *ghost* partons. Our six labels are obtained via the following decisions:

- Jets containing at least one b hadron are labelled as b jet.
- b jets are split into three subcategories: b for jets containing a single b *ghost* hadron, bb for jets containing at least 2 b *ghost* hadrons and b_{lep} for single b *ghost* hadron decaying leptonically.
- Jets containing no b hadrons and at least one c *ghost* hadron are labelled as c jet.
- Jets containing no b or c *ghost* hadrons, also named *light* jets in opposition to the heavy-flavours b and c, are split into the light quark label (uds) and gluon label (g). The flavour assigned to those light jets is obtained via the leading p_{T}, also called the hardest, light-flavour *ghost* parton associated with it via our clustering algorithm.
- If a jet contains no *ghost* hadron or parton, its flavour is undefined and the jet is discarded.

Additionally, we apply an additional lepton veto. From the list of generator-level leptons originating from a resonance such as the decay of a Z, W, or H boson, for example, any reconstructed-level jet having an angular distance $\Delta R < 0.2$ with one of these leptons is removed from the training. This additional cleaning is necessary to avoid mislabelling the jet flavour and assigning a b/c/light label to a jet originating for a lepton clustered by our clustering algorithm.

3.3.2 Input features and preprocessing

Heavy-flavour tagging in high-energy physics involves the identification of signatures coming from b (c) hadrons contained in our jets. Notably, b (c) hadrons have a longer lifetime than lighter ones due to the weak decay associated with the b quark. The lifetime of a b (c) hadron is of the order of 1.5 (0.5-1) 10^{-12} seconds, resulting in a displacement of the order of mm to cm to the primary vertex of the jet before the hadron decays [61]. This displacement allows for identifying the displaced tracks originating from the decay and reconstructing the associated SV. The impact parameter of the tracks associated with a jet, which quantifies the distance of closest approach of a track to the primary vertex, is a key variable for identifying tracks originating from an SV. A reconstructed SV within a jet, formed by the displaced tracks from the primary vertex, is another strong indicator of a b (c) jet. This indicator is refined using variables associated with the SV of b (c) hadrons, such as the corrected SV mass M_{SV}, correcting the invariant mass of the SVs, taking in account the particles not associated with the SVs or not reconstructed, or the flight distance and significance between the SV and the primary vertex [61]. Finally, the identification of a soft lepton, originating from the leptonic decay of a b (c) hadron, corresponding to $\sim$20% ($\sim$10%) of decays, is also a signature that can be exploited for the identification of b (c) jets. Most of the variables used for heavy-flavour tagging are constructed and refined around these properties. Figure 3.9 illustrates some of the properties of the displaced tracks and secondary vertices mentioned.

In this work, our inputs consist of the jet's constituents. They are split into three categories: the charged particle-flow candidates (CPFs), the neutral particle-flow candidates (NPFs) and the secondary vertices (SVs). Except for the 4-momentum of the jet-constituents categories necessary for building the relevant pairwise features of the Particle Transformer architecture, the features used for the training are identical to the DeepJet training during Run 2 [110]. A detailed list of the simulated event samples used for this training is available in Appendix B.

The input tensors are constructed via the b-hive framework, processed by the coffea library [177]. We consider up to 26 CPFs, 25 NPFs, and 5 SVs, obtaining a constituent count that is a multiple of 8 to optimise training speed [175]. When a jet consists of fewer constituents than the maximum values, we use zero padding, a method consisting of filling the entries beyond the number of constituents with zero values, to fill in missing entries. If there are more constituents than our fixed maximum, we eliminate candidates with the lowest transverse impact parameter significance for CPFs and the lowest transverse momentum for NPFs and SVs. The NumPy libraries process the resulting data [178] before being stored using the LZ4 compression algorithm [179], which allows for lossless compression with fast compression and decompression speeds suitable for deep learning training standards. The PyTorch library handles the training pipeline management and training itself [174, 175], enabling the design of an efficient and optimised dataloader for reading and transferring data to GPUs.

For generalization purposes of the model, a 2D reweighting by jet η and jet p_{T} is carried out to avoid any dependency of the training on these two variables in the events used for training. During this reweighting, the 2D spectrum of each truth flavour is reshaped to obtain the same normalised distribution as the one of the b jet truth flavour. After applying the reweighting to our merged samples, we obtain the dataset used for training. Our dataset consists of approximately 50M jets, with an 85% to 15% split between training and validation. A test set consisting of 5M jet of the same composition as the training and validation sets is used for the final evaluation of the models. Figure 3.10 illustrates the effect of the reweighting method on the involved kinematic properties of jets.

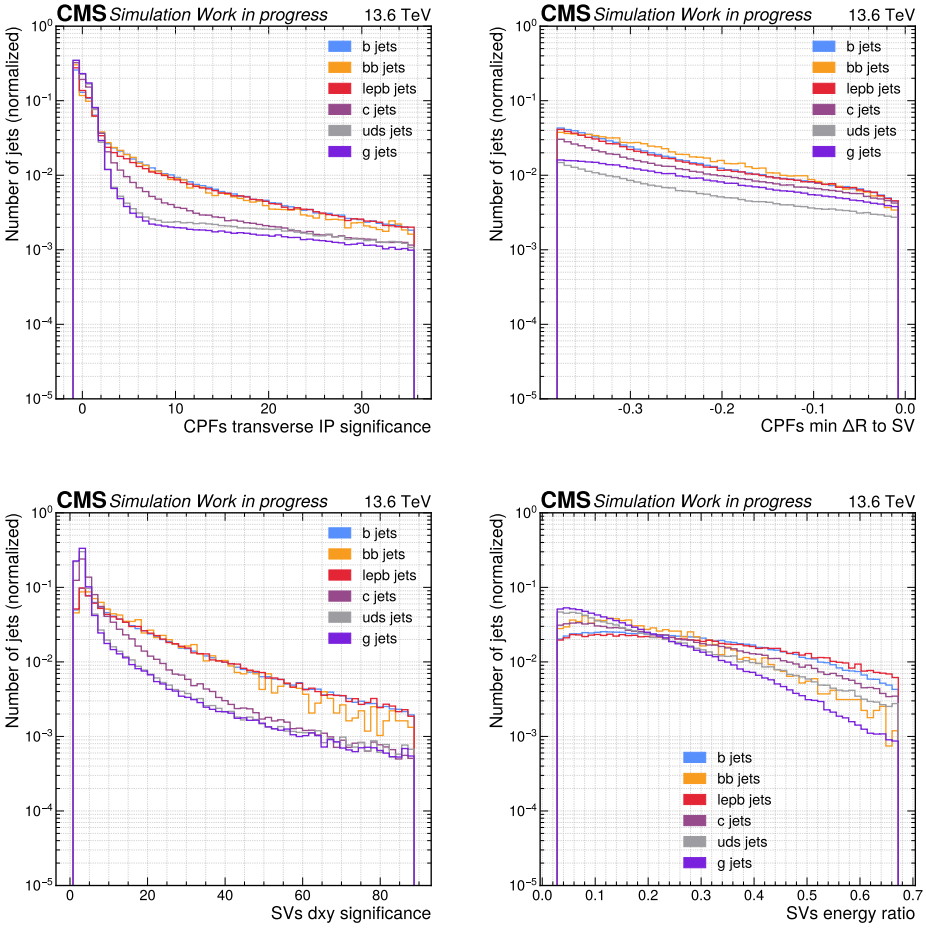

Figure 3.9: Distribution of the 2D impact parameter of the CPFs (upper left), the ΔR between the CPFs and closest SV (upper right), translated by a value -0.4, the SV transverse flight distance (lower left) and the ratio between the SV energy and the jet energy, after the reweighting.

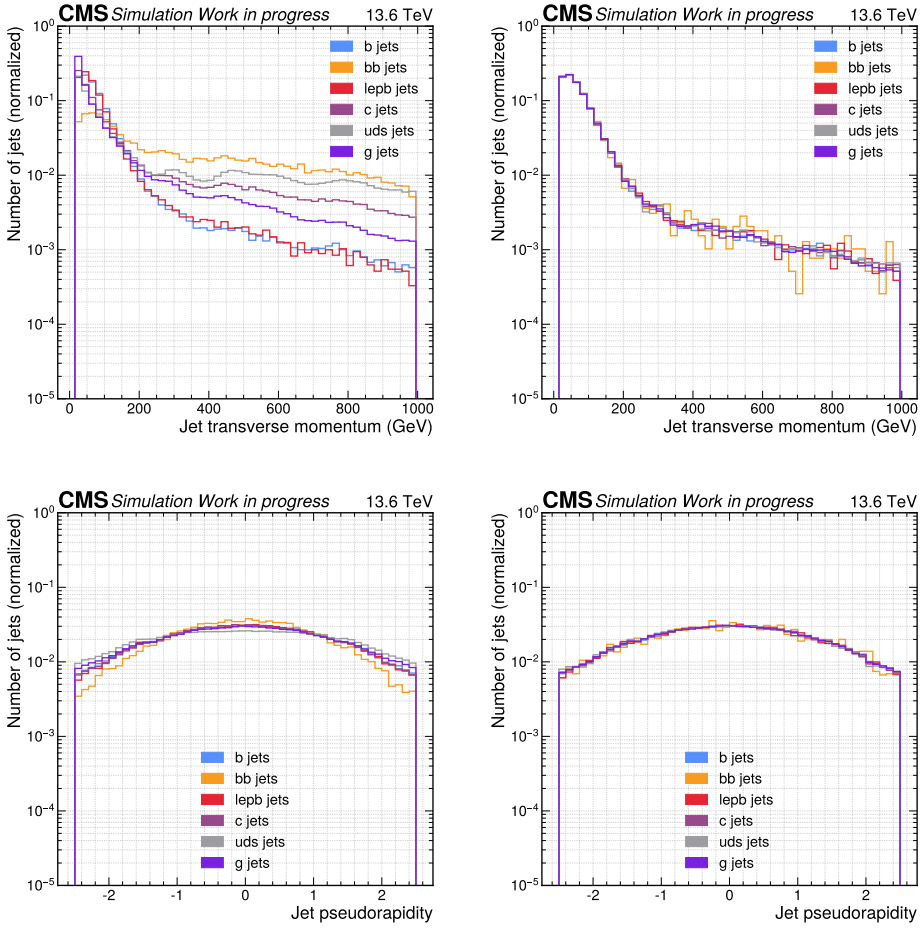

Figure 3.10: Distributions of the p_{T} (upper) and η (lower) of the jets for the different classes before (left) and after (right) applying the reweighting. We can observe that the kinematic properties of the jets have a similar distribution after reweighting, as expected.

3.3.3 Training procedure

The PyTorch library version 2.2.1 [175] was used as the deep learning framework in this study for constructing the neural network model and the training process. We leveraged the latest advancements in learning to optimise GPU usage, including FP16 precision and compiling gradients into optimised kernels for the Tensor Cores of the used NVidia A100 GPUs [180]. The applied optimizer is the Lookahead optimizer [132], with hyperparameters $k = 6$ and $\alpha = 0.5$, and RAdam [130] as the base optimizer with an initial learning rate of 1×10^{-3}, decay rates (β_1, β_2) set to (0.95, 0.999), and a decoupled weight decay [131] of 0.01. The training was conducted over 30 epochs with a batch size of 512, accompanied by a learning rate scheduler. The scheduler used is the cosine scheduler [181] with a linear warmup during the first 5% of the training steps, a minimum rate of 1×10^{-5} and a single cosine cycle. For all algorithms, a snapshot of the model is saved at the end of every epoch and the model with the best validation loss is kept as the final model version. The hyperparameters mentioned here are adjusted from the default values recommended by the authors of the associated components. For the parameters where training performance showed sensitivity, we fine-tuned them through a grid search.

The trained models are DeepJet, ParticleNet, DeepJet Transformer, and Particle Transformer. The DeepJet model is trained with its default parameter sizes [110] to preserve its data compression properties through convolutional layers. It will serve as a baseline for performance comparison. For the other models, a first and main training is done to make the comparison as equal as possible. We set the feature dimensions to 128, the same embedding size based on MLPs. Each model comprises three layers of its main block, of dimension 128, followed by an aggregation in a single operation. For ParticleNet, we consider $k = 8$ neighbours as suggested by the authors in their applications to resolved jets at the CMS experiment. For the Transformer models, we use $h = 8$ head dimensions, the default value for most medium-sized Transformer applications [75, 76, 138, 139].

Also, in order to evaluate the scaling of the performance with the computation complexity of the architecture, we also trained ParticleNet, DeepJet Transformer and Particle Transformer with six layers and the same hyperparameters as above for obtaining the performance with larger models and a smaller training with three layers and 96 as the feature dimension for obtaining the performance of a smaller model.

Finally, we also evaluate the impact of the permutation invariance of the Transformer models for jet tagging by training three variants of DeepJet Transformer, one using the causal mask, a second using a state-of-the-art positional embedding, the Rotary Position Embedding (ROPE) [182] and a last one combining both. We will use it to evaluate the impact of a permutation-invariant architecture on the performance as well as the robustness of the model against random permutation.

3.4 Flavour tagging performances and model complexity

In this part of the chapter, we will evaluate the performance of our algorithms from the perspective of flavour identification in Section 3.4.1 and from the perspective of computational complexity in Section 3.4.2.

3.4.1 Heavy-flavour tagging performance

The b-tagging performance of the different models is evaluated via the following discriminator $prob_{b\ vs\ all} = prob_b + prob_{bb} + prob_{lepb}$. Since the c jets represent an intermediate regime between the b and light jets, we separate their rejections to better understand our algorithms' rejection capacity for these two cases. To obtain a view of the rejection capacity independent of any probability threshold, we use the ROC curve of the b-tagging performance of our algorithms as shown in Figure 3.11.

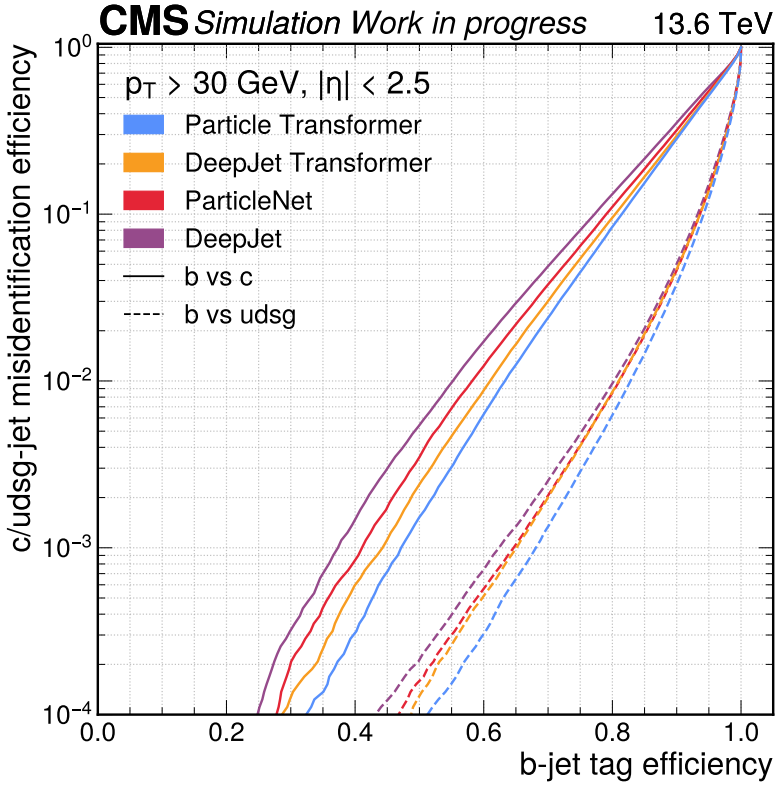

Figure 3.11: ROC curves of the b-tagging performance of DeepJet (purple), ParticleNet (red), DeepJet Transformer (orange) and Particle Transformer (blue). The dashed line represents the udsg jet rejection, while the solid line represents the c jet rejection.

First, we can observe that the ROC curves have similar profiles. We do not encounter models that perform better or worse depending on the chosen misidentification rate, whether for c or udsg rejection. This behaviour is expected from training under the same conditions. We can observe that DeepJet, our baseline model, is, as expected, the least performant model regarding both c and udsg rejection. ParticleNet, the state-of-the-art, exhibits a similar rejection performance for udsg jets than DeepJet Transformer, only falling behind at low misidentification rates ($< 0.01\%$). Still, DeepJet Transformer performs better for c rejection, demonstrating that a Transformer architecture achieves new state-of-the-art performance, highlighting the

potential of this type of architecture. Finally, Particle Transformer, by using new pairwise variables as additional edge features, further improves performance compared to DeepJet Transformer. This performance improvement demonstrates the usefulness of such variables and that Transformer models can be excellent Graph Transformer models for jet algorithms. Tables 3.1 and 3.2 provide the b-tagging efficiency for the loose, medium and tight working points (WPs) usually employed in CMS analyses, respectively 10/1/0.1% misidentification for the b vs c and b vs udsg rejection separately.

Model	Loose WP eff	Medium WP eff	Tight WP eff
DeepJet	77.24%	55.20%	37.33%
ParticleNet	79.05%	58.20%	41.07%
DeepJet Transformer	80.45%	61.00%	44.27%
Particle Transformer	81.15%	63.31%	47.15%

Table 3.1: b vs c efficiency at the loose, medium and tight WP.

Model	Loose WP eff	Medium WP eff	Tight WP eff
DeepJet	93.26%	80.24%	62.38%
ParticleNet	93.60%	81.08%	64.65%
DeepJet Transformer	93.62%	80.99%	65.11%
Particle Transformer	94.18%	82.82%	68.11%

Table 3.2: b vs udsg efficiency at the loose, medium and tight WP.

In the context of c-tagging performance, we use two different discriminators to maximise the discrimination power of our algorithms, described by Equation (3.9). Similarly to the b-tagging performance, we show in Figure 3.12 the c-tagging ROC curves, splitting the misidentification between the b rejection and the udsg rejection.

$$\begin{aligned} prob_{c\ vs\ b} &= \frac{prob_c}{prob_c + prob_b + prob_{bb} + prob_{lepb}} \\ prob_{c\ vs\ udsg} &= \frac{prob_c}{prob_c + prob_{uds} + prob_g} \end{aligned} \tag{3.9}$$

For c-tagging performance, we can observe that, similarly to b-tagging, DeepJet is the least performant model in terms of b and udsg rejection. ParticleNet, the state-of-the-art model, exhibits lower rejection performance than DeepJet Transformer for both b and udsg

rejection. Finally, Particle Transformer further improves the performance of DeepJet Transformer, surpassing the b and udsg rejection performance. These results demonstrate the utility of Transformer models for jet algorithms. Tables 3.3 and 3.4 provide the c-tagging efficiency for loose, medium, and tight WP.

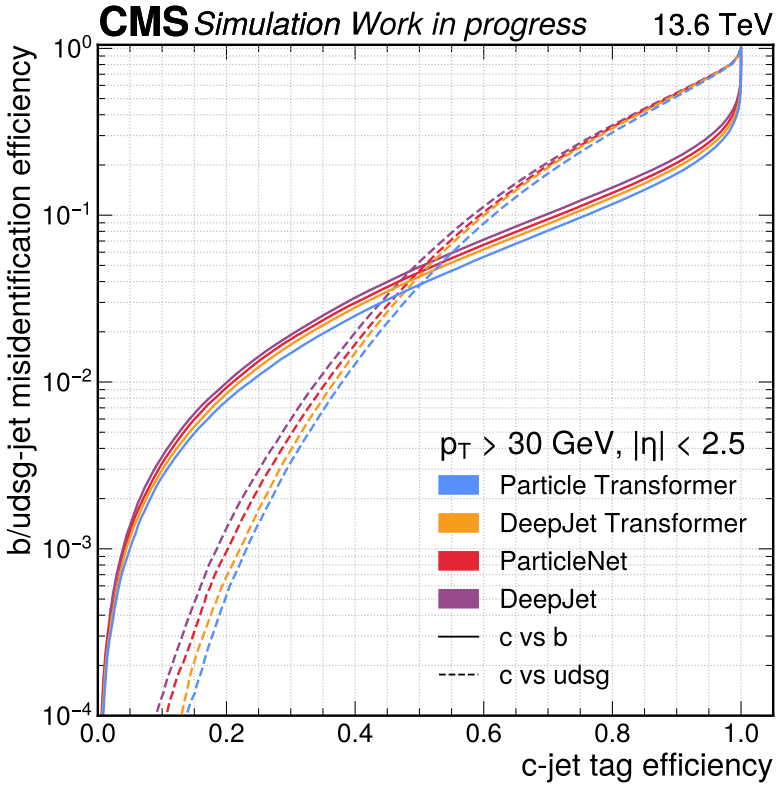

Figure 3.12: ROC curves of the c-tagging performance of DeepJet (purple), ParticleNet (red), DeepJet Transformer (orange) and Particle Transformer (blue). The dashed line represents the udsg rejection, while the solid line represents the b rejection.

Model	Loose WP eff	Medium WP eff	Tight WP eff
DeepJet	69.46%	20.10%	3.92%
ParticleNet	71.60%	21.01%	4.23%
DeepJet Transformer	73.25%	22.04%	4.41%
Particle Transformer	75.93%	23.66%	4.98%

Table 3.3: c vs b efficiency at the loose, medium and tight WP.

Model	Loose WP eff	Medium WP eff	Tight WP eff
DeepJet	58.26%	34.03%	18.51%
ParticleNet	59.41%	35.56%	20.22%
DeepJet Transformer	60.04%	36.79%	22.11%
Particle Transformer	61.51%	37.99%	23.11%

Table 3.4: c vs udsg efficiency at the loose, medium and tight WP.

3.4.2 Flavour-tagging complexity

Another important aspect of using any ML tool is the computational complexity and the prediction time. Indeed, the creation and development of complex models come with a computational cost that can be significant. To evaluate this, we evaluate the inference cost of the algorithms studied here. Using an environment replicating the conditions of CMS software [64] (CMSSW), the software environment in which the algorithms will be used, we measure the computational complexity expressed in floating-point operations per second (FLOPs), the number of parameters (weights and biases), and the prediction speed in iterations per second (it/s) employed by each model. These values give us a general indication of how much computation a model requires, how many neurons characterize the model size and how fast the model can make predictions. Indeed, FLOPs and prediction speed are excellent indicators of inference performance, allowing us to measure both how costly the operations of a model are and how optimised they are on the hardware and software used. The number of parameters is another important factor, representing the model's parametrization capability at a chosen size.

The environment consists of a computing node of an Intel(R) Xeon(R) CPU@2.20GHz, using two cores combined with 32 GB of RAM. We use the ONNXRuntime [183, 184] library distribution for optimising the inference on CPU after using the integrated PyTorch

to ONNX model converter available in the PyTorch distribution [174]. We predict the jet probability with 16 constituents (eleven CPFs, four NPFs and one SV), representing the average number of constituents our training set had. This number is motivated to reflect what our training dataset illustrates and to have a reference for measuring complexity. When applying the algorithms to the data, the number of constituents will vary, and depending on the phase space or the energy of the jets considered, we will have to deal with different numbers of constituents. This measure is, therefore, qualitative and is only intended to highlight the qualities or defects of the different architectures without reflecting a complete measure of the reality of CMSSW, something impossible to do at present with the available tools.

The results displayed in Table 3.5 indicate that our baseline DeepJet is the simplest and fastest model, as expected. It is the only model that compresses the data and processes of each constituent type independently, leading to a model complexity of 2.60 MFLOPs for 266.292 parameters. The measurements confirm that its application aligns with its design objectives. Next, we can observe that ParticleNet, despite having fewer than half the number of parameters of Particle Transformer, 403.018 instead of 1.056.309, incurs a significantly higher computational cost of 56.41 MFLOPs instead of 23.06 MFLOPs for Particle Transformer, an increase of $\sim 145\%$. DeepJet Transformer has fewer parameters than Particle Transformer for the same size of a model with 871.243 parameters. This mostly comes from the absence of pairwise features and the usage of a simple attention pooling instead of the classification token [160]. The impact is also noticeable on the computation complexity. DeepJet Transformer has a computation cost of 19.73 MFLOPs, reducing the cost by $\sim 14\%$ compared with Particle Transformer. This cost reduction represents the impact of using pairwise features.

This is due to the complexity of the *k-nn* operations, which first require the calculation of the pairwise distance with a complexity of $\mathcal{O}(N^2)$, where N is the number of constituents, followed by the selection of the k nearest neighbours with a complexity of $\mathcal{O}(N + k \log k)$. The cost of convolutions is also particularly high, with a convolu-

tion of input dimension d and identical output dimension having a complexity of $\mathcal{O}(Nkd^2)$. On the other hand, the primary complexity of our Transformer models is their attention mechanism. This consists of linear layers with a complexity of $\mathcal{O}(Nd^2)$ followed by the SDPA mechanism with a complexity of $\mathcal{O}(N^2d)$ [111]. The aggregation mechanism of both Transformer models is of the order $\mathcal{O}(Nd)$. At the same time, Particle Transformer additionally includes a unique computational cost due to the calculation of pairwise features with a complexity of $\mathcal{O}(N(N-1)/2)$ followed by linear layers of order $\mathcal{O}(N(N-1)h^2)$, where h is the number of attention heads, which is eight in our case. Thus, the cost of an EdgeConv layer of ParticleNet, with the same dimension on the variables and the same number of convolutions as our Transformer models with linear layers for identical depth, is significantly higher as expected from the complexity estimates, and this is reflected in the overall computational complexity and model inference. These measurements highlight another important advantage of Transformer models. Not only do they perform better, but they also have a lower inference cost, reducing the computational expense flavour-tagging algorithms produce.

However, we can observe that despite the very high computational complexity, the gap narrows when we measure the inference on the CPU. This is due to two factors. First, the complexity of the SDPA operation remains very high and unoptimised for computations outside of GPUs. Second, our Particle Transformer model converted to ONNX must use custom and suboptimal operations to create pairwise features because ONNX does not integrate efficient versions. These two suboptimal elements thus impact the inference of DeepJet Transformer and Particle Transformer, which remains faster than ParticleNet.

The training of the DeepJet Transformer models, including the non-permutation invariant components, have been evaluated with the default ordering of the jet constituents in Figure 3.13. We can observe the performances of the algorithms are similar, indicating the impact of the causal mask and ROPE [182] components are not impacting the tagging performance when both the training and test datasets follow the same ordering. However, when applying random

Model	Num. params	Forward	CPU Inference
DeepJet	266,292	2.60 MFLOPs	2492.38 ± 1.79 it/s
ParticleNet	403,018	56.41 MFLOPs	552.56 ± 0.37 it/s
DeepJet Transformer	871,243	19.73 MFLOPs	961.58 ± 0.94 it/s
Particle Transformer	1,056,309	23.06 MFLOPs	635.88 ± 0.70 it/s

Table 3.5: Model complexity comparison table. The forward computation complexity and CPU inference speed are measured with a baseline resolved jet of 16 constituents. The number of parameters corresponds to the number of weights and biases each model contains.

permutation in the jet constituent order, the performance of the non-permutation invariant models shows degradation. In contrast, the predictions of the permutation invariant DeepJet Transformer model remain unchanged as expected. Particularly, we can see in Figure 3.14 that the causal mask, in red, induced a more significant degradation than the ROPE component, in orange. Combining the two components, in purple, displays a larger performance degradation, highlighting this model's larger permutation sensitivity. This performance degradation illustrates the necessity of building permutation invariant models, ensuring the model's performance is not degraded when we do not use the non-permutation invariant components. The permutation invariance also guarantees the robustness of the model against any permutation that may occur, for example, by a change in the reconstruction of the jet or a software modification.

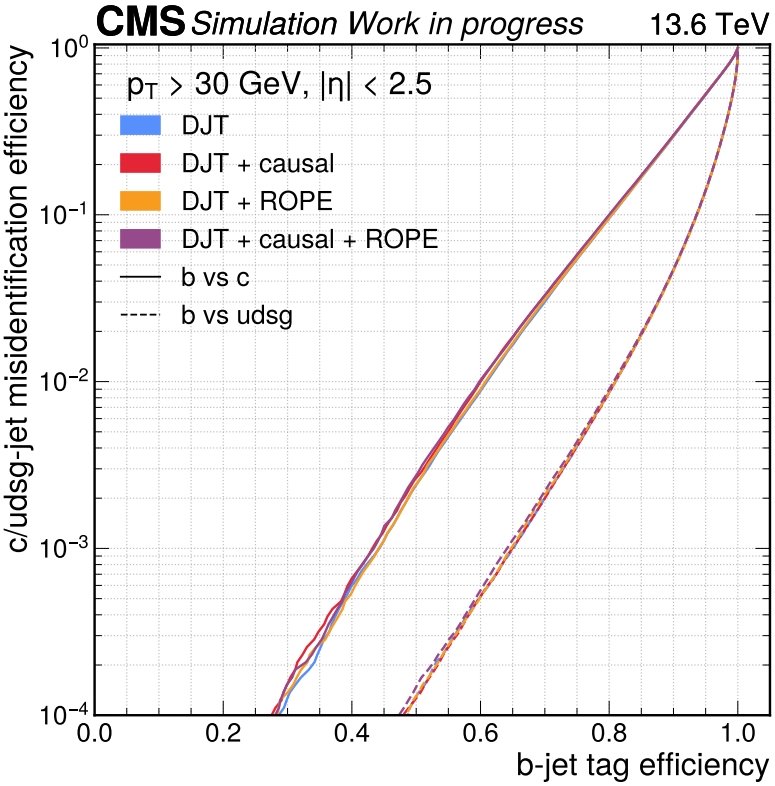

Figure 3.13: ROC curves performance of DeepJet Transformer (blue) and its extension with non-permutation invariant component: causal (red), ROPE (orange) and causal + ROPE (purple). The dashed line represents the udsg jet rejection, while the solid line represents the c jet rejection. The performances are obtained on jets with the default constituent's ordering.

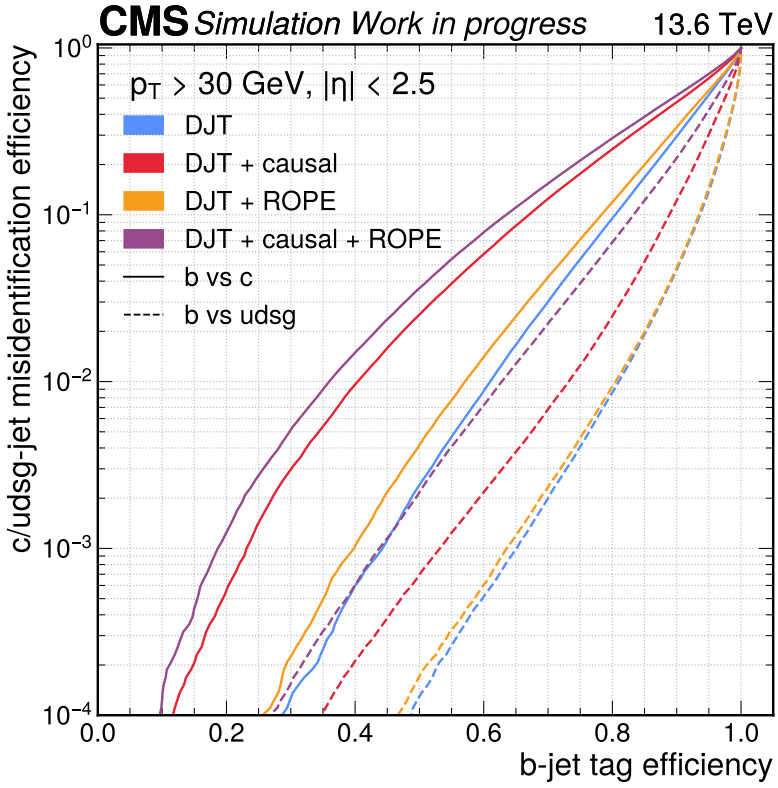

Figure 3.14: ROC curves performance of DeepJet Transformer (blue) and its extension with non-permutation invariant component: causal (red), ROPE (orange) and causal + ROPE (purple). The dashed line represents the udsg jet rejection, while the solid line represents the c jet rejection. The performances are obtained on jets with random constituent's ordering.

The results of training smaller and larger models are combined with the original models to obtain a comparison of the b-jet efficiency percentage at the medium WP, denoted as b_{eff} @ medium WP. The results are compiled in Figure 3.15, where we can observe that the computational complexity of ParticleNet increases significantly with model size due to its initial complexity. In contrast, the Transformer models benefit from lower complexity due to their structure. We also notice that Transformer models seem to exhibit better scaling in performance. The larger versions of DeepJet Transformer and Particle Transformer, with 6 layers and 128 feature dimensions, achieved better performance than the smaller models, whereas ParticleNet did not. The benefits of better performance scaling combined with lower complexity, which also results in shorter inference times, provide additional arguments for using Transformer models for heavy-flavour tagging. The difference in computational complexity and inference time observed between DeepJet Transformer and Particle Transformer is smaller than ParticleNet. The significant performance improvement also originates from the new pairwise features. As a result, ParT appears to have an excellent balance between performance and computational cost.

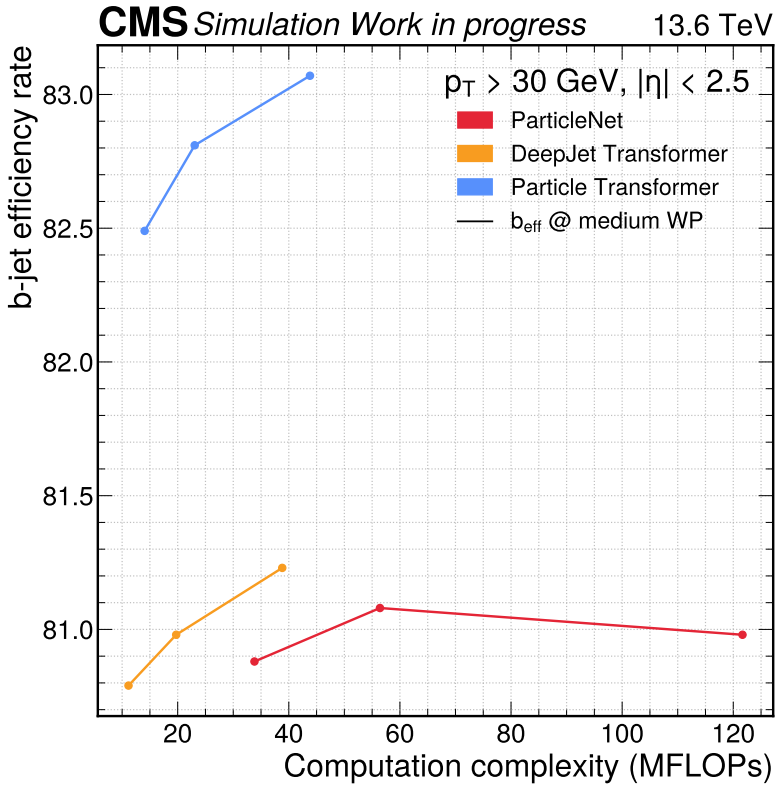

Figure 3.15: b_{eff} @ medium WP performance versus the computation complexity of ParticleNet (red), DeepJet Transformer (orange) and Particle Transformer (blue).

3.5 Summary and outlook

In this chapter, we have explored the application of deep learning algorithms to jet tagging, with a particular focus on representing jets as Particle Clouds. We introduced and compared the evolution of the neural network architectures used in heavy-flavour tagging, ranging from models based on fully connected and recurrent networks to graph networks and Transformers.

The results obtained demonstrate the advantage of Transformer-based models, not only in terms of classification performance but also in terms of computational complexity and inference efficiency. The DeepJet Transformer model, using the same set of input variables, outperforms the state-of-the-art ParticleNet while offering significantly improved computational complexity and inference speed. The Particle Transformer model, which differs from the DeepJet Transformer by its innovative use of kinematic features between pairs, significantly improves tagging performance in exchange for an increased computational cost, which remains significantly lower than that of ParticleNet. Measurements of model complexity also reveal that, despite a higher number of parameters, Transformer-based models remain faster and more resource-efficient than their predecessors based on Dynamic Graph Convolutional Neural Networks.

Additional studies have also demonstrated the importance of adhering to the representation of the jet as a Particle Cloud by showing the performance loss that DeepJet Transformer experiences when it includes non-permutation-invariant components typically used in Transformer models applied to LLMs [76, 111, 182]. We have also demonstrated that the ablation of these components does not reduce our model's performance. These two results indicate that creating permutation-invariant models ensures the robustness and proper functioning of the algorithms.

The study on model size scaling also highlighted the potential of Transformer models, which offer a significantly higher complexity-performance ratio. One can also notice in this analysis, that the ParticleNet architecture did not improve substantially when its depth

was increased, in contrast to DJT and ParT. These results, combined with the observations made by H. Qu et al. regarding the better scaling of Transformer models relative to dataset size [139], further emphasise the advantages of these models in terms of versatility for both training dataset size and model size. This ensures that employing this type of architecture guarantees the best performance compared to the other architectures discussed here, regardless of the model size or the amount of data available.

These advancements pave the way for new research on the use of Transformers for other tasks in particle physics, where the complex structure and interactions of the data can benefit from the flexible and powerful approach of Transformer neural networks. Moreover, integrating these models into the analysis pipelines of experiments such as CMS significantly enhances the precision and efficiency of jet-flavour identification.

In conclusion, this study highlights the immense contribution of deep learning models to jet tagging and lays the groundwork for future developments aiming to fully exploit the capabilities of Transformer neural networks in high-energy physics. We have demonstrated that an extensive understanding of the structure of the physical objects considered, combined with the development of efficient neural network models that respect the properties of these objects, can lead to significant advancements in their identification.

Chapter 4

Robust jet tagging algorithms via adversarial training

In this chapter, we will address the robustness of our jet tagging algorithms through the challenge of mismodelling and its impact on performance. In Section 4.1, we will begin by briefly explaining the context of mismodelling and how it is managed after training through the calibration of algorithms. We will then discuss the perspective of sensitivity to mismodelling from the viewpoint of neural networks and the challenges involved in minimising the model's sensitivity in the context of better and better boundary decisions and how they became more sensible to input changes. We will explain why we will focus here on a particular method based on adversarial attacks, which will be further explored in Section 4.2. In this section, we will review the most commonly used adversarial attacks and their properties. We will then delve into their application in flavour tagging, first by discussing and analysing the results of a preliminary work previously conducted. We will then explain the limitations of that work and how, within the framework of the available resources, we co-developed and subsequently developed two adversarial attacks [185, 186] and a new adversarial training method that does not trade off performance for robustness gain in jet flavour identification. In Section 4.3, we will illustrate the achievable performance on our simulations and the link between the gained robustness and the importance of input features

through the lens of their gradients. Finally, in Section 4.4, we will conclude the results and the achieved work, where we will summarise the findings and highlight which training method with which type of attack provides the best possible robustness, hence establishing the current state-of-the-art in robustness for flavour tagging. We will also emphasise the potential future prospects of such methods, particularly regarding the critical role that the importance of variables can play in developing these techniques, as well as the limitations these methods have in improving robustness against mismodelling in our simulations.

4.1 The mismodelling challenge of flavour tagging

In this part of the chapter, we will first address the issue of calibration of the jet tagging algorithms from the usual perspective in collider physics in Section 4.1.1. Then, we will draw an analogy with the problem from a Deep Learning perspective in Section 4.1.2. We will propose two dual viewpoints on generalisation and explain our motivation for employing one of these methods in the context of this work on the robustness of jet algorithms.

4.1.1 Callibration of jet tagging algorithms

In the previous chapter, we observed the performance of our algorithms applied to Monte Carlo simulation samples. These simulation samples describe the reality of detection conditions with relative accuracy, as illustrated by Figure 4.1 for the 3D impact parameter significance of the most displaced track and the DeepJet b vs all discriminant. Indeed, the description of the physics of events remains imperfect. When applying our jet identification algorithms to the data recorded by the CMS experiment, we must apply corrections to account for these differences between the simulated samples and the recorded data [61]. Among the key mismodelling sources are the

parton shower and fragmentation modelling and the simulation of the detector component responses.

To measure the effectiveness of our algorithms, we measure the identification and misidentification efficiency of our algorithms for given discriminant threshold values, most often our defined WP as follows:

$$\epsilon^f = \frac{N^f_{\text{pass}}}{N^f_{\text{total}}} \tag{4.1}$$

where N^f_{pass} and N^f_{total} represent the number of jets selected by the algorithm at the given threshold and the total number of jets for a given flavour f (b, c, or udsg), respectively. By applying the measurement of these efficiencies on the MC simulation samples ϵ^f_{MC} and the recorded data ϵ^f_{data}, we can obtain calibration factors [61,188–190], usually referred to as scale factors (SFs), defined by Equation (4.2). These SFs will be applied to the simulation samples to improve their description of data in analyses that employ jet identification. An example of b-tagging SFs as a function of jet p_{T} is shown in Figure 4.2.

$$\text{SF}^f = \frac{\epsilon^f_{data}}{\epsilon^f_{MC}} \tag{4.2}$$

These calibration methods are applied to selection regions enriched in b, c, or udsg jets, most often from $\text{t}\bar{\text{t}}$ events or QCD multijet events [61]. Several different methods exist for different phase spaces. One example is the kinematic (kin) method that identifies dileptonically decaying $\text{t}\bar{\text{t}}$ events containing exclusively two b jets via a BDT. We can measure the SFs using jets with a very high b jet purity rate from these events. Other methods based on W+c events [188], naturally enriched in c jets, are used for the calibration of c identification.

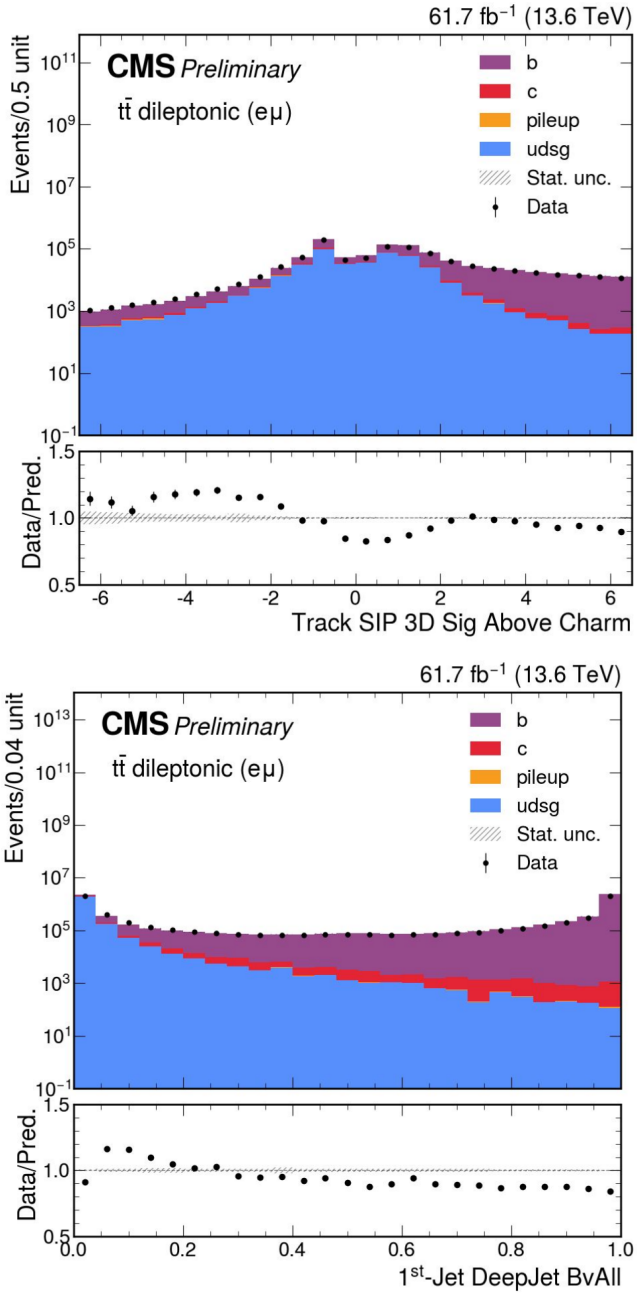

Figure 4.1: data/MC agreement plots of the $t\bar{t}$ $e\mu$ region for the Run 3 2022+2023 data taking. The upper plot represents the data/MC agreement of the 3D impact parameter significance of the most displaced track, and the lower plot is the data/MC agreement of the DeepJet algorithm for the b vs all discriminant before calibration [187].

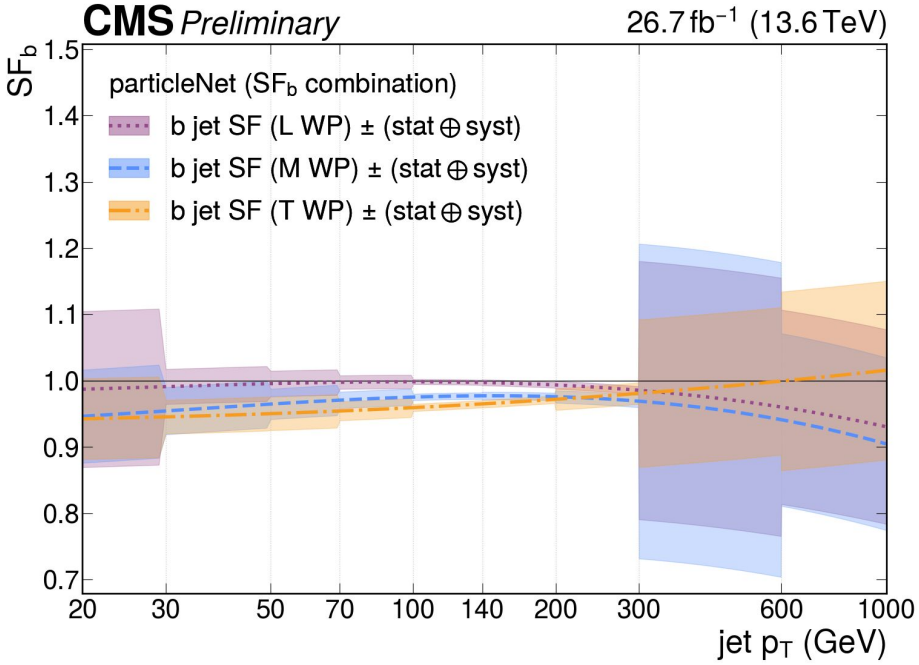

Figure 4.2: Combined b-tagging SFs for the loose, medium, and tight WP of the ParticleNet algorithm with their statistical and systematics uncertainties for the Run 3 2022 data taking [190].

4.1.2 The neural network perspective of mismodelling

The evolution of algorithms towards increasingly larger neural network models, containing more input variables and trained with more effective training methods to improve generalization, can lead to numerous problems in their usage. Indeed, our growing models risk facing the danger of training data memorization, a type of overfitting where a model manages to encode the entire training dataset into its latent variable space, resulting in poor generalization capacity. The interrelation between generalization and memorization is a crucial topic in deep learning, for which it has been demonstrated that neither usual regularization methods nor model architectures alone explain the generalization capacities of increasingly large models [191]. The contribution of implicit mechanisms from training and gradient descent also plays a crucial role in these generalization ca-

pacities.

More specifically, it has been demonstrated that training to minimise a loss function is not sufficient for the model to correctly generalise during its usage. Increasing model sizes, combined with adaptive moment optimizers and complex non-convex learning tasks, can reduce our models' generalization capabilities [192, 193]. The optimal and robust convergence of the training has become more and more challenging, shaping the loss function with multiple local minima. [192]. Thus, as we improve the accuracy of our models trained on our MC simulations, we encounter a new challenge regarding their application to real data. Our models have become much more precise, with increasingly refined and sensitive decision boundaries, making them more susceptible to perturbations. We want to explore methods to make our models more robust and enhance their generalization capability when applied to real data. More specifically, we aim to develop a method independent of the actual data/MC disagreement that allows for the design of generalised robustness derived from mathematical principles and agnostic to mismodelling. Among the methods to improve model generalization capacity, the Sharpness-Aware Minimization (SAM) [193] method is well known. This method consists of a two-step learning process that perturbs the weights and biases of our model to construct a flatter minimum of the loss function landscape, illustrated in Figure 4.3, leading to better generalization. SAM methods have been applied successfully to CV [193, 194] and LLMs [195].

Beyond the discussion of generalization capacity, it is important to note that our model is trained on simulations, leading to convergence within a suboptimal phase space of input variables. This suboptimality, combined with the model's sensitivity to potential mismodelling, underscores the need for calibration. The challenge of modern jet algorithm training, therefore, is twofold: to improve the model's raw performance on simulated data and to mitigate the influence of calibration factors on its performance in real-world analyses. To address this, we propose a training strategy that focuses on minimising the model's sensitivity to variations in input features, enhancing its robustness and reducing the impact of mismodelling.

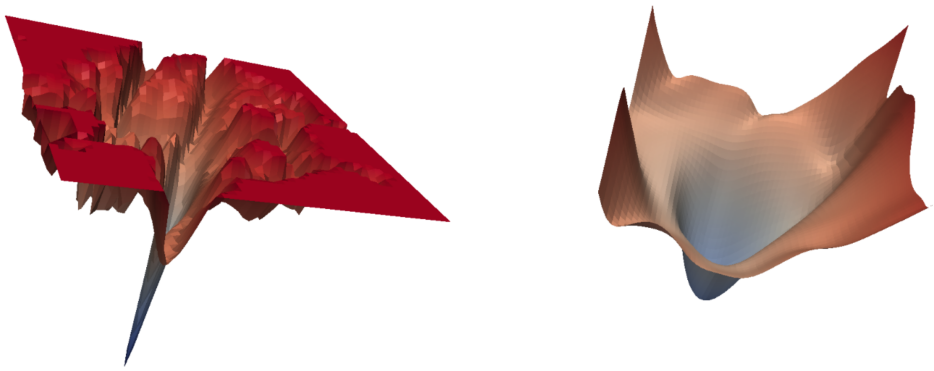

Figure 4.3: 2D projection of the loss landscape of a ResNet-50 model [121] trained with SGD on the left and SAM on the right [193]. The sharpness of the model trained with SAM is reduced, leading to a wider local minimum.

This method aims to optimise performance on simulation samples while ensuring that the algorithm generalises effectively to real data, thus reducing the need for extensive calibration adjustments.

4.2 Adversarial attacks for jet algorithms

In this part of the chapter, we will detail the principle of adversarial attacks in Section 4.2.1 and provide some examples of methods commonly used in research. We will then use these works, combined with preliminary studies on their application to jet tagging, to design new attacks and a new training strategy in Section 4.2.2.

4.2.1 Adversarial attacks and related works

Adversarial attacks are techniques used to deceive machine learning models by introducing small intentional perturbations in the input data designed to cause significant prediction errors in the model. For example, an image correctly classified as a ‘panda’ can be subtly modified to be misclassified as a ‘gibbon’ by the model, as illustrated in Figure 4.4. Adversarial attacks exploit vulnerabilities in machine learning models, highlighting weaknesses in the robustness

and security of the algorithms. Research on such attacks was historically developed in CV [196–198] before becoming widespread in other domains, including recently in LLMs [199, 200] and reinforcement learning [201, 202]. In the context of this thesis, we will mostly focus on white-box adversarial attacks. These attacks have direct access to the model under consideration and all its components, such as weights, biases, graphs, cost functions, inputs, and targets. Conversely, black-box attacks do not have access to the internal information of the model aside from its predictions. This attack must find an indirect method to disrupt the predictions, such as by generating adversarial examples from a surrogate model [203]. For lexical simplicity in this section, when we refer to adversarial attacks without specifying if they are white-box or black-box, hereafter, we will refer to white-box attacks. In the later sections, black-box attacks will be mentioned when used. The general operation of an adversarial attack can be defined as follows:

$$x_{\text{adv}} = x + \square_{\text{adv}}(L, \theta, x, y, \epsilon) \tag{4.3}$$

where x and x_{adv} represent the input variables and the input variables distorted by the adversarial attack, and y the target prediction. The adversarial attack is represented by $\square_{\text{adv}}$ and depends on the selected method. The cost function is shown as L, usually CE or MSE, θ represents the model's weights and biases, and ϵ is a magnitude parameter of the attack that ensures the caused perturbation is small. In most cases, adversarial methods are based on extending gradient backpropagation to the input variables, which we denote as $\nabla_x L(\theta, x, y)$.

Along with the development of adversarial attacks, new training strategies designed to minimise these attacks were developed [198, 204, 205]. These so-called adversarial training strategies aim to expose the model to both the original (nominal) samples and their adversarial versions during the training, thus exposing the model to perturbed data and allowing it to minimise its sensitivity to attacks and, therefore, building more robust models against perturbations creating mismodelling.

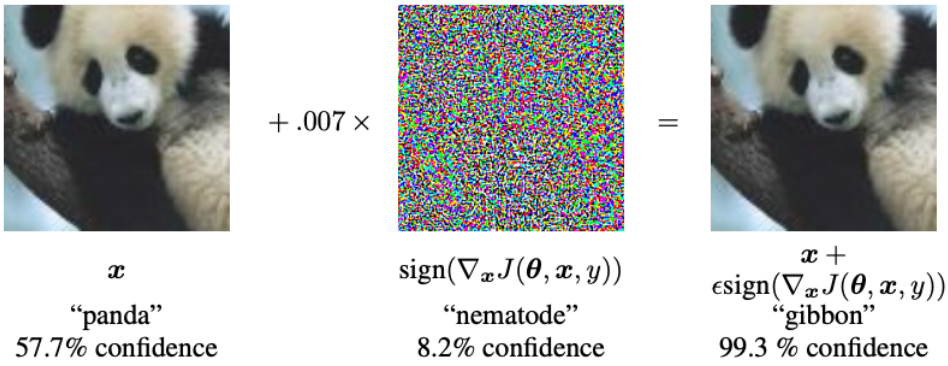

Figure 4.4: Illustration of an adversarial attack applied in CV [196]. The image is modified with respect to its loss function for performing a distortion of its inputs, leading to a mislabelling of the distorted image, which, however, remains completely discernible and understandable to the human eye.

Among the commonly used attack methods, the most common is the Fast Gradient Sign Method (FGSM) [196]. This attack involves creating a perturbation from the gradient sign with respect to the input features. The advantage of the FGSM method lies in its ability to quickly generate attacks based on the gradient of the inputs, as described by Equation (4.4). This attack can be viewed as the linear approximation of maximising the cost function over the hyper ball $\mathbb{B}(x, \sqrt{N}\epsilon)$ centred on the input features and with a radius $\sqrt{N}\epsilon$, N being the number of input features: $\max_{x_{\text{adv}} \in \mathbb{B}(x, \sqrt{N}\epsilon)} L(\theta, x_{\text{adv}}, y)$. In its implementation, the computational complexity of the FGSM method requires a complete forward-backward step to measure the gradient and obtain the desired perturbation, the same cost as a training step on the same samples.

$$x_{\text{adv}} = x + \epsilon \cdot \text{sign} \nabla_x L(\theta, x, y) \tag{4.4}$$

A natural extension of the FGSM method is the iterative Projected Gradient Descent method (PGD)[197], also known as FGSM^k. This method involves iteratively crafting a final adversarial attack x^k_{adv} by performing k FGSM attacks as illustrated by Equation (4.5). Each iteration is controlled by an element-wise clipping $\text{CLIP}_{x,\delta}$, ensuring that each element of the adversarial tensor created at

step t remains within a value range of $[x - \delta, x + \delta]$. PGD attacks can also be seen as a linear approximation of solving the problem $\max_{x_{\text{adv}} \in \mathbb{B}(x, \sqrt{N}\epsilon)} L(\theta, x_{\text{adv}}, y)$. The PGD method significantly increases computational complexity, requiring k forward-backward steps to construct x^k_{adv} iteratively. With typical attacks using a k parameter value around 20 [198], the PGD method is significantly more costly than its simplified FGSM variant.

$$\begin{aligned} x^0_{\text{adv}} &= x \\ x^t_{\text{adv}} &= x^{t-1}_{\text{adv}} + \epsilon \cdot \text{adv}\nabla_x L(\theta, x^{t-1}_{\text{adv}}, y) \end{aligned} \tag{4.5}$$

While the previous methods proposed attacks that modify all input features simultaneously, Papernot et al. introduced a method that allows modifying the input variables one by one [206], known as Jacobian-based Saliency Map Attacks (JMSA). This attack, based on the gradient $\nabla_x F(\theta, x)_l$, where F represents the model's prediction, measures a saliency map S to determine which input variables contribute the most to the prediction of the target class l. This attack is no longer measured through its cost function L but solely through the model's gradient with respect to the prediction class l for which the attack is determined. Based on this evaluation of each variable's importance, JMSA follows an iterative process, described by Equation (4.6), to deceive the model's prediction. At each selection of the most important variable, a value change ϵ is made to better align with the targeted class l. The process is repeated until either the model's predicted class matches the targeted class l or the norm of the change in input features $\delta_x = x - x_{\text{adv}}$ exceeds a set threshold Γ. In JMSA's original application context, CV [206], the L_0 norm is chosen, counting the number of modified pixels. The JMSA paradigm differs from FGSM and PGD methods, as it now seeks to minimise the following problem: $\arg\min_{||\delta_x||} F(x + \delta_x) = l$.

$$
\begin{aligned}
&x_{\text{adv}} \leftarrow x \\
&\textbf{while } F(x_{\text{adv}}) \neq l \text{ and } ||\delta_x|| < \Gamma\textbf{:} \\
&\quad \text{Compute } \nabla_x F(\theta, x_{\text{adv}})_l \\
&\quad \text{Compute } S(\nabla_x F(\theta, x_{\text{adv}})_l, x_{\text{adv}}, l) \\
&\quad i_{\max} = \arg\max_i S(\nabla_x F(\theta, x_{\text{adv}})_l, x_{\text{adv}}, l) \\
&\quad x_{\text{adv},i_{\max}} = x_{\text{adv},i_{\max}} + \text{sign}(\nabla_x F(\theta, x_{\text{adv}})_l[i_{\max}]) \cdot \epsilon \\
&\quad \delta_x = x - x_{\text{adv}}
\end{aligned}
\tag{4.6}
$$

The computational complexity limits of JMSA are similar to the PGD method. Indeed, in the initial paper, applying this method to the MNIST dataset [207], consisting of 28×28 pixel handwritten digit images, typically required modifying 4% of the pixels to change the prediction, which equated to approximately 31 forward-backward loops per images. This number of iterations is of the same order of magnitude as the PGD method.

A final adversarial attack method is the Carlini & Wagner (CW) attack [208]. The CW attack, considered as one of the most powerful and resulting in minimal distortion, involves the minimization of a cost function J defined by Equation (4.7):

$$
J(\theta, x_{\text{adv}}, y) = \lambda \cdot \text{dist}(x, x_{\text{adv}}) + \beta \cdot G(\theta, x_{\text{adv}}, y) \tag{4.7}
$$

where G is a function aimed at maximising misclassification such that $G(\theta, x_{\text{adv}}, y) \leq 0$ when the jet is misclassified. The trivial function using the cross-entropy loss, $-\text{CE} + 1$, has been tested among those functions. Carlini & Wagner suggest other functions in their paper [208], demonstrating that there is no optimal β factor, and by extension, λ, that ensures optimal convergence while simultaneously reducing both the chosen distance dist and the function $-\text{CE} + 1$ [208]. The distance dist between the original input and the adversarial input is minimised according to either the L_0, L_2, or L_∞ norm to construct an attack that disrupts the prediction with the smallest possible change in variables. Equation (4.9) describes the

norms. We can notice the L_0 norm is not a real norm as it does not respect the homogenous property of a norm and we acknowledge the usage of L_0 norm is, therefore, a terminology abuse. The adversarial example is then obtained iteratively in a manner analogous to the PGD method via:

$$\begin{aligned} x^0_{\text{adv}} &= x \\ x^t_{\text{adv}} &= x^{t-1}_{\text{adv}} + \epsilon \cdot \text{adv}\nabla_x J(\theta, x^{t-1}_{\text{adv}}, y) \end{aligned} \tag{4.8}$$

$$\begin{aligned} ||x||_{L_0} &= \sum_n |x_i|^0 \text{ with } |0|^0 = 0 \\ ||x||_{L_2} &= \sqrt{\sum_n |x_i|^2} \\ ||x||_{L_i nf} &= \text{sup}_n |x_n| \end{aligned} \tag{4.9}$$

The CW attack is computationally expensive due to its iterative nature. The authors recommend using the L_2 norm with a maximum of 1000 iterations [208]. This can result in a significant cost to construct an adversarial example for a single element.

4.2.2 Adversarial attacks applied to jets

The first use of adversarial attacks was conducted with FGSM attacks on the DeepJet algorithm [185, 209]. These quickly demonstrated that contrary to preliminary simplified observations [209], such training applied to simulated jets produces an underperforming classifier compared to the nominal model [185]. However, adversarial training has shown its ability to improve robustness against adversarial attacks by reducing their impact on jet misclassification. Figure 4.5 illustrates, for b vs udsg tagging, the trade-off between nominal performance and robustness. For this first approach, the cost function used for adversarial training was cross-entropy, employing adversarial inputs x_{adv} instead of nominal inputs x:

$$L_{\text{adv}}(x, x_{\text{adv}}, y, \theta)_{\text{adv}} = CE(x_{\text{adv}}, y, \theta) \tag{4.10}$$

Adversarial training is compared with nominal training, which utilises only the nominal samples and the standard loss function, in this case, the cross-entropy for classification:

$$L(x, y, \theta)_{nom} = CE(x, y, \theta) \tag{4.11}$$

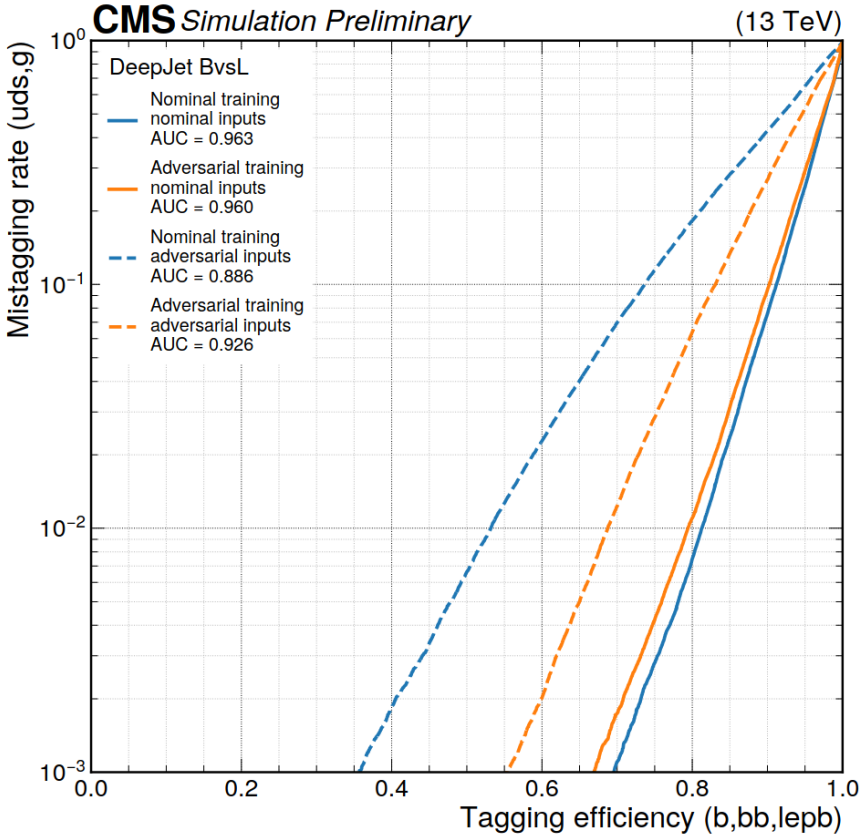

Figure 4.5: ROC curve performance of the b vs udsg rejection of the DeepJet algorithm trained in nominal mode (blue) or with FGSM attacks (orange) [185]. We can observe that the FGSM adversarial training implies a tradeoff between the nominal performance (solid lines) and robustness against the FGSM attacks (dashed lines).

These initial adversarial trainings highlight the limitations of the employed techniques. First, the FGSM attack is not well-suited

to the variables used and constructs rather artificial changes in the variables. Indeed, this attack was developed for CV models [196], where the entire input data consists of normalised values between [0,1] of the RGB channels or a black-and-white channel, depending on the considered type of image. For these, making a change of magnitude ϵ to each variable makes sense and allows one to exploit only the sign of the gradient on the inputs to indicate the direction of the change. However, in the case of jet algorithms, we have a set of input variables of different natures, each with its own distribution. Thus, making a change of magnitude ϵ will not have the same implication if, for example, we modify the transverse impact parameter value of a CPF or its pseudorapidity relative to the jet axis. Moreover, even though it might be tempting to explore other reference methods such as PGD or CW attacks, we would then face limitations in the computational capacity and training time that we can allocate, which is on the order of one week with the currently available resources. Indeed, the training of Particle Transformer, the current state-of-the-art in jet algorithms, requires a training time on the order of one day with the setup described in Chapter 3. It should be noted that the mentioned trainings benefited from the latest advancements in improving GPU usage through the b-hive framework [167], and without its use, the training duration would have been extended by approximately three to four days. Using the FGSM method requires an additional forward-backward loop in addition to that used by nominal training, thus doubling the necessary computation time. If we were to explore methods such as PGD or CW attacks, this would require, at a minimum, a greater number of forward-backward loops to create our adversarial inputs. Using PGD attacks with six iterations would result in a training time on the order of one week, our maximum allocated time, knowing that most applications of PGD in the literature use between 20 and 40 iterations [198, 210, 211]. Since we would also like to explore the use of larger models as well as more input variables, we cannot afford, under these conditions, to use such iterative methods.

Furthermore, the training strategy employed, based solely on cross-entropy on adversarial predictions, is another limiting factor of

this training method. To minimise the impact of training on nominal performance, we would like the training to aim at both making correct predictions for nominal and adversarial examples and minimising the change in prediction between them. To achieve this, a new adversarial training strategy based on nominal and adversarial examples is necessary to establish a new global cost function.

Finally, it is important to note a particular phenomenon for fast adversarial training using attacks such as FGSM. Research has shown particular sensitivity to Catastrophic Overfitting (CO), a phenomenon where the model, instead of gaining robustness, experiences a collapse in robustness to other types of attacks than the one used in training, while accuracy against the training attacks reaches nearly 100% [212,213]. This phenomenon, absent during the training with the DeepJet model, emerged when we attempted this training method on ParT. It is worth noting that the observed phenomenon differed from that described in the context of CV models. Not only did we face a drop in accuracy against other attacks, but we also encountered general overfitting, leading to a decline in the model's nominal accuracy. This significant degradation rendered the ParT models trained with FGSM attacks ineffective despite attempts to adjust parameters such as the attack magnitude ϵ or the model's learning rate. In the context of more advanced models like ParT, which have superior separation capability and a larger number of parameters, FGSM attacks become inappropriate for applications in jet tagging algorithms.

The Normed Gradient Method (NGM)

A first suggested modification was to no longer rely solely on the sign of the gradient for our attacks. Instead, we decided to work directly with the gradient with respect to the input features, $\nabla_x L(\theta, x, y)$. Thus, we allow the attacks to transition from a discrete variable change defined by value changes of $\pm\epsilon$ to a continuous range of values over $\mathbb{R}^N$, where N is the number of input variables. To control the range of gradient values that can vary from one jet to another, as well as the potential evolution of the gradient throughout the training, we

will use the unit vector u_{adv} defined by the L_2 norm as follows:

$$u_{\text{adv}} = \frac{\nabla_x L(\theta, x, y)}{||\nabla_x L(\theta, x, y)||_{L_2}} \tag{4.12}$$

We can interpret this vector as the direction on the hyperball $\mathbb{B}_{(x,1)}$ that applies the highest possible penalty to the chosen cost function L. In other words, it is the direction to follow to maximise the function L and thereby deceive the model.

Furthermore, since the cost function with respect to the input features plays a crucial role, we decided to inject a rescaling factor that encodes the information of the scales of each feature into the attack magnitude parameter ϵ as:

$$\epsilon_i = \epsilon \cdot \sqrt{Var(\tilde{x}_i)} \tag{4.13}$$

where ϵ_i is the attack magnitude of the i-th variable, whose variance is calculated from the values $\tilde{x}_i$ obtained from the central 90% quantiles of the distribution. These values ϵ_i are measured among the constituents of the same nature (CPFs, NPFs, SVs) and on approximately 5% of the training data before the training itself. The parameter ϵ remains a free and adjustable parameter during training to control the magnitude of the attacks.

Thus, we can define a new attack called the Normed Gradient Method (NGM), leveraging the benefit of transitioning from a discrete to a continuous variable change through the use of the unit vector u_{adv}, as well as the exploitation of per-features attack magnitude parameter parameters ϵ_i. Analogous to the previously discussed attacks, the NGM attack seeks to solve the linear problem $\max_{x_{\text{adv}} \in \mathbb{B}(x, u_{\text{adv}} \cdot \epsilon_i)} L(\theta, x_{\text{adv}}, y)$, aimed at maximising the cost function on the surface of the hyperball $\mathbb{B}(x, u_{\text{adv}} \cdot \epsilon_i)$ centered on the input features x and with a radius of $u_{\text{adv}} \cdot \epsilon_i$. It is important to note that the attack magnitude $u_{\text{adv}} \cdot \epsilon_i$ can vary from one jet to another. The following equation defines the new attack:

$$
\begin{aligned}
x_{\text{adv}} &= x + \epsilon_i \cdot u_{\text{adv}} \\
x_{\text{adv}} &= x + \epsilon \cdot \sqrt{Var(\tilde{x}_i)} \cdot \frac{\nabla_x L(\theta, x, y)}{||\nabla_x L(\theta, x, y)||_{L_2}}
\end{aligned}
\tag{4.14}
$$

The Rectified Normed Gradient Method (R-NGM)

The NGM method, although effective in preserving correlations against adversarial attacks, is not entirely satisfactory within the paradigm considered. Indeed, we could use the gradient with respect to the input features as an indicator of feature importance. By emphasising the connection between adversarial robustness and the importance of input features, we observe that the gradient on the input features, $\nabla_x L(\theta, x, y)$, or its unit version u_{adv}, takes a form analogous to methods used for evaluating input feature importance [214–216].

Thus, we can consider that $\nabla_x L(\theta, x, y)$ linearly approximates the contribution of each variable to the minimization (or maximization in the case of an adversarial attack) of the cost function. As a result, a high value represents a significant contribution, while a low or near-zero value represents the opposite. It is important to note that the gradient sign here is only relevant to indicate the direction in which the modification should occur. For two input features x_a and x_b with gradient values of $+A$ and $-A$, their importance is similar; only the direction of change to apply our penalty will differ. Therefore, with this analogy, we decided to rectify the initially designed NGM method by removing the $\sqrt{Var(\tilde{x}_i)}$ terms. What remains is the global control parameter for attack magnitude ϵ and the unit vector u_{adv}. Compared to the NGM attack, this new Rectified Normed Gradient Method (R-NGM) attack seeks to solve the problem $\max_{x_{\text{adv}} \in \mathbb{B}(x,\epsilon)} L(\theta, x_{\text{adv}}, y)$, maximising the cost function on the surface of the hyperball $\mathbb{B}(x, \epsilon)$ centered on the input features x and with a radius of ϵ, for which we consider the unitary vector u_{adv} as the direction of this best penalty in the first order regime. Contrary to the NGM method, the magnitude of the attacks remains the same for all jets, similar to the previous methods we discussed.

The Rectified Normed Gradient Method attack is defined as follows:

$$\begin{aligned} x_{\text{adv}} &= x + \epsilon \cdot u_{\text{adv}} \\ x_{\text{adv}} &= x + \epsilon \cdot \frac{\nabla_x L(\theta, x, y)}{||\nabla_x L(\theta, x, y)||_{L_2}} \end{aligned} \tag{4.15}$$

Analogy between adversarial attacks and attribution method

Adversarial training methods aim to minimise the impact that a change in inputs can have on the prediction of a neural network. We can establish a link between these methods and the evaluation of the importance of input features. Among the most well-known attribution methods is the Integrated Gradients method [214] (IG), whose formulation is as follows:

$$IG_{\theta_k}(x) = (x - x') \times \int_{\alpha=0}^{1} \frac{\partial \theta_k(x' + \alpha(x - x'))}{\partial x} d\alpha \tag{4.16}$$

where the attribution for the prediction of the k-th class, IG_{θ_k}, depends on the input features x from an anchor point x' and is integrated from the prediction of the k-th class, denoted by θ_k.

In the context of IG, for interpreting a multi-class classification task, replacing the specific output of a class with the cost function, such as CE, offers a more comprehensive view of feature importance. Indeed, this approach simplifies and generalises the concept to all classes. Moreover, the contribution of each feature will be evaluated not only based on the prediction of the class to which the object belongs but also across all prediction errors. Thus, this modification of the attribution method is more representative of the overall behaviour of the model. We therefore define our attribution IG_L based on the cost function L as follows:

$$IG_L(x) = (x - x') \times \int_{\alpha=0}^{1} \frac{\partial L(\theta, x' + \alpha(x - x'), y)}{\partial x} d\alpha \tag{4.17}$$

From Equation (4.17), we can define the Riemann approximation, in m steps, for the integral analogously to the original method [214]:

$$IG_L(x) \approx (x - x') \times \sum_{i=1}^{m} \frac{\partial L(\theta, x' + \frac{i}{m}(x - x'), y)}{\partial x} \times \frac{1}{m} \tag{4.18}$$

The choice of anchor is an important aspect of this attribution technique. Although the authors recommend selecting the least likely example from the class we wish to study [214], we diverge from this method. Indeed, this approach works only when we want to evaluate the importance for a single predicted class. In our case, we aim to evaluate the importance with respect to classification as a whole, and there is no simulated jet that can serve as a 'neutral' anchor. Thus, we adopt the second recommended option, which is to use a null anchor. In the same way that a black image or a text with no words is used, we choose the 'empty' jet, consisting only of values equal to 0, as the anchor. Thus, after setting $x' = 0$, we observe that by simplifying the Riemann method to a single iteration, similar to our adversarial attacks, we obtain the following equation:

$$IG_L(x) \approx x \times \nabla_x L(\theta, x, y) \tag{4.19}$$

We then observe the link between this feature importance evaluation method and our attacks. Under the same single-step approximation regime and using an appropriate anchor, we observe that the FGSM, NGM, or R-NGM attacks are variants of the approximation of $IG_L(x)$, modifying the factor x through the sign function or a normalization factor $\frac{\sqrt{Var(\tilde{x}_i)}}{||\nabla_x L(\theta,x,y)||_{L_2}}$ or $\frac{1}{||\nabla_x L(\theta,x,y)||_{L_2}}$. Our attacks thus exploit the importance of input features through their gradient to generate perturbations that most effectively affect the cost function and, therefore, the prediction.

A trade-off adversarial training strategy for jet algorithms

Inspired by the TRadeoff-inspired Adversarial DEfense via Surrogate-loss minimization (TRADES) method [198], we adapt the training method by introducing a hybrid cost function. For each epoch, 50% of the iterations are trained in nominal mode to ensure the proper convergence of our model towards the performance optimum on nominal samples. For the remaining 50%, the adversarial mode, we employ a function that measures the trade-off between the accuracy on adversarial examples, represented by the cross-entropy evaluated with adversarial samples, and the minimization of the difference between the probabilities predicted by the nominal and adversarial examples. The choice to use the adversarial mode 50% of the time is motivated by the training time limitations encountered. This trade-off ensures that adversarial training is feasible within the maximum time allotted. If the time constraint were to be removed, then an optimization of the adversarial mode fraction, or even fully adversarial training, should be considered. We choose the Kullback–Leibler (KL) divergence for the second loss. This new cost function, denoted as L_{adv}, is defined by the following equation:

$$L_{\text{adv}}(x, x_{\text{adv}}, y, \theta) = \begin{cases} CE(\theta(x), y) \text{ if nominal mode} \\ CE(\theta(x_{\text{adv}}), y) + \lambda \cdot KL(\theta(x), \theta(x_{\text{adv}})) \\ \text{otherwise} \end{cases} \tag{4.20}$$

where the parameter λ is a free hyperparameter helping to control the trade-off. A grid search has been applied to find the best value of λ. This grid search showed the training was not sensitive to this hyperparameter until it reached high values of order $\mathcal{O}(50)$, for which the model could not converge anymore. Therefore, we fixed the value of λ to one.

Similarly, grid searches were performed to find the highest ϵ value the NGM and R-NGM training could afford without degrading the nominal performance of the model. A value of ϵ of 0.01 and 0.002 have been found for NGM and R-NGM, respectively.

4.3 Adversarial performances

In this last part of the chapter, we will evaluate in Section 4.3.1 the performance of our different training methods to assess the nominal performance and the robustness they provide. Then, in Section 4.3.2, we will address the issue of robustness from the perspective of the gradient with respect to the input features and whether the relationship between them and their importance on the prediction can be measured and used as a metric. In this Section, we will also address a problem arising from adversarial training methods, which leads to gradient masking, and we will demonstrate that our training methods have not been affected by this through adapted black-box attacks.

4.3.1 Robustness of the taggers

To train and evaluate the performance of our adversarial training strategies, we used the same setup and data as described in Section 3.3. The employed structure is that of ParT, which is the state-of-the-art architecture in terms of performance. We trained three models with three different training strategies. First, a nominal training was performed to serve as a baseline, particularly for performance evaluated on nominal samples. NGM and R-NGM training were also conducted to evaluate the robustness achieved by the two methods and to compare the robustness gain relative to nominal training. No FGSM training is shown here due to the catastrophic overfitting problem mentioned in the previous section.

The performances on the nominal test samples are illustrated in Figure 4.6. The two adversarial models achieve the same nominal performance as the baseline nominal training. This highlights the first success of our method. We were able to build an adversarial training strategy capable of achieving the same level of nominal performance and thus potentially improving the robustness of our algorithms without any loss of performance on our MC simulation samples.

To rank the robustness of the different training strategies, the

performance was also evaluated on three kinds of adversarial attacks, the FGSM, NGM and R-NGM. The performance against the NGM attacks is shown in Figure 4.7 and against R-NGM attacks in Figure 4.8. Finally, the robustness against the FGSM attacks is shown in Figure 4.9. Only the performance of b-tagging is shown in this section. The performance of c-tagging follows the same trend, and thus, the same conclusion and the performance plots are available in the Appendix D.

The performances of the R-NGM and NGM trainings are significantly better compared to the nominal training against the different adversarial attacks. This confirms the benefit of adversarial training for enhancing the robustness of the algorithms against attacks of different natures. Adversarial training using a specific attack guarantees an improvement in robustness against the three types of attack considered, indicating that the benefit appears to be generalised.

We can observe across all adversarial attacks that the R-NGM training achieves the best robustness. This training even manages to perform similarly with NGM training on NGM attacks, even surpassing it in udsg rejection. Thus, adversarial training with the R-NGM attack has demonstrated the greatest robustness, being able to first converge and achieve the same level of performance as nominal training and also providing robustness against the three types of attacks considered in this thesis. The R-NGM attack, based on the link between the gradient on input features and their importance in contributing to the prediction, has constructed the most relevant attack for better robustness.

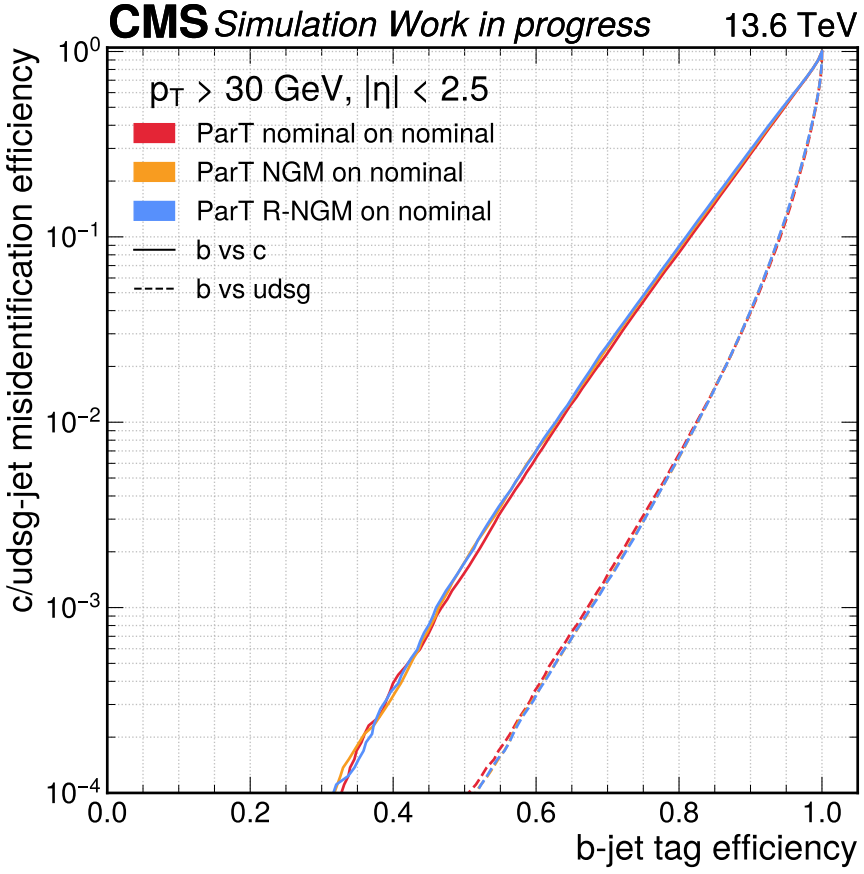

Figure 4.6: ROC curves performance on nominal samples of the b vs udsg (dashed lines) and b vs c (solid lines) rejection of the Particle Transformer algorithm trained in nominal mode (red) or with NGM training (orange) and R-NGM (blue). We can observe that the three models perform similarly when evaluated on nominal samples.

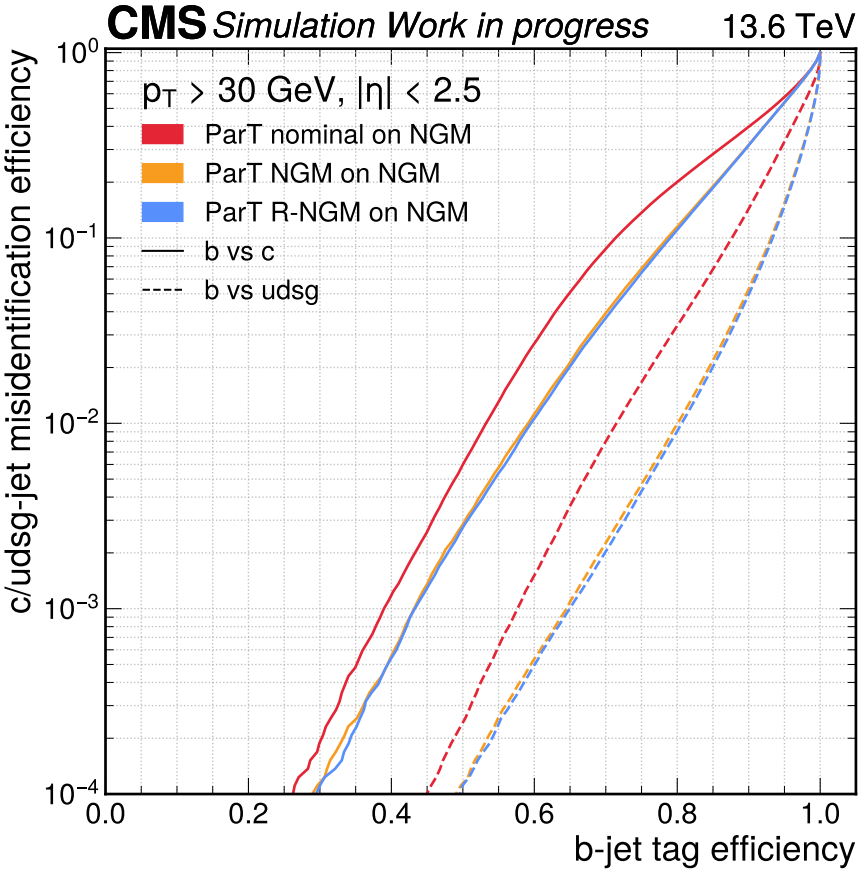

Figure 4.7: ROC curves performance on NGM samples of the b vs udsg (dashed lines) and b vs c (solid lines) rejection of the Particle Transformer algorithm trained in nominal mode (red) or with NGM training (orange) and R-NGM (blue). We can observe that the two adversarially trained models perform better than the nominal one. The R-NGM training achieves a similar robustness against NGM attacks than the NGM training, even slightly better for the b vs udsg rejection.

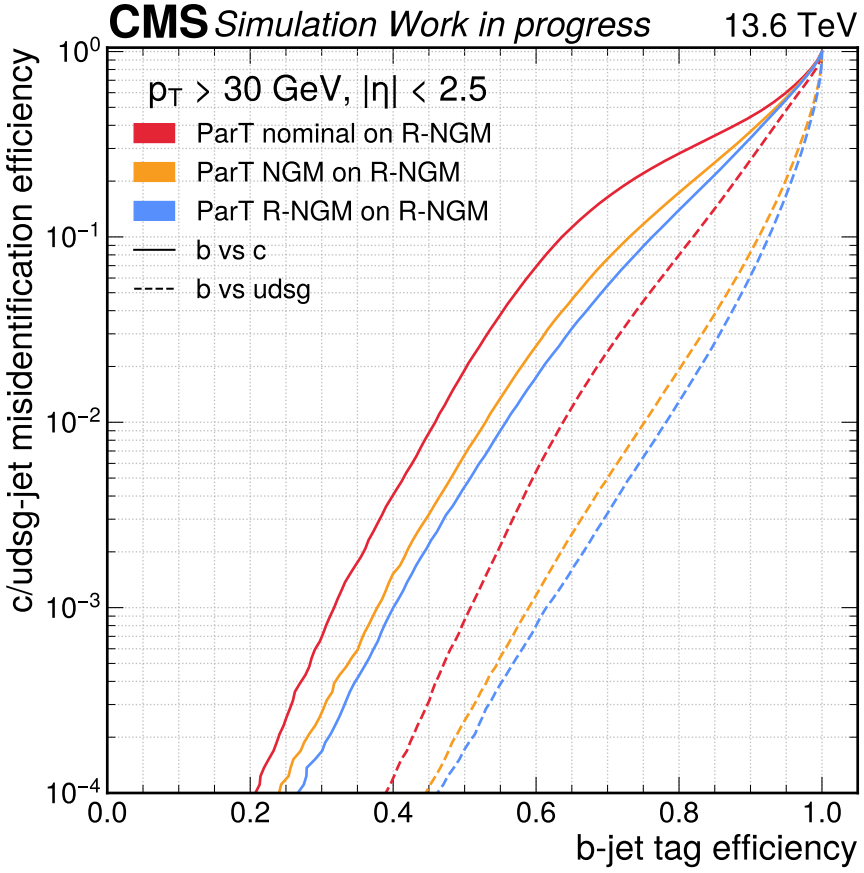

Figure 4.8: ROC curves performance on R-NGM samples of the b vs udsg (dashed lines) and b vs c (solid lines) rejection of the Particle Transformer algorithm trained in nominal mode (red) or with NGM training (orange) and R-NGM (blue). We can observe that the two adversarially trained models perform better than the nominal one. The R-NGM training achieves better robustness than the NGM training.

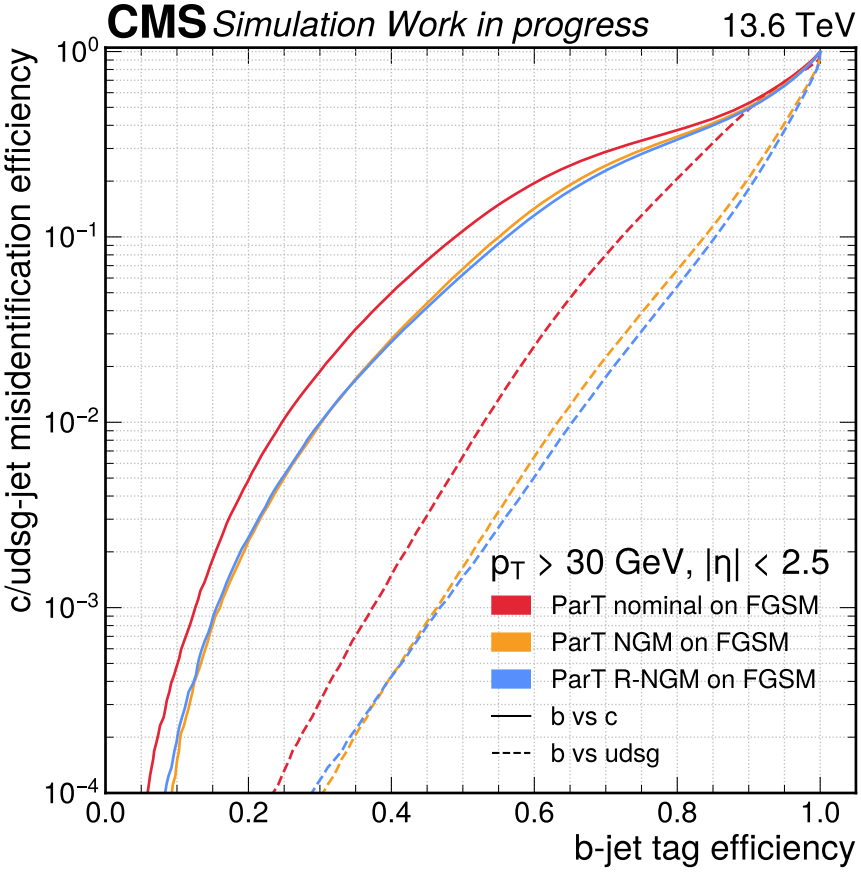

Figure 4.9: ROC curves performance on FGSM samples of the b vs udsg (dashed lines) and b vs c (solid lines) rejection of the Particle Transformer algorithm trained in nominal mode (red) or with NGM training (orange) and R-NGM (blue). We can observe that the two adversarially trained models perform better than the nominal one. The R-NGM training achieves better robustness than the NGM training.

4.3.2 Measurement of the gradients and Gradient Masking

To better understand the behaviour of our adversarial strategies and ensure the proper functioning of the obtained models, we measured the mean gradient on our test dataset. Indeed, the purpose of adversarial training is to make the model more robust to perturbations, and we can establish a strong link with the gradient obtained on our input variables and the input features importance[214–216]. We evaluate their norm, here chosen as the L_2 norm, denoted as $||\nabla X_i||^2$. To ensure greater robustness, models subjected to white-box attacks will reduce their sensitivity to input variables, naturally leading to a decrease in the gradient compared to our nominal model [217, 218].

Training strategy	$\|\|\nabla X_{\text{CPF}}\|\|^2$	$\|\|\nabla X_{\text{NPF}}\|\|^2$	$\|\|\nabla X_{\text{SV}}\|\|^2$
Nominal	0.05881	0.01032	0.00133
NGM	0.01098	0.01518	0.00059
R-NGM	0.00590	0.00057	0.00023

Table 4.1: Mean of the gradients for the CPFs, NPFs, and SVs elements evaluated on our test dataset for the three training strategies considered.

We can observe in Table 4.1 that the mean L_2 norm of our gradients is indeed lower for our adversarial trainings, except for the NGM method, where we observe that $||\nabla X_{\text{NPF}}||^2$ is higher than the value obtained from nominal training. This effect is likely due to the specific ϵ_i per feature, which may not be well-suited for the NPF features. Finally, we can observe that the gradients from R-NGM are the lowest, which aligns with the robustness observed in the b- and c-tagging performances.

However, before concluding that the link between low gradients and reduced sensitivity to input feature changes demonstrates the observed robustness gains, we must ensure that our model does not suffer from gradient masking (GM). This is a phenomenon in which adversarial training, particularly those based on single-step white-box attacks like FGSM, produces gradients that are too small or obscure the model's actual sensitivity to certain variables [217, 219, 220]. These gradients can lead to inappropriate attacks, creating the

false impression of robustness gains in our models when, in reality, no such robustness has been achieved.

In our situation, the use of the gradient norm u_{adv} for both the NGM and R-NGM methods ensures that we produce attacks of the same magnitude, intended to maintain the same maximum attack power on the hyper ball $\mathbb{B}_{(x,1)}$ under the linear approximation of the gradients. Despite this, it is not guaranteed that such attacks cannot suffer from some form of obfuscation during training.

To ensure that our models have not learned to defend against white-box attacks by producing gradients that result in less effective attacks, we will evaluate a black-box attack, which does not rely on the weights and gradients of the model being tested for robustness. The method employed is the transfer attack [203], which involves using an external model to derive the gradients and the adversarial samples and evaluate the performance obtained with our black-box model being the NGM or R-NGM model. In our setup, we will use our nominal training to derive the adversarial examples x_{adv} and evaluate their performance on our NGM and R-NGM models. The attacks employed are the NGM and R-NGM attacks, illustrated in Figures 4.10 and 4.11, respectively. We can observe that the reported robustness against the black-box attack is not degraded and is even better than the white-box version, as expected. Additionally, we see that the robustness of both adversarial models is significantly better than that of the nominal model, which served as our baseline in this case.

Thus, we can conclude that our adversarial training did not suffer from gradient masking that would artificially inflate performance by giving a false sense of robustness. These two training methods have strengthened our model, as evidenced by the results against black-box attacks. For such attacks, the R-NGM adversarial training method also demonstrated the best robustness, further solidifying its status as the current state-of-the-art in adversarial training for jet tagging. Additionally, the correlation between the L_2 norm of the gradients on input features $||\nabla X_{\text{NPF}}||^2$, reported in Table 4.1 and the robustness of the models, reported in Figures 4.10 and 4.11, was observed. With the rejection of the gradient masking hypothesis, we

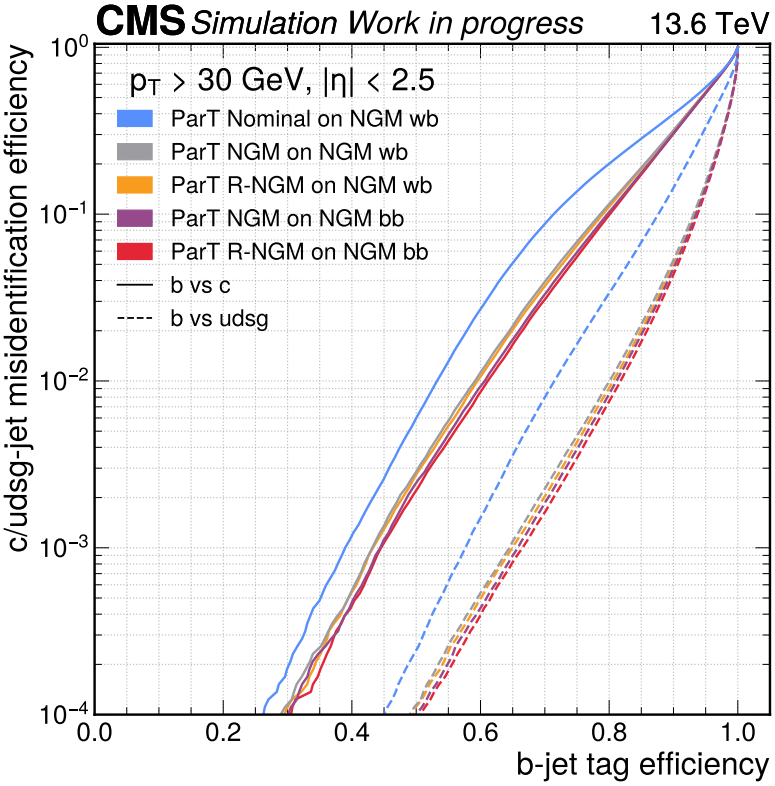

Figure 4.10: ROC curves performance on NGM white-box (wb) and black-box attacks (bb). The performance of the nominal ParT model on wb attacks (blue), of the R-NGM model on wb (orange) and bb (red) and the NGM model on wb (grey) and bb (purple) are shown. The performance of the b vs udsg (dashed lines) and b vs c (solid lines) rejections are shown.

can confidently conclude that the link between a low gradient norm and robustness against input changes is valid. These observations, combined with the success of the R-NGM method based on the link between adversarial training and the importance of input features, open new perspectives for the evaluation of robustness as well as potential improvements in the training method and even the attacks themselves.

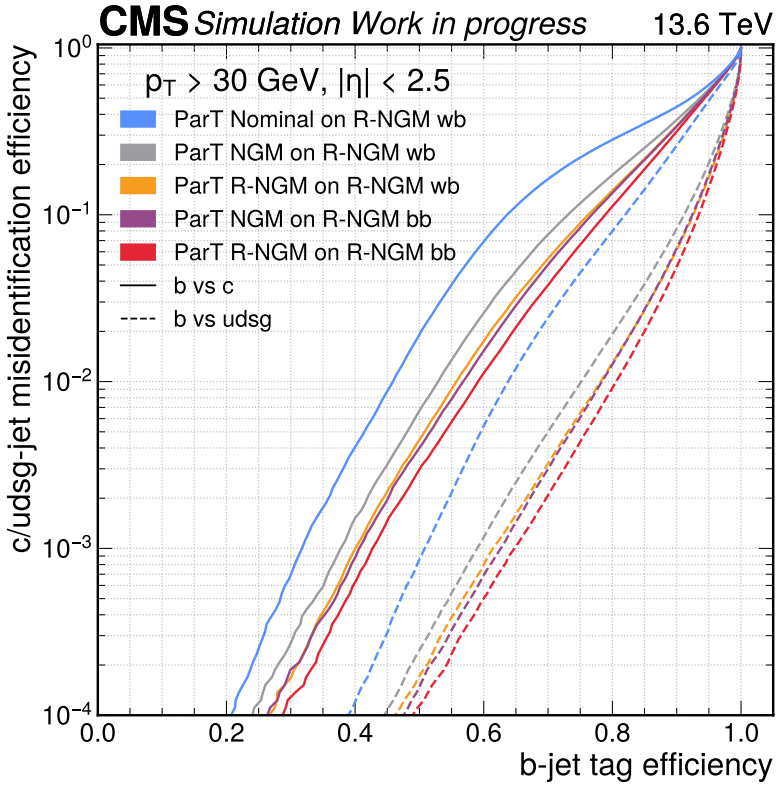

Figure 4.11: ROC curves performance on R-NGM white-box (wb) and black-box attacks (bb). The performance of the nominal ParT model on wb attacks (blue), of the R-NGM model on wb (orange) and bb (red) and the NGM model on wb (grey) and bb (purple) are shown. The performance of the b vs udsg (dashed lines) and b vs c (solid lines) rejections are shown.

4.4 Summary and outlook

4.4.1 Adversarial training for robust jet tagging algorithms

In this chapter, we explored the problem of applying jet identification algorithms to data from the perspective of neural networks and their training. After discussing the methods and challenges of jet calibration, we addressed the generalization capabilities from the neural network's standpoint, where we initially introduced the idea based on the principle of Sharpness-Aware Minimization [193], which aims to find a minimum of the cost function that is less sensitive to perturbations in the weights and biases of the neural network. However, this method is not well-suited to the problem we are addressing here. Therefore, we shifted the focus towards the objective of finding a flatter minimum, meaning one that is less sensitive to perturbations, this time concerning perturbations in the input features without introducing any information about the mismodelling itself, to derive a method solely from our training data that allows for improved generalised robustness.

Next, we introduced the concept of adversarial attacks, which addresses the previously formulated objective by considering the perspective of perturbations on input features, such as maximising them with respect to the gradient propagated to them. These methods maximise the prediction penalty imposed on the model by inducing minimal perturbations, with the magnitude being fixed. After introducing the concept, we developed the most commonly employed adversarial attack methods, focusing both on the development of the perturbations and the mathematical problems that these perturbations seek to maximise or minimise, as well as the impact on the computational cost that they induce. Finally, we introduced an initial adversarial training attempt applied to jet flavour identification based on the Fast Gradient Sign Method (FGSM) attack.

This training demonstrated that more robust models could be obtained, particularly against FGSM attacks. However, this initial

approach is not without limitations, primarily due to the reduced nominal performance compared to standard training and the inherent nature of FGSM attacks. The training method also highlighted certain constraints, suggesting that a more tailored cost function could yield better results. From this starting point, we introduced the Normed Gradient Method (NGM) attack and, later, its evolution, the Rectified Normed Gradient Method(R-NGM). Inspired by the relationship between the gradient on input features and feature importance, these methods allowed us to establish a link between a continuous attack based on a unit vector u_{adv}, which captures the significance of each variable of each constituent with respect to the cost function, and thus the quality of the prediction. The training was further enhanced by introducing a new hybrid cost function inspired by the TRADES method [198], which better balances nominal and adversarial performance by requiring our models to minimise the difference in predictions between nominal and adversarial examples.

We then evaluated the performance of our various training strategies. First, this initial evaluation allowed us to observe that we achieved the same nominal performance level for our adversarial trainings as the nominal training, ensuring the best possible performance on our unmodified MC simulation samples as intended. Secondly, we noted that both adversarial training methods improved the robustness of our models against the three types of white-box attacks considered: FGSM, NGM, and R-NGM. This highlights the enhancement in robustness of our models compared to the nominal training. Furthermore, we observed that the highest level of robustness was achieved by the R-NGM method for both b- and c-tagging, as summarised in Figures 4.12 and 4.13.

We then investigated the relationship between the gradient on the input features and the robustness obtained. First, we observed that our adversarial methods resulted in significantly lower L_2 norms of the gradients $||\nabla X_i||^2$, except in the case of $||\nabla X_{\text{NPF}}||^2$ for the NGM method. Once again, the R-NGM method exhibited the lowest gradients. On the one hand, adversarial training aims to reduce the model's sensitivity to variations in the input features. On the other hand, a lower gradient norm indicates that small changes in the input

have a reduced impact on the model's prediction, which is a sign of increased robustness. Therefore, these norms provided us with a new metric for evaluating the evolution of robustness. Given the evident link between our linear attack methods and the gradient, we also aimed to avoid the phenomenon of gradient masking, which could obscure the use of the gradient in white-box attacks and give us an illusion of robustness. In the case of gradient masking, the model would perform poorly against other types of attacks, such as black-box attacks. To ensure that we were not in this regime, we evaluated the robustness of our adversarial models through transfer attacks, using our nominal training to derive the attacks. The performance against these black-box attacks demonstrated excellent robustness for both of our adversarial training methods, ensuring, as expected, that the robustness and lower gradients were not due to gradient masking. In the face of transfer attacks, the R-NGM training again demonstrated the highest robustness.

Thus, we have successfully designed an adversarial training method that ensures the construction of more robust models against changes in input features while maintaining performance comparable to nominal training on our MC simulation samples. This training method aims to reduce the sensitivity to mismodelling in our MC simulation samples in a mismodelling agnostic way, thereby limiting its impact during calibration and, subsequently, in analyses. Robustness evaluations against both white-box and black-box attacks, as well as gradient evaluations with respect to the input features, all favoured the R-NGM training method, which currently represents the state-of-the-art in adversarial training. This highlights the crucial role that the gradient plays and its connection to input feature importance in determining the model's sensitivity. In the following chapter, we will employ this training method to develop the final model aimed at jet identification for the CMS experiment.

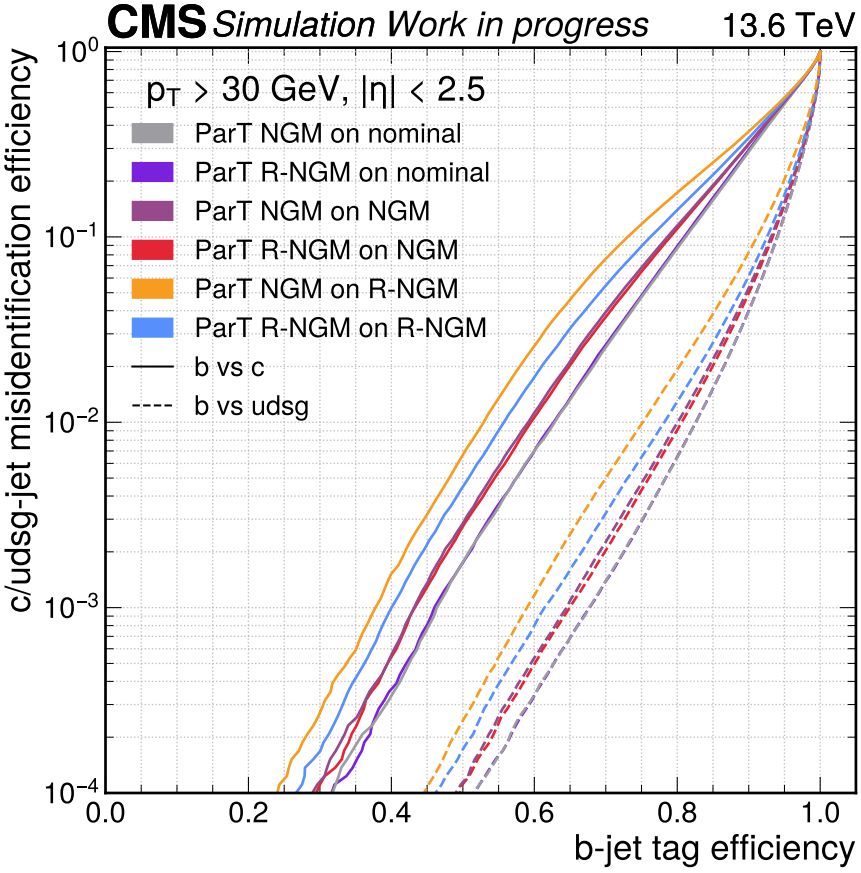

Figure 4.12: ROC curves performance of the b vs udsg (dashed lines) and b vs c (solid lines) rejection of the ParT algorithm trained in NGM mode and evaluated on nominal (grey), NGM (dark purple) and R-NGM (orange) and the ParT algorithm trained in R-NGM mode and evaluated on nominal (purple), NGM (red) and R-NGM (blue). The R-NGM training achieves better robustness than the NGM training.

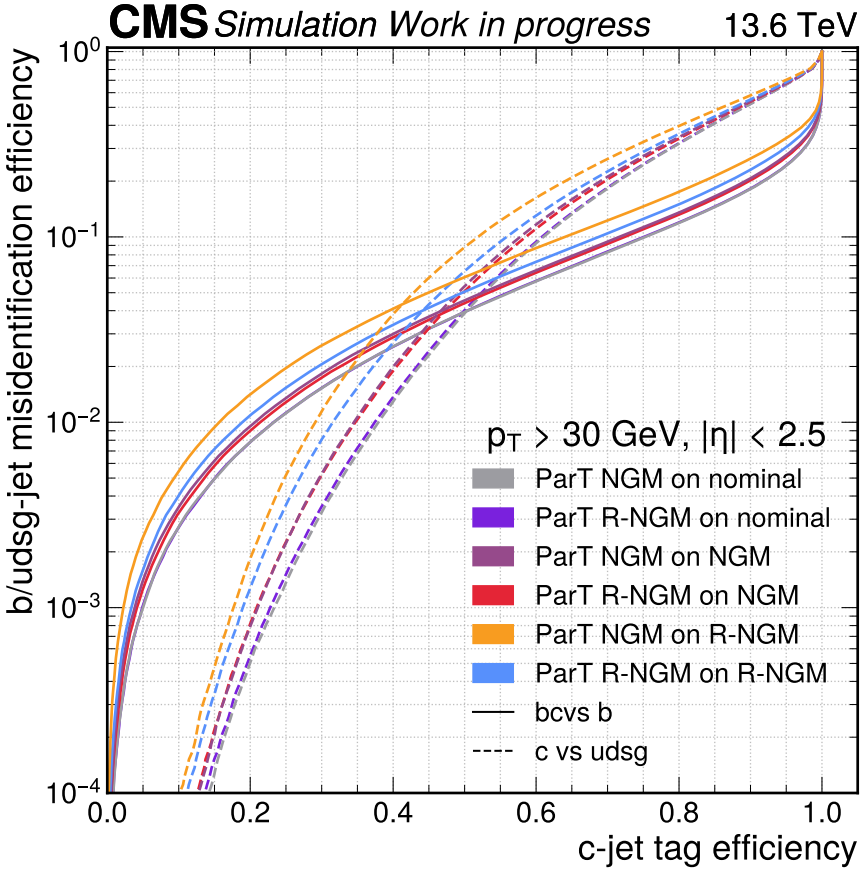

Figure 4.13: ROC curves performance of the c vs udsg (dashed lines) and c vs b (solid lines) rejection of the ParT algorithm trained in NGM mode and evaluated on nominal (grey), NGM (dark purple) and R-NGM (orange) and the ParT algorithm trained in R-NGM mode and evaluated on nominal (purple), NGM (red) and R-NGM (blue). The R-NGM training achieves better robustness than the NGM training.

4.4.2 Limitations of the method and outlook

Despite the R-NGM adversarial training method having demonstrated significant improvements in robustness and established itself as the state-of-the-art for training models against input feature perturbations, we can draw some limitations of the actual methodology. First, limitations lie in the method's dependence on the linear approximation of the gradients. While this approach has proven effective, it may not fully capture the non-linearities in more intricate models or scenarios where the input feature space is highly complex and relies on feature interdependence. A future development idea would be to extend the attack applied to the second order by using or not using the Hessian matrix [221,222]. We could also enhance the generalization capability of the model using other training methods, such as Sharpness-Aware Minimization. However, the first private efforts were conducted and have shown no substantial robustness gain when using SAM.

Additionally, another significant limitation of the method lies in its dependence on the intrinsic quality of the MC simulation samples. It is important to acknowledge that perfect robustness, capable of accurately predicting the class of jets, cannot be achieved if crucial variables exhibit distributions that significantly deviate from those modelled in the simulation samples. The method's objective has been to maximise robustness without compromising nominal performance, or in other words, to determine the maximum value of ϵ that can be employed without sacrificing performance on the simulated jets. If our simulated jets differ significantly due to substantial mismodelling effects, such as incorrect modelling of the hadronic shower or the hadronization process, we may encounter jets whose constituent nature varies significantly. This type of mismodelling cannot be captured by any adversarial attack.

Additionally, the current adversarial training paradigm might be less effective in reducing mismodelling. Indeed, the current objective is to determine the maximum ϵ value that can be used for a TRADES-type training [198] without sacrificing nominal performance. However, this conservative approach could limit the robust-

ness of our models and not be adapted to the training approach, as the training strategy employed is primarily designed for problems involving performance trade-offs. Modifying the training strategy with a new cost function and allowing the choice of an ϵ value more appropriate to the magnitude of changes induced by mismodelling on crucial variables, such as impact parameter features, could potentially further improve our adversarial training in the future.

Thus, even though the R-NGM method represents a significant advancement in reducing sensitivity to input features and thereby ensuring greater robustness of our models, it does not eliminate the need for calibration. Calibration remains necessary, as certain types of mismodelling cannot be properly addressed by the method proposed here. Furthermore, we could envisage improvements to the method to better capture nonlinear dependencies and ensure greater robustness in the future. Adversarial training with R-NGM is, therefore, an additional tool at our disposal. The very nature of R-NGM underscores the importance of focusing on the quality of jet simulation, as a reduction of the mismodelling produced here could also contribute to reducing the impact of calibration on final performance.

Chapter 5

Unified jet tagging algorithm

In this chapter, we will discuss the final components that led to the development of the latest model for jet identification in the CMS experiment. This new model is a unified jet tagging algorithm based on the Particle Transformer architecture. It is named Unified Particle Transformer (UParT) and extends the usual heavy-flavour tagging to the identification of hadronic taus as well as strange jets. Additionally, efforts have been made to incorporate energy regression and estimate its resolution to derive a flavour-aware jet energy correction. We will also employ adversarial training with R-NGM to achieve the most robust model possible. Therefore, in Section 5.1, we will address the elements necessary for extending the new classes and energy regression. Section 5.2 will describe the model training setup, while Section 5.3 will present and analyze the performance obtained on simulation samples, compared with previous algorithms, and the robustness when applied to data. We will conclude by summarising the results and evaluating the performance of this new algorithm in Section 5.4.

5.1 Extending the jet tagging algorithm

In this first part of the chapter, we will address the components involved in extending the jet tagging algorithms in the CMS experiment. We will detail the extension to hadronic taus in Section 5.1.1,

s jets in Section 5.1.2, and the regression of jet energy correction and resolution in Section 5.1.3.

5.1.1 Hadronic tau identification

The identification of τ leptons plays a crucial role in many analyses, such as studies of the Higgs boson through channels like $H \to \tau^+\tau^-$ [223, 224], searches for the production of Higgs boson pairs HH via a $b\bar{b}\tau^+\tau^-$ final state [225,226], and Beyond the Standard Model (BSM) searches [227, 228]. Approximately 35% of τ decays are leptonic, for which the final state is straightforward, consisting of an isolated electron or muon that is easily identifiable, along with an imbalance in the event's momentum, measured by the missing transverse momentum arising from the neutrinos in the leptonic decay channels.

The remaining 65% of τ decays are classified as hadronic decays, producing a final state composed of charged hadrons, neutral pions, and a neutrino ν_τ. These more complex final states, summarised in Table 5.1, are more challenging to identify and require distinction from jets originating from quarks or gluons, as well as from electrons and muons [229]. Similarly to the effort on inclusive ParticleNet mentioned in Section 3.2, these hadronic decays of τ leptons, denoted as τ_h, are the physical objects of interest in the context of an extension of jet flavour identification. We aim to extend the jet identification to include these τ_h decays. Thus, we will introduce new classes to our model's predictions for τ_h and for muons and electrons. These new classes are defined as follows:

- Muons are identified by matching the jet, within $\Delta R < 0.4$, with a generator-level muon originating from the decay of a H, Z, W boson or a τ lepton, with a transverse momentum $p_\mathrm{T} > 8$ GeV.
- Electrons are identified by matching the jet, within $\Delta R < 0.4$, with a generator-level electron originating from the decay of a H, Z, W boson or a τ lepton, with a transverse momentum $p_\mathrm{T} > 8$ GeV.
- τ_h are identified by matching the jet, within $\Delta R < 0.4$, with a

generator-level τ_h originating from the decay of a H, Z, W boson, with a transverse momentum $p_{\mathrm{T}} > 15$ GeV.

The labels for quarks and gluons remain consistent with the definitions established in the previous training setup described in Section 3.3.1. The labelled τ_h are then divided into subclasses based on the τ_h charge, the number of neutral pions, and the number of charged hadrons, commonly called 'prong.' We consider the following subclasses: one prong without a neutral pion, one prong with one neutral pion, one prong with two neutral pions, three prongs without any neutral pion, and three prongs with one neutral pion. These five subclasses, which can be linked to the five main hadronic decay channels noted in Table 5.1, are further separated based on the tau charge, resulting in a total of ten τ_h classes that will be used for our training. Additionally, we also consider the muon class and electron class as defined above. This leads to a total of twelve additional classes.

Decay mode	**Resonance**	B **(%)**
Leptonic decays		35.2 %
$\tau^- \to e^- \bar{\nu}_e \nu_\tau$		17.8
$\tau^- \to \mu^- \bar{\nu}_\mu \nu_\tau$		17.4
Hadronic decays		64.8 %
$\tau^- \to h^- \nu_\tau$	ρ(770 MeV)	11.5
$\tau^- \to h^- \pi^0 \nu_\tau$	ρ(770 MeV)	25.9
$\tau^- \to h^- \pi^0 \pi^0 \nu_\tau$	a_1(1260 MeV)	9.5
$\tau^- \to h^- h^+ h^- \nu_\tau$	a_1(1260 MeV)	9.8
$\tau^- \to h^- h^+ h^- \pi^0 \nu_\tau$		4.8
Other		3.3

Table 5.1: Decay modes and their branching ratios (B) for τ decays [14]. $h^\pm$ represent a charged hadron, also called prong

Historically used in the CMS experiment, the Hadron-Plus-Strips (HPS) algorithm facilitates the categorization of the τ_h decay channel and the reconstruction of the visible four-momentum by combining information from charged hadrons and neutral pions [230–232]. The HPS algorithm specifically targets the reconstruction of neutral pion decays, which decay into a pair of photons, each subsequently decaying into an e^+e^- pair. HPS then associates the charged

hadrons and neutral pions to assign a decay mode and reconstruct the four-momentum. In our approach, we propose, through our subclasses, an alternative to the HPS label, allowing the assignment of the decay mode and the charge.

To further motivate this approach, previous efforts to identify τ_h using neural networks, referred to as DeepTau [229], have demonstrated the utility of neural networks in such applications by improving identification efficiency by approximately 10-30% depending on the considered misidentification rate. Additionally, we have expanded the number of input features to enable our neural networks to effectively perform the task of identifying the newly introduced classes or improve the heavy-flavour tagging performance thanks to the tracker hit information [233]. These new variables are detailed in Appendix B under the label 'Training 2024.'

5.1.2 Strange jet identification

In the context of the effort to develop a unified jet tagger at the CMS experiment, an initiative was launched to explore the potential of s-tagging. Strange jets have received little attention at the phenomenological level regarding their behaviour in current colliders and detectors [234, 235]. This is mainly due to their primary signatures originating from charged kaons, which the general-purpose detectors at the LHC have difficulty exploiting, unlike certain previous detector designs [236]. However, exploring their potential could open up new prospects in the future, such as measuring the $|V_{ts}|$ term of the CKM matrix [237,238] via the $t \to W^+ s$ channel [239], or the Yukawa coupling of the s quark to the Brout-Englert-Higgs field [240]. This research and development initiative was also motivated by the ability to separately identify the flavour of the partons involved in the labelling process through 'ghost association', similar to the flavour definitions in Section 3.3.1. This allows us to distinguish between the uds classes and remain consistent with the definitions of heavy-flavour, gluon, and tau jet. We have chosen the following separation:

- Jets containing a strange parton as the hardest light-flavour ghost parton are labelled s jets.
- Jets containing a down parton as the hardest light-flavour ghost parton are labelled d jets.
- Jets containing an up parton as the hardest light-flavour ghost parton are labelled u jets.

In our goal to integrate s-tagging capabilities within our existing classification tools, we can already rely on the properties that differentiate heavy-flavour jets from light jets in the context of conventional b/c-tagging, as well as tau-tagging and the discrimination between light-flavour (uds) jets and gluon jets (g), also known as Quark/Gluon (QG) tagging [189, 241]. Thus, only the distinction between s jets from u and d jets remains to be explored. Recent phenomenological studies have investigated these capabilities [234, 235, 242], and we will explore here whether the key properties identified are present or accessible to aim for the best possible classification.

The identification and discrimination between u vs d (and by extension, u vs s) is achievable due to the difference in the charge of the initial quark, leading to variations in charge-related variables weighted by p_{T} [234], defined as:

$$Q_p = \frac{1}{(p_{\mathrm{T}}^{\mathrm{jet}})^p} \sum_{j \in \mathrm{jet}} Q_j (p_{\mathrm{T}}^j)^p \tag{5.1}$$

where $p_{\mathrm{T}}^{\mathrm{jet}}$ and p_{T}^j are the transverse momenta of the jet and the j-th constituent, respectively, and Q_j is the charge of the latter. The parameter p serves as a weighting factor for the variable. In our approach, we do not explicitly define this variable; instead, we provide our models with information related to the charges of the components and their transverse momenta, allowing the model to extract the necessary relationships. Beyond this discriminating power against u quarks alone, we also consider several properties stemming from the higher 'strangeness' content of s jets compared to d jets with

the same charge fraction. These properties and their distinguishing power also assist in u vs s tagging.

Among the key properties, the leading p_{T} constituent in s jets is generally a strange hadron, primarily kaons [235]. In contrast, in d jets, the hardest p_{T} constituent is typically a light hadron, most frequently a pion. This induces the following difference: the main constituents of s jets tend to have a higher transverse momentum than those of d jets, in line with the fact that kaons produced in pp collisions tend to have a higher transverse momentum. This observation is consistent with the measured fragmentation functions [243]. In addition, the number of charged constituents is also lower for s jets [235].

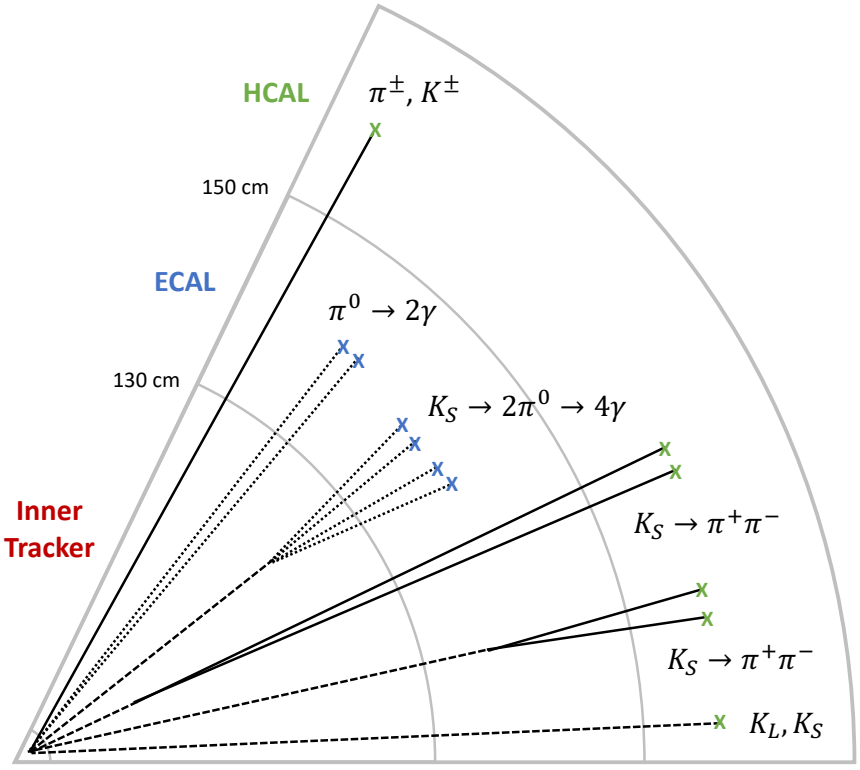

Figure 5.1: Sketch of pions and kaons in a hadronic collider [234]. The solid lines indicate charged particles, dashed neutral particles and dotted photons.

Among the main strange components observed in our jets, we find the charged kaons $K^{\pm}$ and the two neutral kaons K_L and K_S.

While there is no practical way to identify or distinguish $K^{\pm}$ from charged pions $\pi^{\pm}$ in CMS, we can exploit certain properties of the neutral kaons, particularly related to the K_S and its decays. The K_S, with a decay length of approximately $\sim$3 cm [234], often decays within the tracker into a pair of charged pions ($\sim$70%) or neutral pions ($\sim$30%). As illustrated in Figure 5.1, we could detect pairs of displaced pions. Although $\sim$30% of K_S decays result in π^0, which in turn decay into a pair of photons, the larger abundance of neutral pions in d jets leads to a higher abundance of photons in d jets compared to s jets. This property can be combined with the fact that s jets contain a larger energy fraction carried by neutral kaons. In comparison, d jets are more characterised by a higher fraction of neutral pions decaying into photon pairs. This difference provides another signature arising from the neutral component of the jets, resulting in a higher energy deposition in the electromagnetic calorimeter for d jets. In contrast, s jets will exhibit a larger deposition in the hadronic calorimeter [234]. We can observe this effect via the following quantity:

$$\Delta E = E_{\mathrm{NPF,ECAL}} - E_{\mathrm{NPF,HCAL}} \tag{5.2}$$

which computes the energy difference of the neutral particles in the ECAL and HCAL. Similarly to the case of Q_p, we provide the jet constituents' input features necessary for building it to the neural network. The neural network will then engineer the optimal selection and maximum distinction power.

Thus, we have explored the main discriminative variables that allow us to distinguish s jets from d and u jets. Figures 5.2 and 5.3 summarise the variables showing a distinction effect between s, d, and u jets with the input features considered in our training, except for ΔE and Q_p with $p = 1$ built from the input features for visualization purposes. As we can observe, the distinguishing power is reduced compared to properties derived from heavy-flavour jets. We can also observe that s jets have more distinction power against u jets thanks to the ΔE variable.

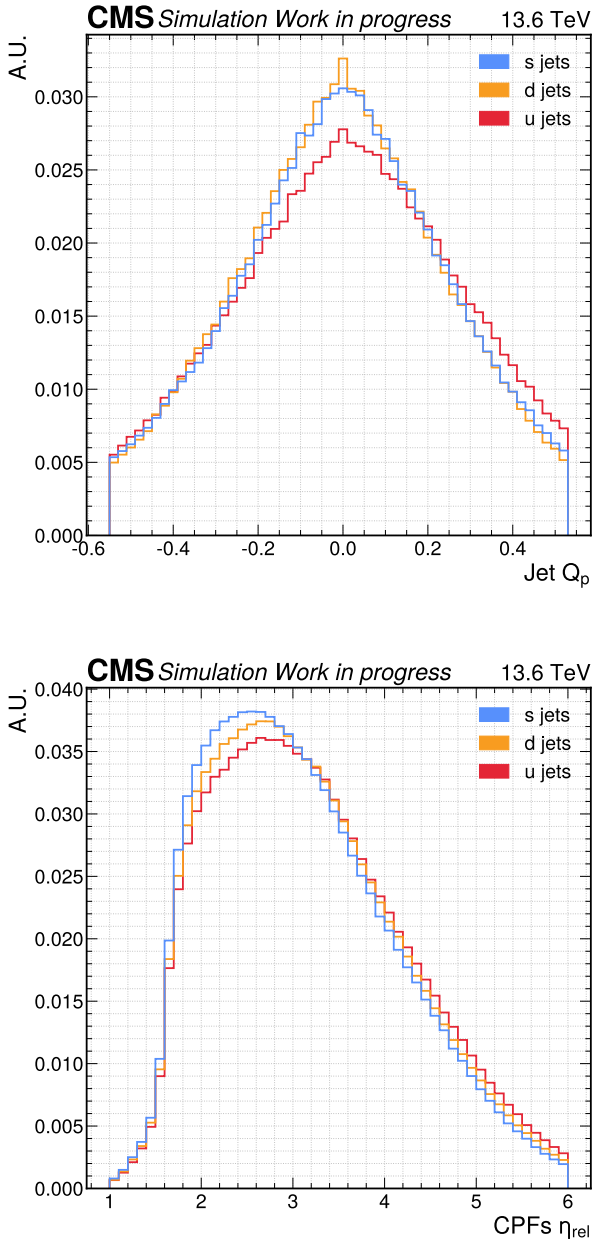

Figure 5.2: Distribution of Q_p, p=1, (above) and the relative η of the CPFs projected on the jet momentum axis (below) for the u (red), d (orange) and s (blue) jets.

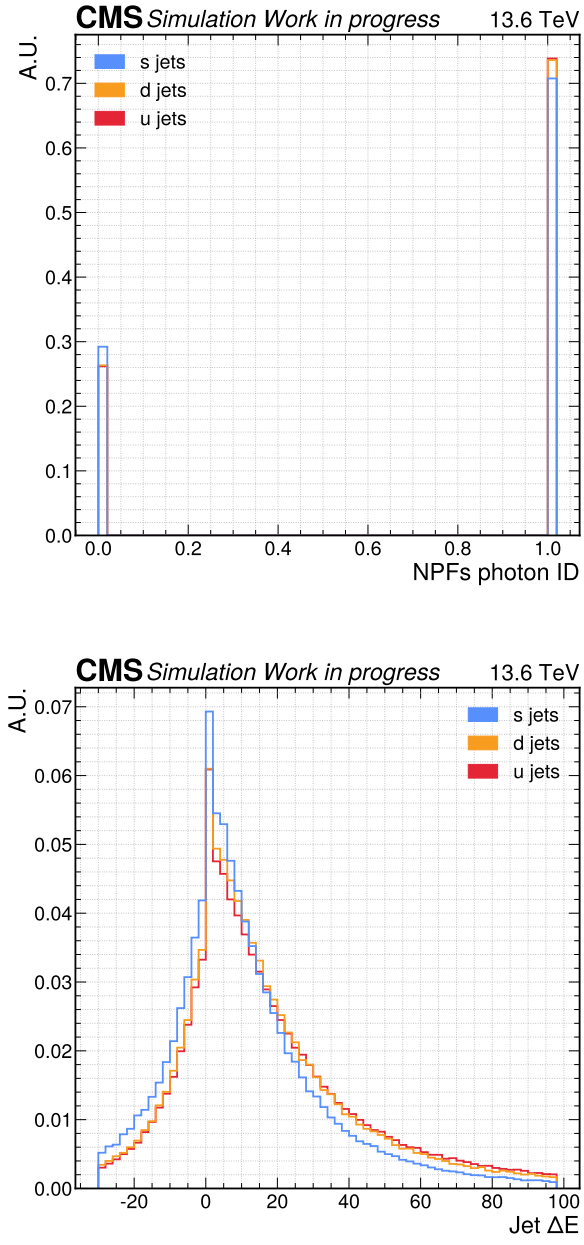

Figure 5.3: Distribution of the photon boolean ID (above) and ΔE (below) for the u (red), d (orange) and s (blue) jets.

5.1.3 Jet energy regression

In parallel with the extension of the classification task, we also expand the model's predictions to simultaneously perform regression and estimate the resolution of the jet transverse momentum p_{T} similar to the extended ParticleNet model. Our targets are correction ratios similar to the L2L3 JECs [70, 71]. For constructing those corrections, we define our target prediction to be $z = \frac{p_{\mathrm{T,\,gen}}}{p_{\mathrm{T}}}$ where p_{T} is the raw transverse momentum of our reconstructed jet, without any correction applied and $p_{\mathrm{T,\,gen}}$ is the transverse momentum of the generator-level jet associated with. Similar to the ParticleNet jet energy regression, we aim to predict the jet correction to the generator level, including or not the neutrino contribution, aiming to provide a more adequate jet energy correction for heavy-flavour jets decaying in a leptonic channel. To achieve this goal, we have designed a new initial loss function to accommodate the new objectives:

$$\begin{aligned} L_{\mathrm{incl}} = {} & CE(y, y_{\mathrm{truth}}) + \lambda_1 \cdot \log[\cosh(z - z_{\mathrm{truth}})] \\ & + \lambda_1 \cdot \log[\cosh(z_\nu - z_{\mathrm{truth},\nu})] + \lambda_2 \cdot (\rho_{0.16}(z_{16}) + \rho_{0.84}(z_{84})) \end{aligned} \tag{5.3}$$

$$\rho_\upsilon(z) = \begin{cases} \upsilon z, & \text{if } z > 0 \\ (\upsilon - 1)z, & \text{otherwise} \end{cases} \tag{5.4}$$

where $CE(y, y_{\mathrm{truth}})$ represents the cross-entropy typically used for classification, with y and y_{truth} denoting the predicted class and the true class, respectively. The terms z, z_ν, z_{16} and z_{84} represent the predicted jet p_{T} with and without the neutrino contribution and the 16th and 84th regressed quantiles, respectively, while z_{truth} and $z_{\mathrm{truth},\nu}$ represent the true value of the correction to be applied for the jet p_{T} with and without the neutrino contribution. The $\log\cosh$ loss function is used to perform the regression of this ratio. $\rho_{0.16}(z)$ and $\rho_{0.84}(z)$ represent the quantile regression at the 16th and 84th quantiles of the distribution of z, thus enabling the estimation of the resolution σ under the assumption of a normal distribution for the target jet energy correction z_{truth}. The quantile regression func-

tion is defined by Equation (5.4). Two parameters, λ_1 and λ_2, are introduced to control the magnitude of each loss function and, consequently, their contribution, ensuring the optimal convergence of the three tasks of classification, regression and resolution estimation.

5.2 Unified jet algorithm training

The training setup used for this inclusive training is a modified version of the setup described in Section 3.3, incorporating the new lepton classes as well as the target z_{truth} for regression and resolution estimation. The loss function has also been adjusted accordingly to align with our objectives. We employ the state-of-the-art architecture model, Particle Transformer, utilising six Particle Attention blocks and two Class Attention blocks. The feature dimension is set to 128, and each SDPA uses eight heads. The output layer produces $N_{\text{class}} + 3$ predictions, where N_{class} corresponds to the number of predicted classes trained with cross-entropy, and the remaining three predictions are used for regression and the estimation of the 16th and 84th quantiles.

We use the inclusive loss function defined by Equation (5.3), with the parameters λ_1 and λ_2 set to 1 after conducting a grid search to determine the optimal values. This grid search led to the conclusion that the magnitude adjustment parameters did not result in any preferential outcomes, while extending the search to extreme values on the order of $\mathcal{O}(10)$ and $\mathcal{O}(10^{-2})$ led to suboptimal regimes. The training is performed adversarially to enhance the model's robustness. The adversarial training utilises the R-NGM attack, where the gradients is derived with respect to the cross-entropy for simplicity. The magnitude of the attacks, ϵ, was also optimised, with the optimal value found to be 0.001 for obtaining the same nominal performance as the nominal training. In adversarial mode, the final loss function used for training is defined as follows:

$$L_{\text{adv}} = \begin{cases} L_{\text{incl}}(y, y_{\text{truth}}, z, z_{\text{truth}}) \text{ if nominal mode} \\ L_{\text{incl}}(y_{\text{adv}}, y_{\text{truth}}, z_{\text{adv}}, z_{\text{truth}}) + \lambda \cdot KL(y, y_{\text{adv}})) \\ \qquad\qquad \text{otherwise} \end{cases} \tag{5.5}$$

To achieve the best possible training, we extended the data sources processes to enrich our dataset with the leptons we aim to identify and a better covering of the $p_{\text{T}} > 300$ GeV regime. We also decided to cover a broader range of processes and topologies, paying particular attention to sources of jets with higher energies, specifically those with $p_{\text{T}} > 300$ GeV and even very high $p_{\text{T}} > 1$ TeV. The maximum jet p_{T} covered, previously set at 1 TeV, is now extended to 2 TeV for improving both the jet classification and jet energy regression in this phase space. The minimum jet p_{T} value is conserved at 15 GeV. The sample list can be found in Appendix C, in the section titled Training 2024. In addition to the usual 2D reshaping, we applied a maximum ratio of 2.5 concerning the b class to prevent our dataset from having excessive class imbalance. The final training set consists of approximately 30 million jets. The training was performed over 30 epochs using the optimizer and hyperparameters described in Section 3.3.

We also use the latest version of the PUPPI algorithm for pileup mitigation optimised for τ_h tagging [244]. It has been observed that previous versions of PUPPI resulted in an inefficiency in the reconstruction of τ_h at low p_{T}. The primary cause stems from a selection criterion on the transverse momentum of charged particles not used in any vertex fit. The reduction of this criterion from 20 GeV to 4 GeV was selected for its ability to recover the efficiency loss without causing significant degradation in other jet-reconstructed variables [244].

5.3 Performances

This section compares the performance of the new UParT tagger with existing models for AK4 jet identification, including ParticleNet, RobustParT, and DeepJet. The ParticleNet model is the latest inclusive version, as outlined in Section 3.2. RobustParT is the Particle Transformer model trained with adversarial NGM but without the trade-off loss function [198], as discussed in Section 3.3. DeepJet [110] is re-trained to meet Run 3 requirements at the CMS experiment, following the same setup as RobustParT. In Section 5.3.1, we will compare the taggers' performance in heavy-flavour tagging with tau-tagging and regression evaluated for ParticleNet and UParT. s-tagging performance will be assessed solely for UParT, as no previous models have performed this task. Section 5.3.2 will evaluate the robustness of UParT's training, confirming the anticipated robustness gain from the R-NGM method.

5.3.1 Comparison with previous AK4 jet tagging algorithms at the CMS experiment

Heavy-flavour tagging performance

The performance of heavy-flavour tagging will be evaluated using two test datasets: the first consisting of fully hadronic $t\bar{t}$ events, and the second of QCD multijet events. The first dataset will allow us to assess performance within the typically evaluated p_T range of 20 to 300 GeV [61, 110]. In contrast, the second set will enable the evaluation of jets with higher transverse momentum, using a minimum p_T value of 300 GeV. All algorithms will evaluate the b vs all performance as previously defined, which corresponds to the following probabilities for the inclusive models:

$$\begin{aligned} prob_{\text{b vs all}}^{\text{ParticleNet}} &= \frac{prob_b}{prob_g + \sum_{q \in \text{Quark}} prob_q} \\ prob_{\text{b vs all}}^{\text{UParT}} &= \frac{prob_b + prob_{bb} + prob_{lepb}}{prob_g + \sum_{q \in \text{Quark}} prob_q} \end{aligned} \tag{5.6}$$

where we exclude the probabilities related to τ_h, muons, and electrons from the discriminant calculation to reconstruct the same discriminant as previously employed. ParticleNet contains only one b class while UParT considers the b, bb and lepb class defined in Section 3.3. For charm tagging, the definitions of c vs b and c vs udsg used earlier are maintained and require no modification.

We can observe in Figure 5.4 that UParT demonstrates superior performance both in c and udsg rejection. For the udsg rejection, the performance is equivalent to ParticleNet until tight rejection ($< 1\%$ misidentification) for $t\bar{t}$ events, while it also demonstrates superior performance for QCD multijet events. Evaluated across two different energy scales, UParT sets a new state-of-the-art performance for b-tagging within the CMS experiment. The performance gain is particularly significant for c-jet rejection, both in $t\bar{t}$ events and QCD multijet events. The efficiency values at different WPs are reported in Tables 5.2 and 5.3.

Model	Loose WP eff	Medium WP eff	Tight WP eff
DeepJet	79.90% - 71.89%	59.19% - 45.43%	41.47% - 24.04%
RobustParT	82.32% - 78.13%	63.79% - 56.47%	46.36% - 37.45%
ParticleNet	84.31% - 79.61%	66.18% - 58.02%	48.05% - 38.07%
UParT	85.02% - 80.79%	68.42% - 60.96%	51.82% - 42.17%

Table 5.2: b vs c efficiency at the loose, medium and tight WP. The $t\bar{t}$ (first value) and QCD multijet (second value) events are evaluated.

Model	Loose WP eff	Medium WP eff	Tight WP eff
DeepJet	94.00% - 88.54%	84.40% - 71.82%	69.92% - 50.89%
RobustParT	94.77% - 90.82%	86.09% - 78.31%	73.93% - 62.68%
ParticleNet	95.49% - 92.87%	86.91% - 80.87%	73.45% - 63.37%
UParT	95.45% - 92.89%	86.82% - 81.50%	74.74% - 65.13%

Table 5.3: b vs udsg efficiency at the loose, medium and tight WP.The $t\bar{t}$ (first value) and QCD multijet (second value) events are evaluated

For c-tagging performance, as illustrated in Figure 5.5, we can observe that UParT also demonstrates superior performance compared to previous taggers. However, the improvement gain is smaller than observed across the ROC curves for b-tagging. Nevertheless, the improvement remains significant, especially at high p_{T}, as reported

in the efficiency Tables 5.4 and 5.5.

Figures 5.6, 5.7 and 5.8 show the b jet efficiency of our taggers as a function of the jet p_{T} and jet η using QCD multijet samples for loose, medium and tight WP, respectively. We can observe that UParT consistently shows higher efficiency than the previous state-of-the-art ParticleNet. We can also notice that ParticleNet performs worse than RobustParT in the high jet p_{T} region (> 300 GeV) for the medium WP and both in the jet p_{T}, except the first low p_{T} bin, and jet η for the tight WP. This confirms the performance of UParT, while the previous state-of-the-art, ParticleNet, indicates anomalous behaviour in the jet p_{T} and jet η b jet efficiency compared to the expected performance of ROC curves.

We have succeeded, with the aid of new input variables and a six-layer model, in achieving the expected new best performance by surpassing the previous state-of-the-art ParticleNet and the earlier Particle Transformer model. In addition to the new classes and the task of energy regression, we were able to create a model that performs effectively in heavy-flavour tagging, achieving a new state-of-the-art in comparison with previous models.

Model	Loose WP eff	Medium WP eff	Tight WP eff
DeepJet	72.79% - 58.49%	21.47% - 9.16%	4.11% - 0.81%
RobustParT	78.01% - 67.77%	25.01% - 14.03%	5.32% - 2.63%
ParticleNet	80.63% - 71.11%	25.90% - 14.12%	5.35% - 2.13%
UParT	82.00% - 73.05%	27.94% - 17.73%	6.08% - 4.14%

Table 5.4: c vs b efficiency at the loose, medium and tight WP.

Model	Loose WP eff	Medium WP eff	Tight WP eff
DeepJet	58.97% - 41.99%	37.82% - 22.38%	23.91% - 12.45%
RobustParT	62.06% - 47.29%	41.34% - 29.04%	27.56% - 18.22%
ParticleNet	64.64% - 57.50%	44.45% - 36.81%	29.62% - 23.19%
UParT	64.28% - 59.05%	44.77% - 38.27%	30.85% - 24.92%

Table 5.5: c vs udsg efficiency at the loose, medium and tight WP.

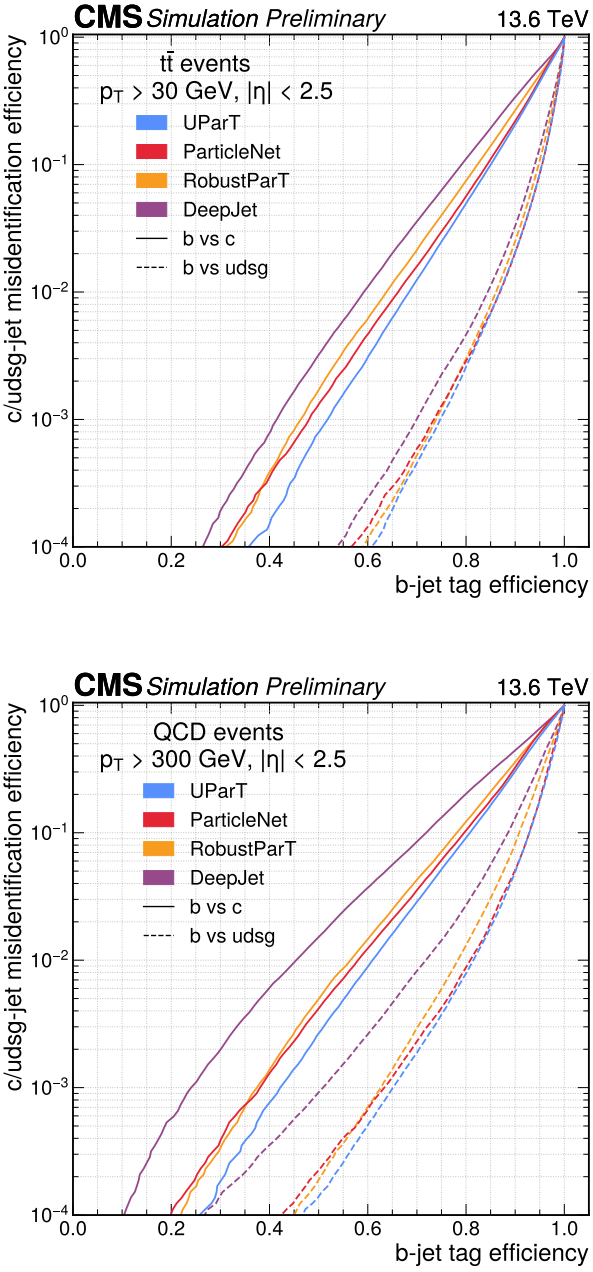

Figure 5.4: ROC curves of the b-tagging performance, evaluated on $t\bar{t}$ events (above) and QCD multijet (below), of DeepJet (purple), RobustParT (orange), ParticleNet (red) and UParT (blue). The dashed line represents the udsg jet rejection while the solid line represents the c jet rejection [186].

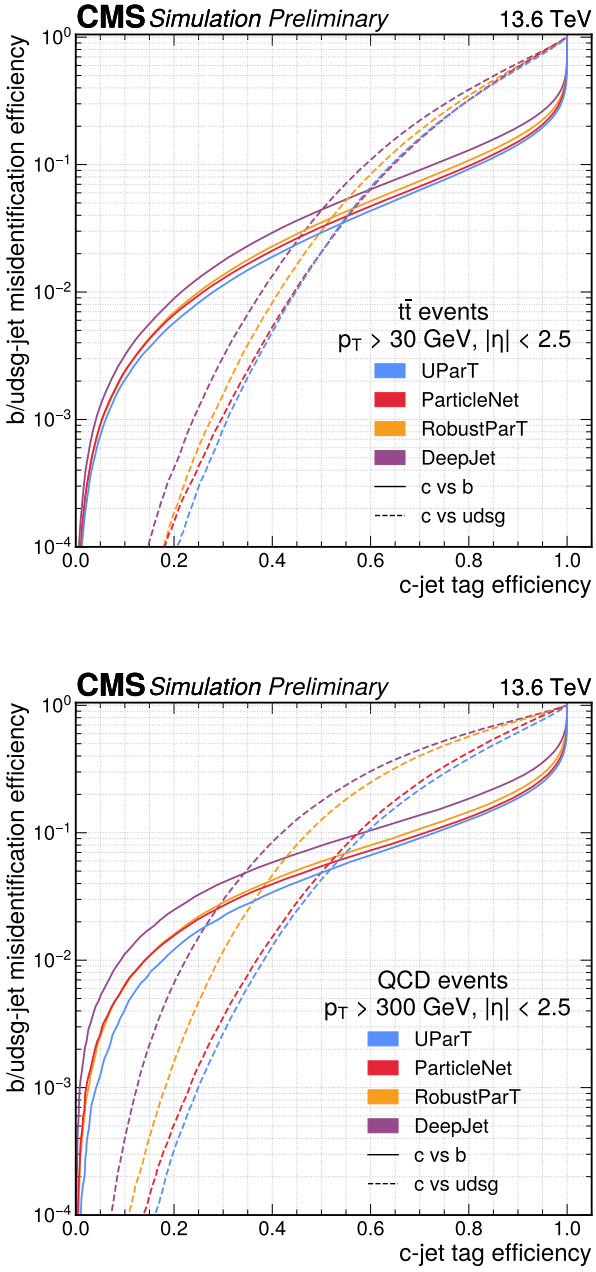

Figure 5.5: ROC curves of the c-tagging performance, evaluated on $t\bar{t}$ events (above) and QCD multijet (below), of DeepJet (purple), RobustParT (orange), ParticleNet (red) and UParT (blue). The dashed line represents the udsg jet rejection while the solid line represents the b jet rejection [186].

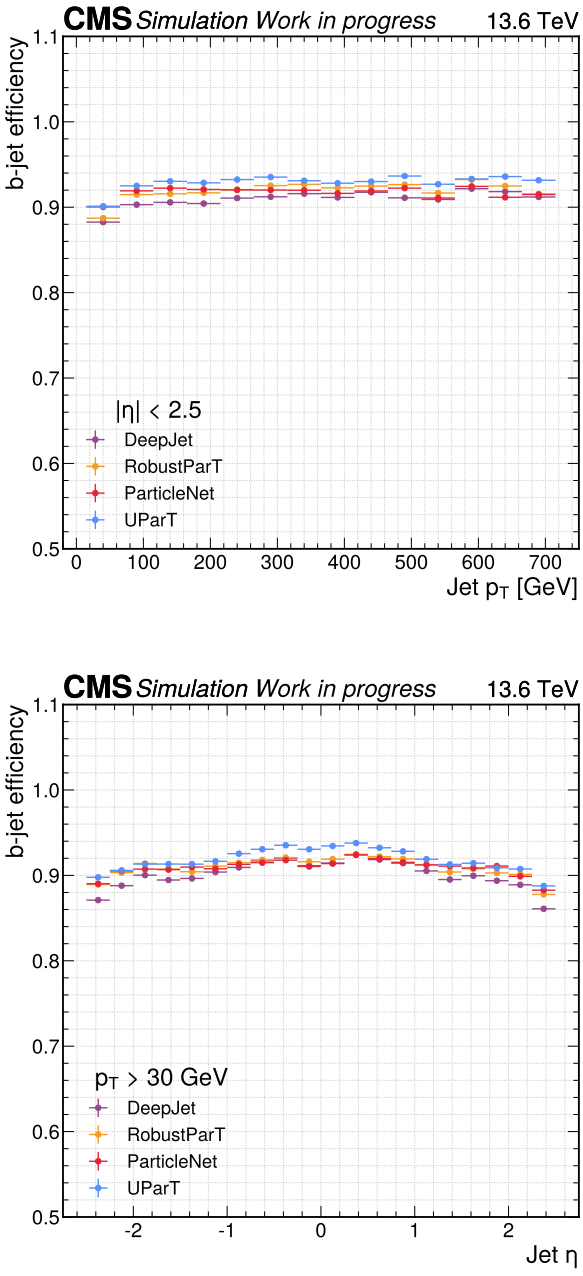

Figure 5.6: Performance of DeepJet (purple), RobustParT (orange), ParticleNet (red) and UParT (blue) as a function of the jet p_{T} (above) and jet η (below) for the loose WP (10% light misidentification rate) using QCD multijet.

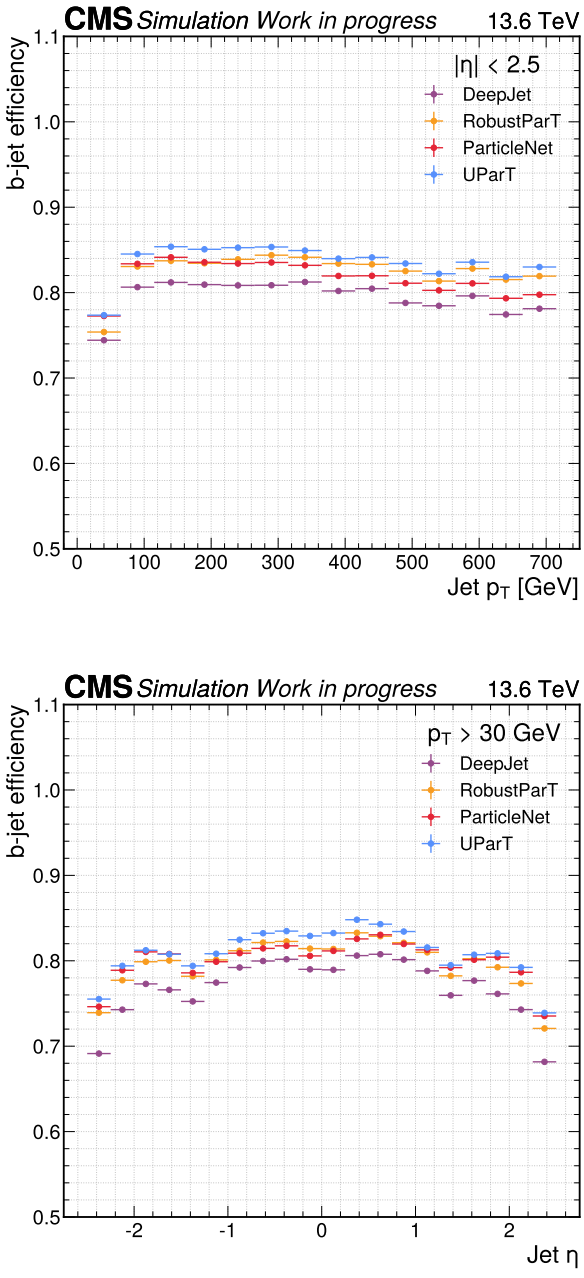

Figure 5.7: Performance of DeepJet (purple), RobustParT (orange), ParticleNet (red) and UParT (blue) as a function of the jet p_{T} (above) and jet η (below) for the medium WP (1% light misidentification rate) using QCD multijet.

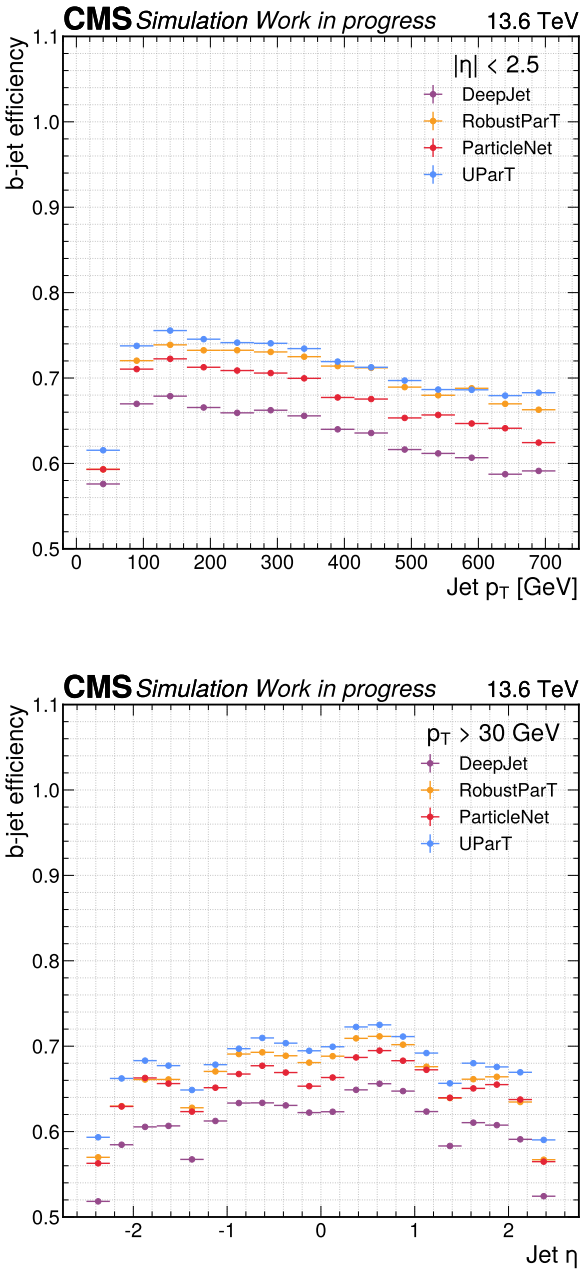

Figure 5.8: Performance of DeepJet (purple), RobustParT (orange), ParticleNet (red) and UParT (blue) as a function of the jet p_T (above) and jet η (below) for the tight WP (0.1% light misidentification rate) using QCD multijet.

τ_h jet tagging performance

The performance of τ_h-tagging is evaluated using Drell-Yan (DY) events, where we utilise the maximum classification capability between τ_h and each other class, referred to as $prob_{\tau_h \text{ vs X}}$ and defined by Equation (5.7), where X represents the class against which we want to distinguish τ_h. We compare the performance of our UParT model against the ParticleNet model, the first and precursor of inclusive taggers.

$$prob_{\tau_h \text{ vs X}} = \frac{\sum_{i \in \tau} prob_i}{\sum_{i \in \tau} prob_i + prob_X} \tag{5.7}$$

As observed in Figure 5.9, ParticleNet and UParT exhibit similar performances. ParticleNet performs better at higher misidentification rates, whereas UParT excels at lower rates. This is likely due to two factors identified after UParT's training and implementation in CMSSW. Firstly, ParticleNet employs a variable that matches its constituents with the components of τ_h candidates reconstructed by the HPS algorithm [231, 232]. Secondly, an error was introduced in the construction of the LostTracks collection. This error originates from improper access to the position indices of the LostTracks after they have been sorted by transverse impact parameter significance in our ntupler [166]. This error resulted in accessing incorrect memory, leading to the creation of erroneous variables. A cleaning of the anomalous LostTracks has been applied during UParT's training. Consequently, this issue on important variables, specifically for τ_h-tagging performance, led UParT performance to be suboptimal compared to ParticleNet, explaining the observed performances here.

A retraining of UParT, including the missing variables and resolving the encountered issues, would be feasible to improve the performance of UParT in τ_h-tagging in a future version.

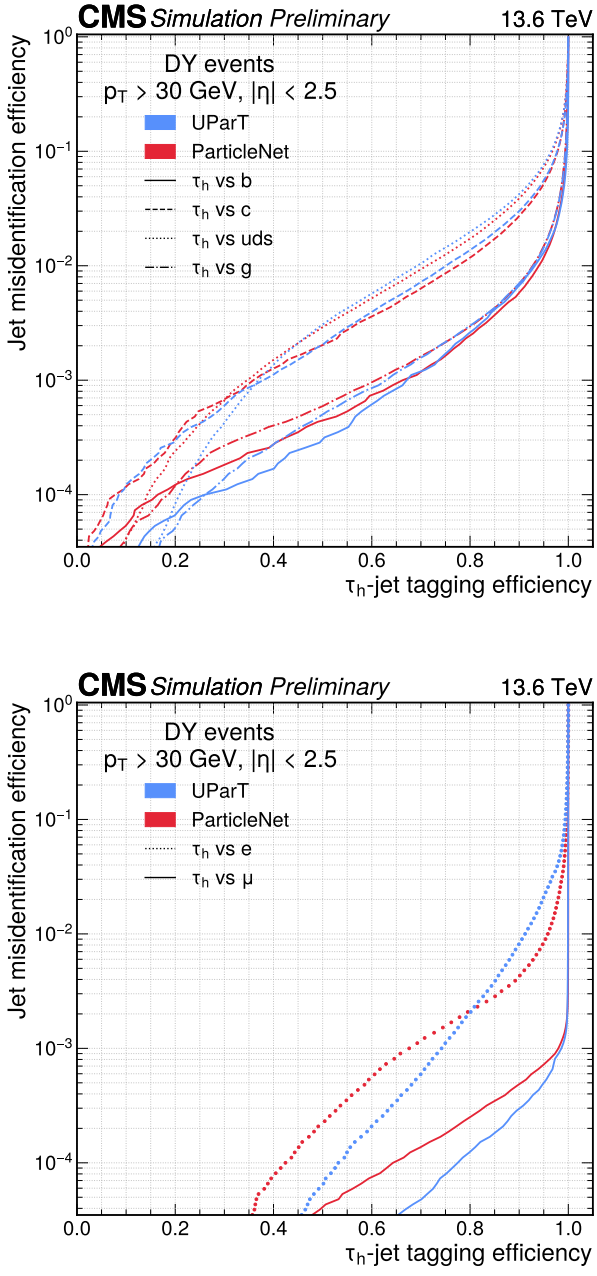

Figure 5.9: ROC curves of the τ_h-tagging performance, evaluated on DY events, of ParticleNet (red) and UParT (blue). The upper plot evaluates the performance against quarks and gluon jets, and the lower plot against leptons [186].

Strange jet tagging performance

The performance of s-tagging will be evaluated on $t\bar{t}$ events and QCD multijet events, shown in Figure 5.10, where we utilise the maximum classification capability between s jets and each other class, similarly to the τ_h performance evaluation. The discriminators of the s-tagging performance are defined as:

$$prob_{\text{s vs X}} = \frac{prob_s}{prob_s + prob_X} \tag{5.8}$$

Firstly, we observe that the most challenging discrimination is indeed s vs d, followed by s vs u, which benefits from better separation due to the difference in the charge fraction of the originating quark, resulting in a variation in the Q_p variable that the model has successfully captured. UParT can produce a low-efficiency s-tagger, achieving an efficiency of 19.72% and 24.59% on $t\bar{t}$ samples at the loose WP, as reported in Table 5.6. The performance against heavy-flavour and gluon jets is as expected, showing good rejection power for these classes.

Rejection	Loose WP eff	Medium WP eff	Tight WP eff
s vs d	19.72% - 19.96%	2.66% - 2.54%	0.37% - 0.25%
s vs u	24.59% - 32.32%	5.43% - 6.62%	0.98% - 1.44%

Table 5.6: s-tagging efficiency of UParT at the loose, medium and tight WP for the $t\bar{t}$ (first value) and QCD multijet (second value) for the u and d rejections.

Given the performances obtained, our model exhibits two performance regimes in s-tagging. The first regime, against first-generation quarks u and d, allows for the first time to individually isolate s jets with a low efficiency, approximately 20% for a misidentification rate of 10%. On the other hand, we have the regime against heavy quarks b and c and gluon jets. The distinction against these is well-known in the CMS experiment, and the rejection at a 10% misidentification rate exceeds 50% for each class. Thus, analogous to the evaluation of c-tagging performance, we define the following discriminants to separate the classification into s vs ud and s vs bcg:

$$
\begin{aligned}
prob_{\text{s vs ud}} &= \frac{prob_s}{prob_s + prob_u + prob_d} \\
prob_{\text{s vs bcg}} &= \frac{prob_s}{prob_s + prob_b + prob_{bb} + prob_{lepb} + prob_d + prob_g}
\end{aligned}
\tag{5.9}
$$

The results obtained by these two regimes are summarised in Table 5.7 and Figure 5.11. We observe that we have indeed established two distinct regimes as expected, with the ability to perform s-tagging with an efficiency of 22.28% for the loose WP of the ud rejection. Thus, we have successfully demonstrated that the training of UParT has successfully constructed an s-tagger, which should allow us to establish its calibration and explore its capabilities in analyses.

Rejection	Loose WP eff	Medium WP eff	Tight WP eff
s vs ud	22.28% - 24.47%	3.54% - 3.54%	0.49% - 0.39%
s vs bcg	70.73% - 62.27%	21.30% - 17.00%	4.11% - 2.89%

Table 5.7: s-tagging efficiency of UParT at the loose, medium and tight WP for the $t\bar{t}$ (first value) and QCD multijet (second value) for ud and bcg rejections.

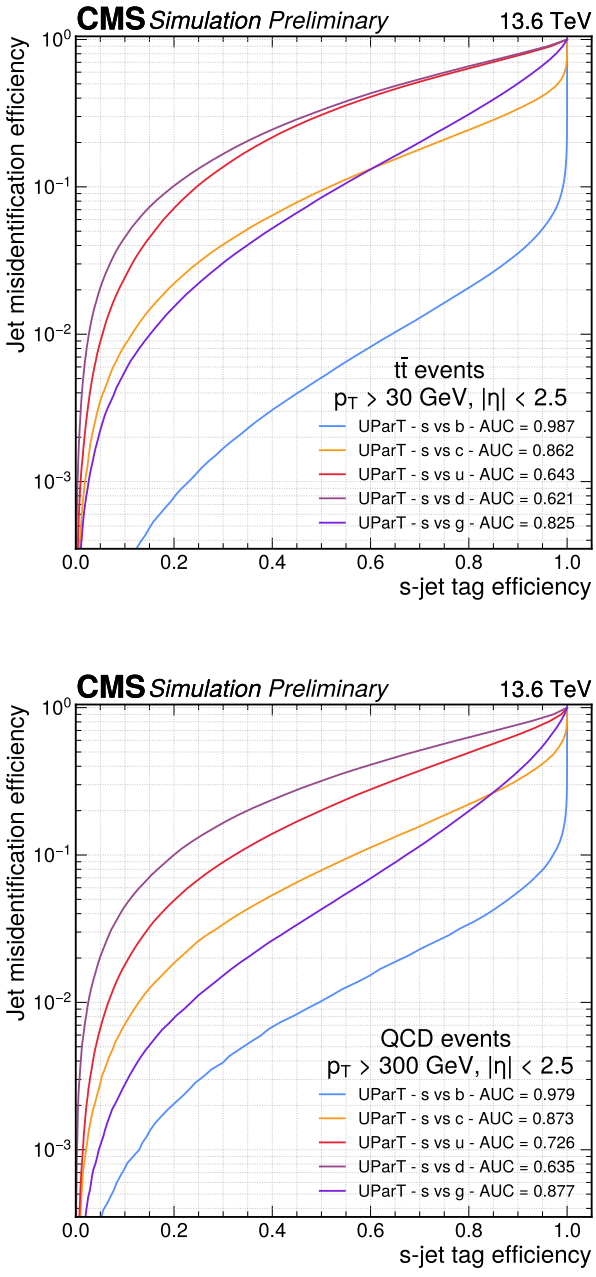

Figure 5.10: ROC curves of the s-tagging performance, evaluated on $t\bar{t}$ events (above) and QCD multijet (below) [186].

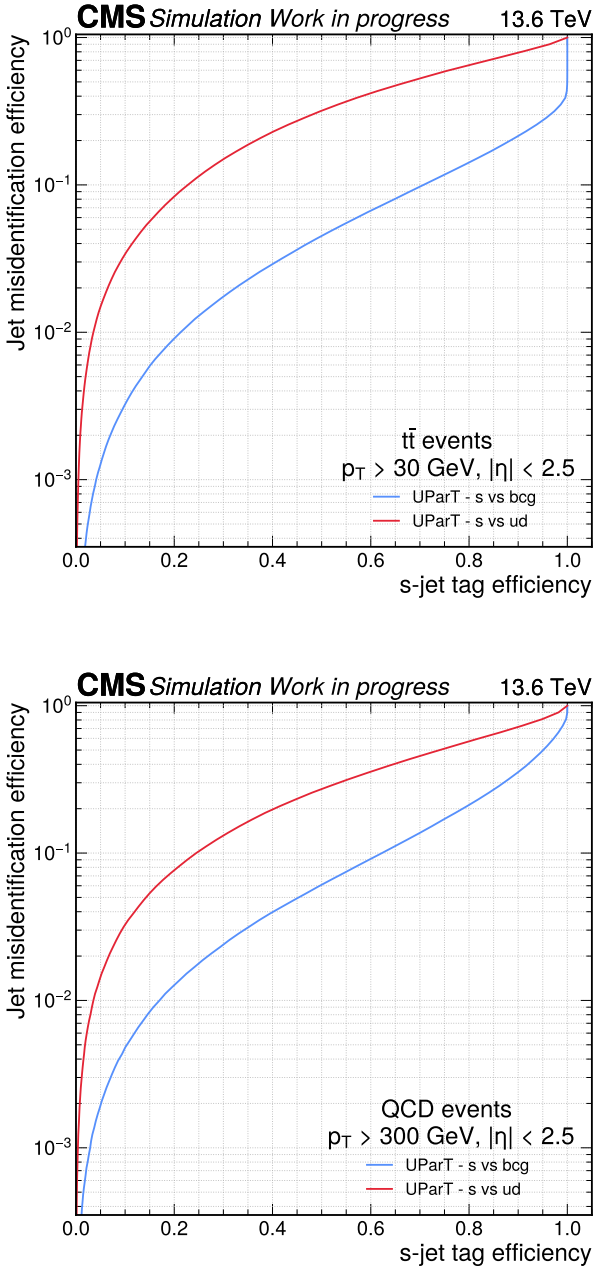

Figure 5.11: ROC curves of the s-tagging performance, evaluated on $t\bar{t}$ events (above) and QCD multijet (below). The red curve represents the ud rejection, and the blue the bcg rejection.

Jet energy regression performance

We evaluate the performance of the regression task using $t\bar{t}$ events samples. We first assessed the performance of the regression of the ParticleNet and UParT algorithms by evaluating the median of the regressed response for the regression without and with the neutrino respectivelly: Median$\left(\frac{\text{regr}_{p_\text{T}} \cdot p_\text{T}^\text{uncor}}{p_\text{T}^\text{ptcl}}\right)$ and Median$\left(\frac{\text{regr}_{p_\text{T},\nu} \cdot p_\text{T}^\text{uncor}}{p_\text{T}^\text{ptcl,}\nu}\right)$. The predicted regressions without and with neutrinos are the regr_{p_T} and $\text{regr}_{p_\text{T},\nu}$ and p_T^uncor is the uncorrected transverse momentum of the reconstructed jet. The uncorrected transverse momentum of the generator-level jet associated with the reconstructed jet, including or not the neutrino contribution, are the terms p_T^ptcl and $p_\text{T}^\text{ptcl,}\nu$. The closer the median value of the response is to 1, the more accurately the model predicts the correction that needs to be applied to recover the transverse momentum that the jet possesses at the generator level. We illustrate the estimated resolution similarly through the median of the quantile regression.

Figures 5.13 and 5.14 represent the median of the regressed response, not including and including the neutrino contribution, respectively. We split the median response into the b, c, uds and g jet flavour categories to evaluate the performance of our regression obtained on the usual flavour labels used in heavy-flavour tagging. We can observe an improvement in the response, particularly for high values of $|\eta|$. This effect likely originates from the update of the PUPPI algorithm [147] and the changes in the Run 2 conditions between the simulation samples on which ParticleNet was trained while UParT was trained with the latest samples and conditions available. Additionally, regarding the resolution estimation, illustrated in Figure 5.15, UParT and ParticleNet exhibit a similar quantile regression response per jet. Thus, combining the new algorithm and using Run 3 MC utilising the latest version of PUPPI for training improves the jet energy response without affecting the quantile regression response.

It should be noted that the initial results of the ParticleNet regression were analyzed [147] and indicate an improvement in resolution of approximately 15% compared to the usual jet energy correction. These preliminary results, combined with the achieved im-

provement in the regression task by UParT, suggest that our jet algorithms' flavour-aware jet energy regression provides a consistent and promising energy correction [147]. This result underscores the value of extending our unified tagger through regression.

The quality of the regression for τ_h jets is illustrated in Figure 5.12, where UParT and ParticleNet perform similarly. It is worth noting that the regression of ParticleNet appears to have peaked at a response of 1, indicating that its quality remains better at correcting the momentum of the associated tau at the generator level. Thus, by highlighting once again the loss of performance in UParT, as detailed earlier, we observe that this also impacts the energy correction of τ_h, clearly illustrating the relationship and 'flavour-aware' nature of these corrections.

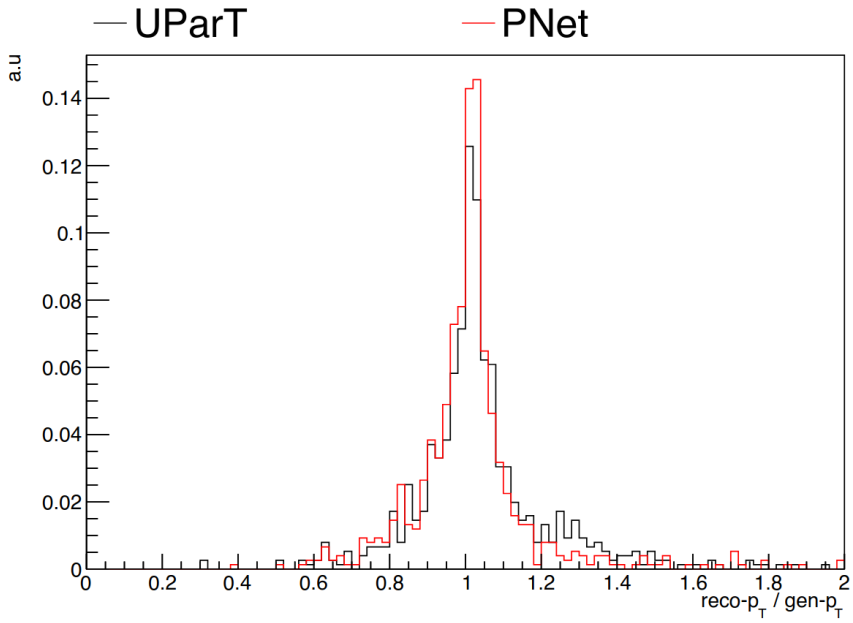

Figure 5.12: Regressed response for τ_h of ParticleNet (red) and UParT (black).

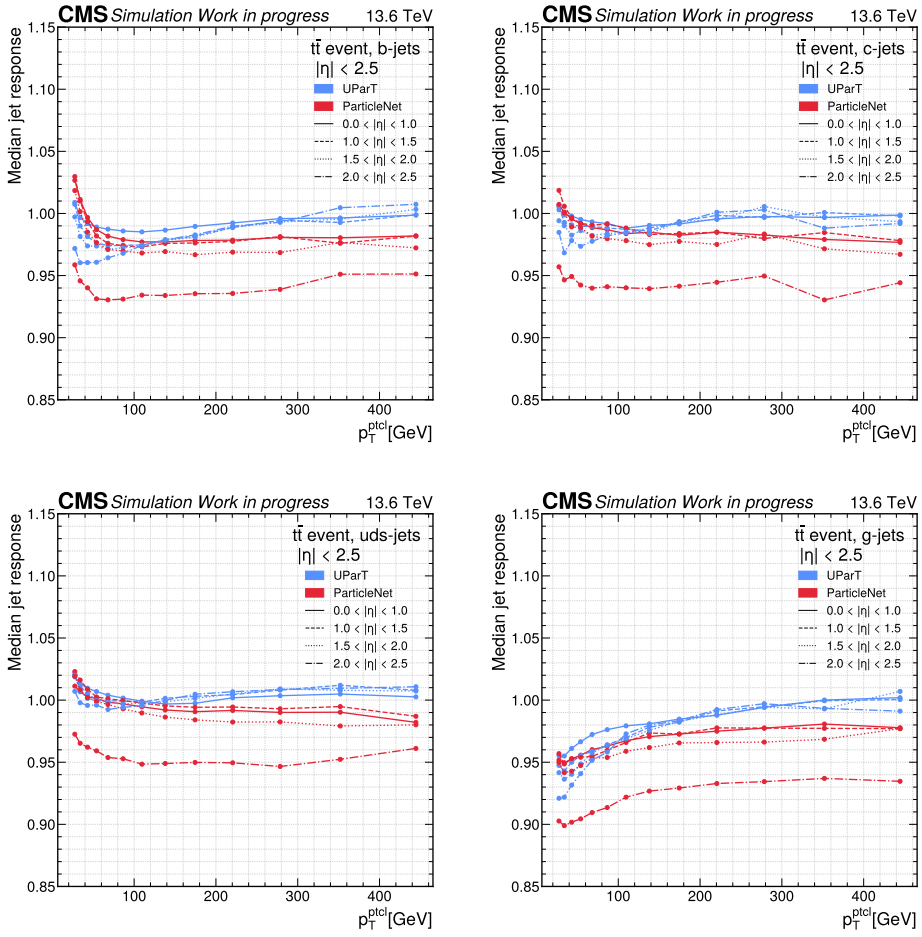

Figure 5.13: Median of the regressed response for b (upper left), c (upper right), uds (lower left) and g (lower right) jets. The performance of ParticleNet (red) and UParT (blue) are evaluated for different values of $|\eta|$ of the jet.

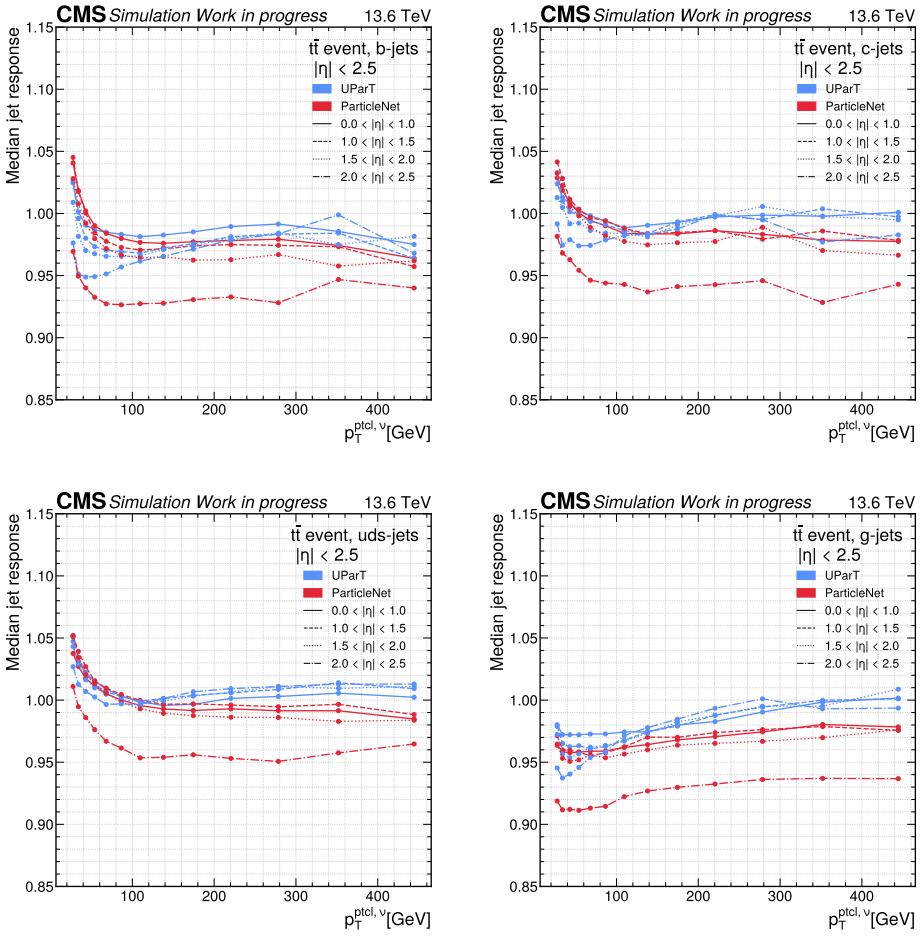

Figure 5.14: Median of the regressed response with neutrino contribution for b (upper left), c (upper right), uds (lower left) and g (lower right) jets. The performance of ParticleNet (red) and UParT (blue) are evaluated for different values of $|\eta|$ of the jet.

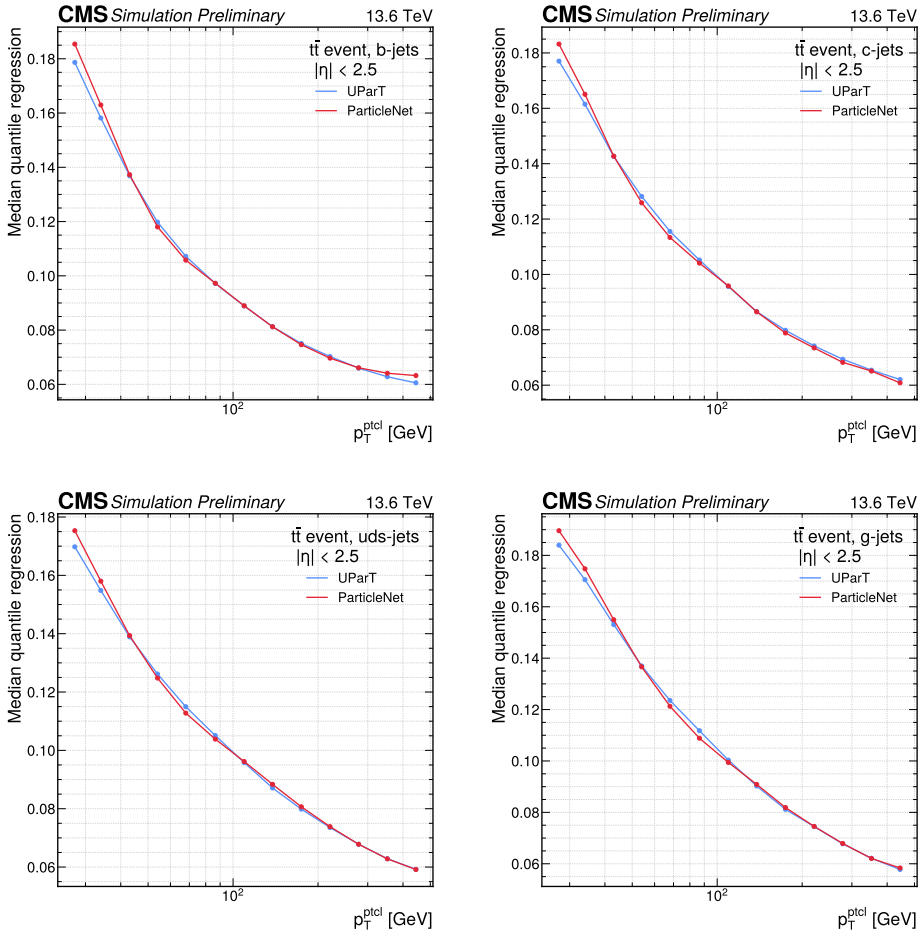

Figure 5.15: Median of the regressed quantile resolution for b (upper left), c (upper right), uds (lower left) and g (lower right) jets. The performance of ParticleNet (red) and UParT (blue) are evaluated for different values of $|\eta|$ of the jet [186].

5.3.2 Robustness of the Unified Particle Transformer

To evaluate the robustness of our model against attacks, we trained a nominal model in parallel under the same conditions as those described in Section 5.2, without R-NGM training. We evaluate the jets against NGM, R-NGM attacks, and the nominal performance. The magnitude of the attacks, ϵ, is set to 0.002 for R-NGM attacks and 0.01 for NGM attacks. We evaluate the usual white-box attacks and use the transfer attack method for R-NGM with our nominal model to obtain a black-box attack, validating the absence of gradient masking. In this section, we report the results obtained for b-tagging and c-tagging.

Figures 5.16 and 5.17 illustrate the performance against the various white-box attacks and the nominal performance. We observe that the results are consistent with the detailed discussion in Chapter 4, with our R-NGM model demonstrating improved robustness against attacks. Also, the R-NGM model achieves the same nominal performance as the nominal one when evaluated on nominal samples. The robustness against black-box attacks is illustrated in Figure 5.18 and shows the robustness gain against this type of attack, ensuring that the robustness is not due to a gradient masking phenomenon.

Thus, the R-NGM training of UParT meets expectations regarding increased robustness against adversarial attacks, confirming that the model is less sensitive to changes in the input features.

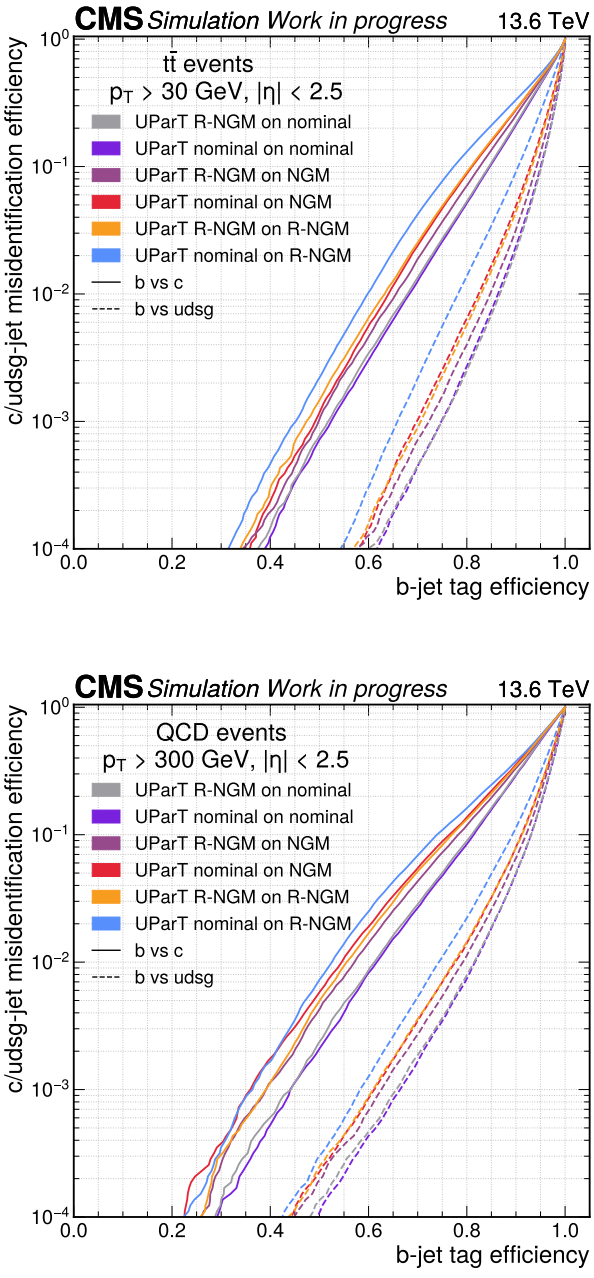

Figure 5.16: ROC curves of the b-tagging white-box adversarial performance of UParT, evaluated on $t\bar{t}$ events (above) and QCD multijet (below). The dashed line represents the udsg jet rejection while the solid line represents the c jet rejection.

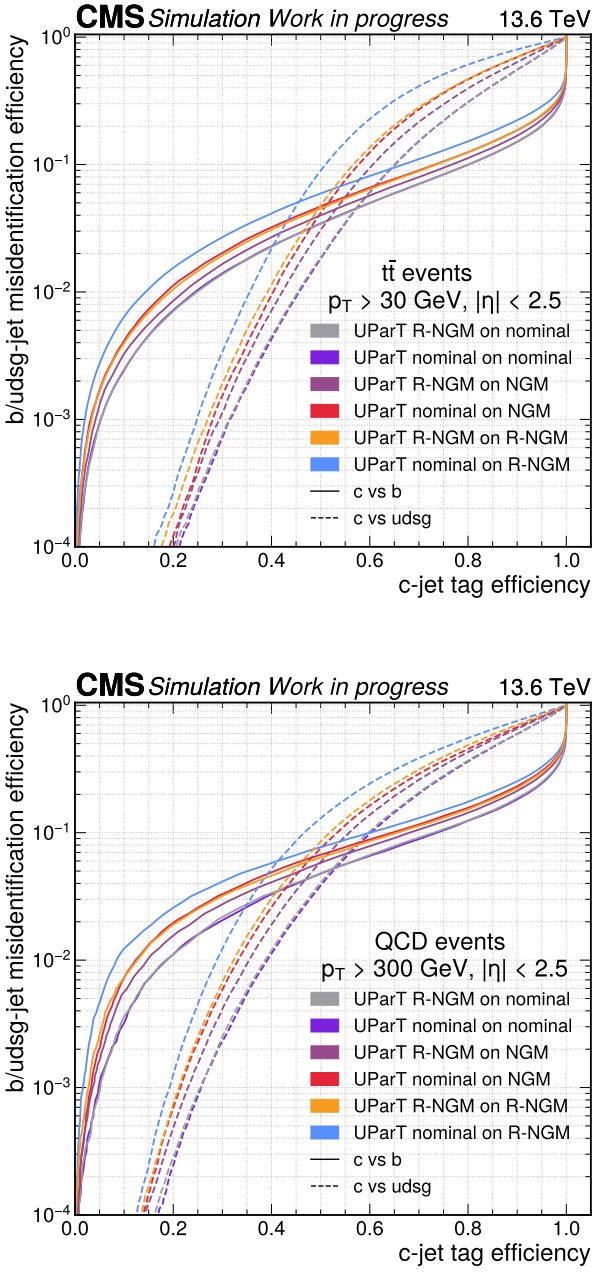

Figure 5.17: ROC curves of the c-tagging white-box adversarial performance of UParT, evaluated on $t\bar{t}$ events (above) and QCD multijet (below). The dashed line represents the udsg jet rejection while the solid line represents the b jet rejection.

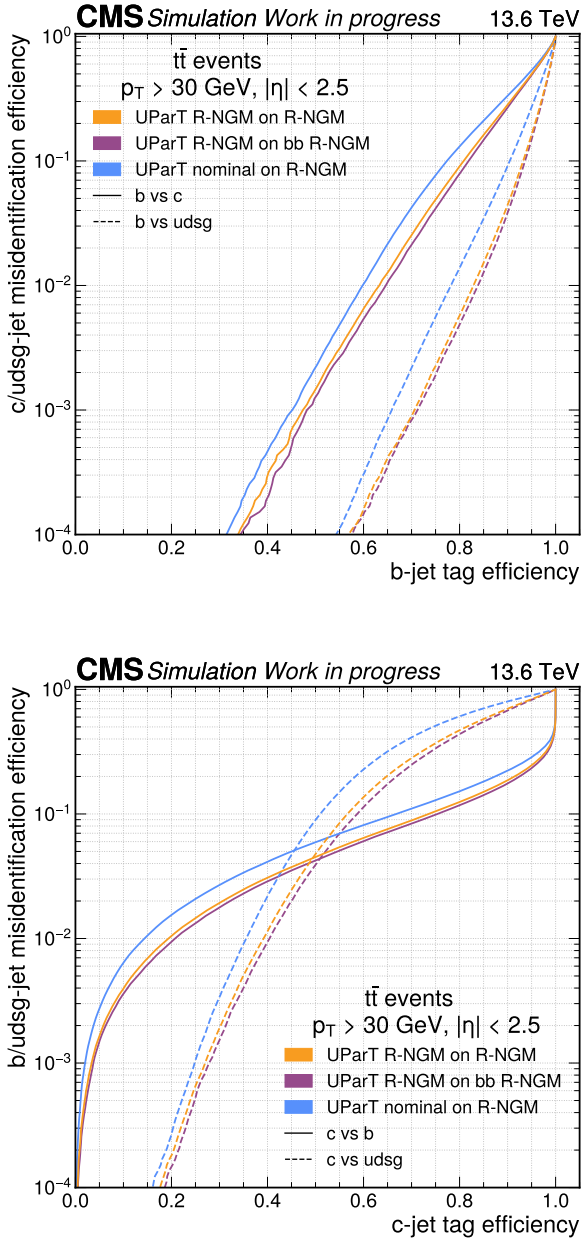

Figure 5.18: ROC curves of the b-tagging and c-tagging black-box R-NGM adversarial performance of UParT, evaluated on $t\bar{t}$ events. The dashed line represents the udsg jet rejection, while the solid line represents the c jet rejection. The white-box R-NGM are shown to compare with the black-box attacks.

5.4 Conclusion and outlook

5.4.1 Results

In this chapter, we have shown how we established a new Transformer-based algorithm that extends the application of resolved jet algorithms from heavy-flavour identification only to include the identification of τ_h and s jets, as well as energy correction regression and the estimation of its resolution.

We have established how to extend the definition of our classes to enable us to obtain the new targets necessary to perform our tasks. Additionally, we benefited from the update of the PUPPI algorithm for better efficiency in the reconstruction of τ_h jets at low p_{T} [244]. We also extended the input variables to improve the performance of heavy-flavour tagging, τ_h tagging. We demonstrated that the current set of variables has sufficient discriminative power to separate s jets from u and d jets. In training, we benefited from the efforts made in the previous chapters, particularly regarding the training framework and the establishment of adversarial training to train our UParT model, ensuring the best robustness given the available computational capacity and time.

The results are conclusive: UParT establishes the best performance in heavy-flavour tagging, becoming the state-of-the-art for this task and deep learning-based regression, opening the door to the potential future use of flavour-aware jet energy correction. Regarding τ_h tagging and jet energy regression, we achieved similar or slightly lower performance than the previous state-of-the-art algorithm, ParticleNet. Subsequent studies explained the source of this underperformance, caused by the absence of a crucial variable and a software error that led to key variables for τ_h being incorrectly constructed, thus reducing identification capability. Finally, we also evaluated that our UParT model created a tagger capable of identifying s jets in the CMS experiment, which has never been achieved before in the CMS collaboration. Our s-tagger can distinguish s jets from ud jets with an efficiency of approximately 22% for 10% misidentification, ensuring sufficient separation for future calibration. Finally, we

demonstrated that the expected robustness of the model against adversarial attacks was achieved, ensuring lower sensitivity of the model to changes in input features.

Figure 5.19 shows the evolution of flavour tagging algorithms in the CMS experiment. This review of b-taggers from Run 1 to Run 3 highlights the efforts made and the performance gains achieved by the different algorithms. Between the initial Run 2 and the current Run 3 taggers, the light jet rejection for b-tagging was improved by a factor of 59, while the c jet rejection by a factor of 6.9. This improvement in rejection highlights the significant improvements in flavour tagging algorithms thanks to the construction of deep neural networks designated for this task. A specific discriminant, named UParT ($k_c = 0.14$) defined as $prob_{k_c} = \frac{prob_b}{k_c prob_c + (1-k_c) prob_{udsg}}$ is used as a post-training reweighting of the probabilities by ATLAS [245] and has been tuned for UParT for comparison purposes. Compared with ATLAS taggers, illustrated in 5.20, UParT also displays state-of-the-art performance compared to the latest ATLAS tagger GN2.

Attention can also be given to the first jet energy regression performance from an inclusive training [147]. The initial results for the ParticleNet model regression are promising in terms of performance achieved. However, these results have also highlighted the limitations of the proposed training strategies. First, using jets with a minimum raw p_{T} of 15 GeV for training is a limiting factor, which required setting a minimum p_{T} value of 50 GeV in the first evaluation to avoid bin-by-bin migrations due to resolution effects. Reducing the minimum p_{T} value for training could help mitigate this effect. Also, at low p_{T}, the closure for our jets is not perfect and is off-trend by about 2-8% in the barrel and 10-20% in the endcap. Therefore, an MC truth correction is still necessary at this stage.

UParT is now integrated into the CMS data-taking for 2024. We should soon have the first results of the algorithm's post-calibration performance. These measurements should allow us to conclude the question of the robustness gain achieved through our adversarial training.

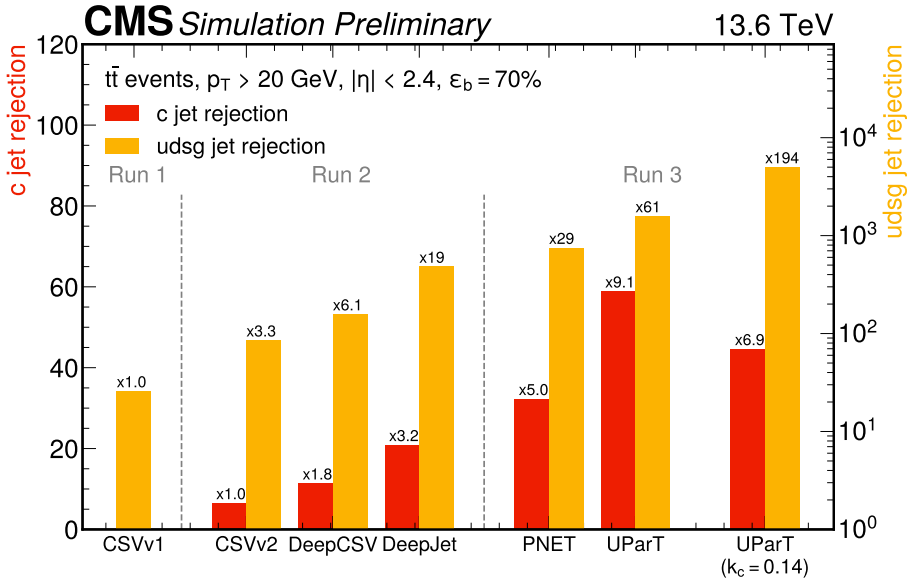

Figure 5.19: Evolution of the udsg, in yellow and c jet, in red, rejection for a fixed b jet identification efficiency of 70% for CMS taggers from Run 1 to Run 3 [186]. The UParT ($k_c = 0.14$) represent a post-training reweighting similar to ATLAS discriminator development [245]

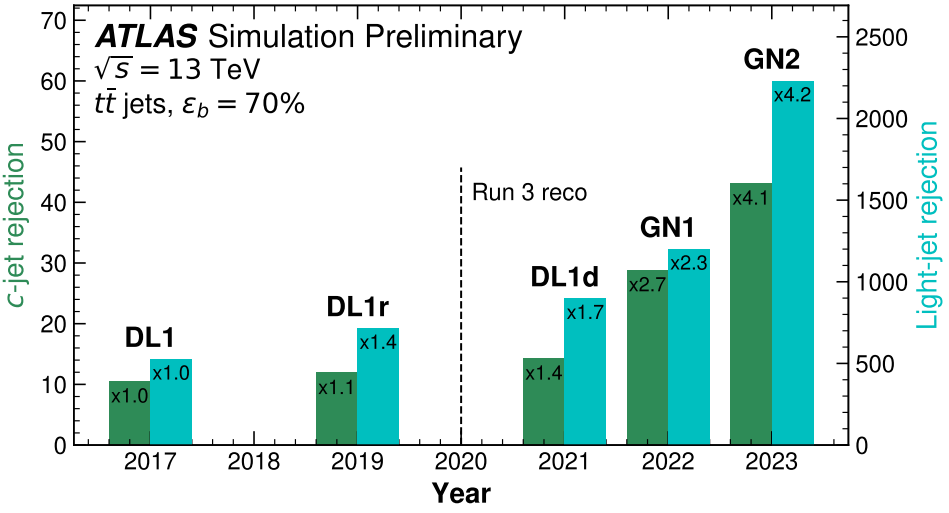

Figure 5.20: Evolution of the udsg, in blue and c jet, in green, rejection for a fixed b jet identification efficiency of 70% for ATLAS taggers from Run 2 to Run 3 [246].

5.4.2 Development perspectives and ideas

The prospects for improving UParT are, in some cases, already known and addressed. Indeed, the issues related to τ_h tagging and regression performance could be resolved by introducing the missing variables and retraining with the corrections to the LostTracks collection. These elements have already been added to our DeepNtuples sample producer [166] in later corrections to be ready for a future retraining campaign. We could also expand the range of variables by extending the hit-related variables in the tracker to distinguish by layer, to improve the understanding of these. Additionally, we could extend the pairwise features, for example, by introducing, for charged particles, the closest approach distance and its resolution, as well as the distance of this approach from the primary vertex, to provide lower-level information on secondary vertices similar to the variables used by vertexing algorithms [61].

The calibration results of UParT will also conclude the robustness's impact on the data/MC agreement and its impact on calibration. The first results of the Robust Particle Transformer, which used NGM training with a previous sub-optimal setup, compared to the nominal training of the inclusive ParticleNet have shown no significant improvement of the SF derived [190], indicating the gain of robustness reached could be insufficient concerning the magnitude of changes brought by the simulation mismodeling. However, the results for the Run 3 2022 $e\mu$ + jets region indicate a promising gain in the data/MC agreement. Figure 5.21 illustrates the b vs all discriminant of the inclusive ParticleNet, UParT with nominal training and UParT with the R-NGM training. These domains are pre-calibrated to compare the taggers' distribution and their robustness. We observe some differences, first in the low-probability region, where the UParT model with R-NGM training shows an improved data/MC agreement compared to the nominal training and to ParticleNet inclusive, albeit to a lesser extent. Similarly, Figure 5.22 shows the same distributions but with the discriminant having transformed under $\tanh^{-1}(\text{prob})$ to expand the high-probability distribution and observe the behaviour at closure. We also notice the benefit of the R-NGM method, which outperforms both inclusive ParticleNet and

nominal UParT in this context. Thus, the preliminary results of UParT demonstrate improved robustness compared to the performance that would have been achieved with nominal training and the previous state-of-the-art available.

If the adversarial attack method fails to improve the data/MC agreement, several avenues can be considered. First, we could continue to focus on techniques aimed at robustness and generalization capacity by investing more resources, for example, in more advanced attacks such as CW attacks [208], or by exploring second-order attacks [221, 222]. Another approach would be to explore methods like Sharpness-Aware Minimization [193] or to develop a data-driven method based on control regions containing jets with a high purity in a specific flavour. Methods of this kind have already been explored [247], and creating such a domain adaptation method could allow the model to immediately learn the crucial differences between our simulated jets and our data, enabling better capture and mitigation of these differences.

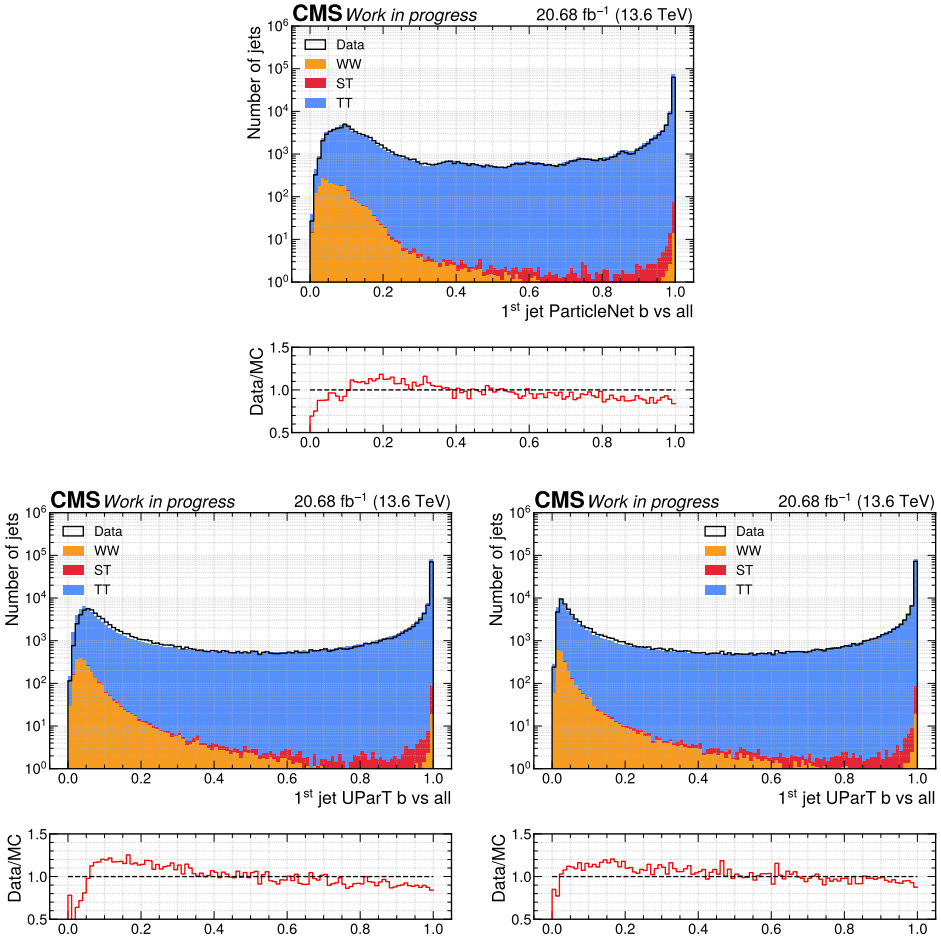

Figure 5.21: Distribution of the b vs all discriminant of ParticleNet (upper), the UParT nominal (lower left) and adversarial (lower right) for the e μ + jets domain region in the Run 3 2022EE era F+G.

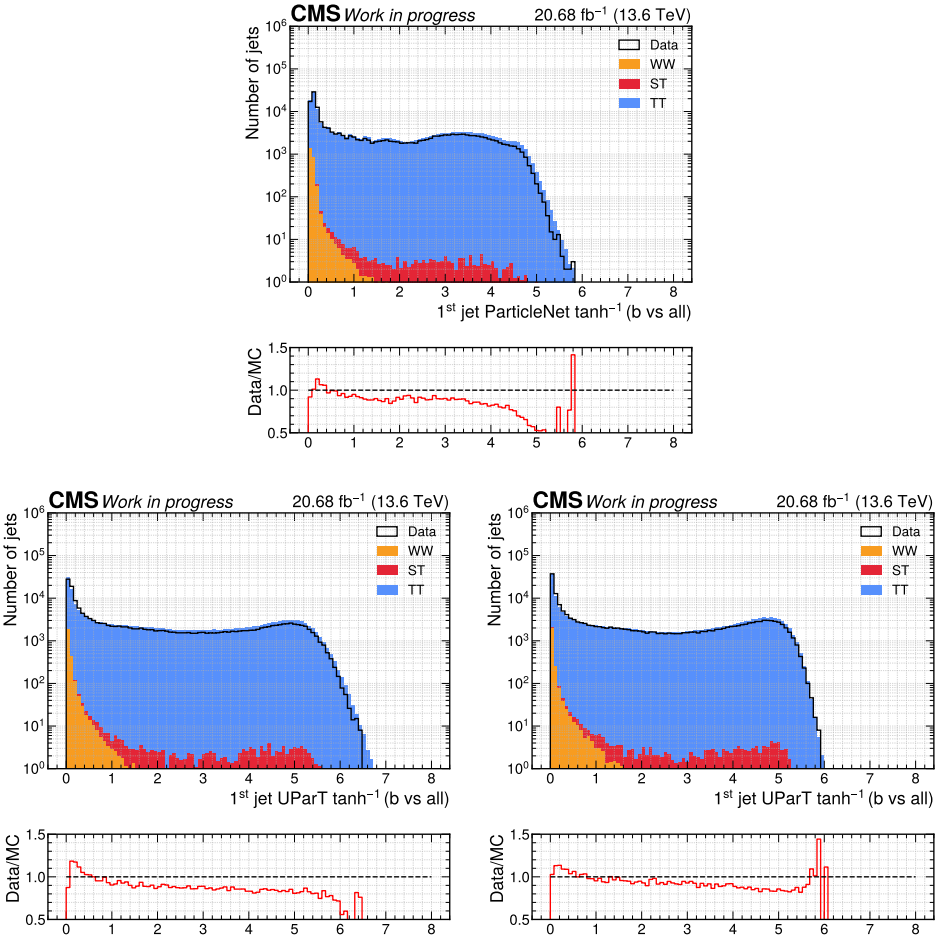

Figure 5.22: Distribution of $\tanh^{-1}$ of the b vs all discriminant of ParticleNet (upper), the UParT nominal (lower left) and adversarial (lower right) for the e μ + jets domain region in the Run 3 2022EE era F+G.

Chapter 6

Conclusion and future work

6.1 General conclusions

This thesis explored Deep Learning for jet algorithms at the CMS experiment, focusing on applying artificial neural network architectures towards a unified and robust jet algorithm approach. We began by discussing artificial neural network architectures in Chapter 3 in the context of heavy-flavour identification. We started and introduced the key elements of jet structure that we aim to preserve, namely the Particle Cloud representation and the previous models up to the state-of-the-art at the beginning of this research, ParticleNet [129]. We developed a new class of architecture: the Transformer-based models DeepJet Transformer [138] and Particle Transformer [139]. We demonstrated that these structures offer an alternative solution to the limitations posed by the ParticleNet model, and we established similarities with the properties we want to preserve up to demonstrating the permutation invariance of the new models. These models outperformed the previous state-of-the-art in both classification performance and computational efficiency, with the Particle Transformer model showing superior results due to its pairwise bias, better capturing constituent interrelations. Further studies demonstrated the scalability of Transformer models, maintaining robustness and performance and the benefit of permutation invariance for the model's safety.

In Chapter 4, we addressed the issue of the performance of our algorithms when applied to the data collected by the CMS experiment. We briefly explained the mismodelling affecting our Monte Carlo simulations, necessitating calibration to correct these effects in physics analyses. As mismodelling results in a performance loss, we approached the problem from the perspective of Deep Learning. We aimed to modify our model training to improve its robustness to variable changes without introducing information from the data distribution. We introduced the concept of adversarial attacks, designed to fool neural networks with subtle distortions to the input variables. We developed adversarial attacks tailored for jet flavour tagging, the NGM and R-NGM attacks, and a training method balancing robustness and performance. Models trained with these techniques, particularly R-NGM, improved robustness without sacrificing performance on non-distorted inputs. Transfer attacks further validated this, confirming the link between reduced gradient norms and enhanced robustness.

In Chapter 5, we continued the effort towards a unified jet tagging algorithm. We included elements previously developed for the extended version of ParticleNet, such as hadronic tau tagging and jet energy regression. As part of this doctoral research, we also expanded the classification capabilities of the tagger by introducing strange jet tagging, a first for the CMS experiment. We employed these elements alongside the state-of-the-art architecture, Particle Transformer, and the adversarial training method, R-NGM, discussed and developed in the previous chapters. We combined all these efforts into a model that can be used by the CMS Collaboration for the 2024 data-taking period. After the training, we evaluated the obtained performance and concluded that our new model, UParT, sets a new state-of-the-art in terms of performance for heavy-flavour tagging, surpassing previous models in b-jet identification efficiency by 3% to 20% or more, depending on the considered working point and rejection. We also improved the jet regression performance while performing similarly to the previous state-of-the-art in hadronic taus tagging and, for the first time, provided an s-tagger with low efficiency for separating jets originating from s quarks from those originating from

light quarks, u and d. Robustness studies confirmed that adversarial training behaved as expected and improved the model's resilience to changes in the input features.

Thus, we can conclude that the results obtained in this doctoral work demonstrate the utility of various aspects of Deep Learning in the context of jet algorithms. Using neural network structures that are well-suited to the physical objects under study improves the tasks required from the algorithms, such as classification. We have also shown that the choice of architecture can reduce computational complexity without compromising performance, as demonstrated by the development of Transformer models. The training of models is another crucial aspect, where we studied the impact of adversarial training methods to reduce the model's sensitivity. Furthermore, we demonstrated that it is possible to extend the task of heavy-flavour tagging to unified classification across different quark flavours, as well as hadronic taus and jet energy correction regression and resolution. Jets play an essential role in physics at hadron colliders like the LHC, and developing the methods presented here is, therefore, crucial for improving the results of future physics analyses.

6.2 Future work

The continuation of this doctoral work can be divided between the improvement and further development of the algorithms, and their practical application. For the first element, the ongoing development can follow the *Better, Faster, Stronger* vision. The *Better* component would focus on performance improvement, where new variables could enhance the current state-of-the-art, particularly with new low-level variables that are underutilised or, currently, only partially exploited. Work on the architecture could also lead to performance improvements. Recent advancements such as state-space models [148–150] or models based on the Kolmogorov-Arnold representation theorem [248] to construct learnable activation functions are two of the many paths that have emerged over the past year in the active field of Deep Learning research. However, the most significant impact of

these structures will be on the *Faster* aspect, as the quadratic complexity of Transformer models makes their use for highly complex objects more costly. Improving computational complexity towards linear models is a crucial challenge that may not seem as important in the case of small-radius jets, such as AK4 jets. However, for applications on jets with larger radii, such as AK8 jets, or in the context of an event-based model that considers all Particle-Flow candidates, moving towards linear complexity will be crucial to reducing the algorithms' computational footprint in the future. Finally, the *Stronger* component, which represents the robustness of our model, will likely depend heavily on the conclusions drawn from the upcoming calibration campaign. This will confirm whether the achieved robustness reduces the impact of mismodelling and will guide us in refining adversarial training methods. Additionally, training methods involving unlabelled data could improve the tagger's response to mismodelling. Methods such as domain adaptation [247] or unsupervised pre-training on data could help better control our algorithms' dependence on the distributions of variables from simulations. An area of development beyond this vision could be the extension of the tasks currently performed by our algorithm to the identification of the charge of the parton/hadron initiating the jet [249].

A particular attention is also given to improving training methods to reduce both the training time and the carbon footprint of models within the CMS experiment. In this context, a new experimental development phase of b-hive [167] is underway, and initial trials have achieved a training speed of $\sim$135 iterations per second for ParT, representing an $\sim$145% increase in training speed compared to the previous state-of-the-art [167]. Compared to the training capabilities two years ago [170,171], this translates to a factor 8.4 increase in speed, reducing training durations from one week to less than one day. Current evaluations were conducted with a single GPU, while the experimental developments were primarily targeted for multi-GPU training. As a result, further acceleration is expected, making it possible to train on datasets and models that are an order of magnitude larger than the current sizes in an acceptable time scale and resource consumption.

Another area of future development will be the calibration of the s-tagger. As we have seen in the results, we successfully developed a tagger capable of identifying s-jets with low efficiency. However, the necessary calibration for it has not yet been developed. The main challenge in this calibration will be to devise a method for constructing a control region enriched with s-jets to correct the mismodelling effects. An initial intuition would be to exploit phase spaces already known from calibration processes. From this perspective, the use of the decay channel $W \to cs$ seems promising, as it can be exploited like current methods targeting semileptonic $t\bar{t}$ events or W + c regions [61, 188]. However, this method, based on the W boson decay, is sensitive to the CKM matrix [237, 238] and the values we have set for generating our MC simulations. Thus, while this method could be employed for specific channels such as $H \to s\bar{s}$, it would introduce a bias in the context of measurements involving the CKM matrix itself, such as the measurement of V_{ts} [239], which we want to avoid. Therefore, using another channel, such as $Z \to s\bar{s}$, could be a potential approach for this analysis. This method would likely depend on a tag-and-probe technique [61, 188], which must be adapted. The selection of the process, triggers, and selection criteria will require thorough analysis.

Appendix A

Permutation equivariance of Transformer models

In this appendix, we develop and demonstrate that any Transformer block avoiding positional encoding and causal masking is permutation-equivariant. By adding a permutation-invariant pooling layer, we can create a permutation-invariant model capable of fulfilling the requirements of the Particle Cloud representation. Specifically, we demonstrate the permutation invariance of the predictions of the DeepJet Transformer and Particle Transformer models.

To demonstrate invariance, we will rely on an essential property of permutation matrices: they are orthogonal ($PP^T = PP^{-1} = \mathbb{1}$). Beyond this property, we will ensure the permutation invariance of our algorithms through the permutation invariance and equivariance properties. We begin by demonstrating that the composition of two permutation-equivariant functions retains equivariance and the composition of a permutation invariant and permutation equivariant functions is permutation equivariant:

Lemma A.0.1 *Given two functions, $f : R^{N,d} \to R^{N,m}$ and $g : R^{N,m} \to R^{N,k}$, if f and g are permutation-equivariants, then $f \circ g$ is permutation-equivariant.*

Proof A.0.1

$$\begin{aligned}\forall X \in R^{N,d}, \forall P \in P_N : g \circ f(PX) &= g(f(PX)) \\ &= g(Pf(X)) \\ &= Pg(f(X)) \\ &= Pg \circ f(X)\end{aligned}$$

Lemma A.0.2 *Given two functions, $f : R^{N,d} \to R^{N,m}$ and $g : R^{N,m} \to R^k$, if f is permutation-invariant and g is permutation-equivariant, then $f \circ g$ is permutation-invariant.*

Proof A.0.2

$$\begin{aligned}\forall X \in R^{N,d}, \forall P \in P_N : g \circ f(PX) &= g(f(PX)) \\ &= g(Pf(X)) \\ &= g(f(X)) \\ &= g \circ f(X)\end{aligned}$$

By utilising these properties and ensuring that every function used in the chain of functions defining our Transformer blocks maintains permutation equivariance, we ensure that our algorithms produce a representation of the jet constituents that respects the properties of the Particle Cloud. Applying a permutation-invariant pooling function followed by linear layers for classification guarantees that the predictions of our algorithms remain invariant under permutation of the jet constituents. Thus, we demonstrate the permutation invariance of the DeepJet Transformer and Particle Transformer models.

Permutation equivariance of the activation functions, linear and normalization layers

Activation Functions

All activation functions used in our models are permutation-equivariant. Most are element-wise activation functions acting independently on each variable. For those the equivariance is

trivial and does not require any demonstration. The only non-trivial function employed is the softmax function, which is also permutation-equivariant. It can be decomposed into an element-wise function acting independently on each variable, e^{x_i}, and normalization by a denominator $\sum_k e^{x_k}$. The denominator, being the sum of elements, is invariant under permutation due to the commutativity of the sum and is shared among all elements along the chosen axis k. Thus, the softmax function retains equivariance by ensuring that the ordering of inputs does not affect the normalization step.

Linear Layers

The equivariance of any MLP or linear layer is trivial. This result follows that linear transformations preserve the relationships between input elements, and matrix multiplication is associative:

$$\begin{gathered}\mathrm{Lin}(PX, W) = (PX)W = P(XW) = P\mathrm{Lin}(X, W) \\ \forall X \in \mathbb{R}^{N,d}, \forall P \in P_N, \forall W \in \mathbb{R}^{d,d'}\end{gathered} \tag{A.1}$$

where W is the weight matrix and X the input tensor.

Normalization Layers

Similarly, both batch normalization and layer normalization are permutation-equivariant. This property is trivial and follows from the construction of the normalization functions, for which we recall the following initial equation:

$$y = \frac{x - E(x)}{\sqrt{Var(x) + \epsilon}} \cdot \gamma + \beta \tag{A.2}$$

where the mean $E(x)$ and variance $Var(x)$ are permutation-equivariant quantities by construction, regardless of the axis on which they are measured. ϵ is a scalar variable applied element-wise. Similarly to the linear layer, the affine transformation affects only the variables axis and, therefore, does not impact the permutation, thus

preserving the permutation equivariance of the normalization methods used here.

Permutation equivariance of the embedding and feedforward layer

The feedforward layer, composed of linear transformations and activation functions, retains permutation equivariance due to the previously discussed properties of these components.

Similarly, the embedding layers used by our algorithms are composed of MLP layers, defined as a series of normalization, linear and activation layers, applied to each partition of our input variables before all of the partitions are concatenated to create a single tensor representing all of the constituents of the jet. However, one can note that at this stage, equivariance applies to permutations between elements within each partition. After concatenating the entire set into a single tensor, we can construct a unique object representing all the constituent elements of our jet and thus apply global permutations. By employing only MLP layers at this stage, we ensure that our model is permutation-equivariant of any partition permutation, allowing us to create a global representation of the set of each jet.

Permutation Equivariance of Scaled-Dot-Product Attention

To ensure the permutation equivariance of the scaled-dot-product self-attention [111], we can begin by unfolding the elements defined by the equation:

$$\begin{aligned} \mathrm{MHA}(X) &= \mathrm{Concat}(h_1, ..., h_n)W^O, \\ h_i &= \mathrm{Attention}(XW^{Q,i}, XW^{K,i}, XW^{V,i}) \end{aligned} \tag{A.3}$$

The linear layers $XW^{Q,i}$, $XW^{K,i}$, $XW^{V,i}$, and W^O are permutation-equivariant as previously demonstrated. Since the concatenation occurs along the variables axis, it is trivially equivariant.

It remains to demonstrate that the attention mechanism itself is equivariant.

To demonstrate equivariance, let us inject the permutation matrix $P \in P_N$ into the scaled-dot-product formula:

$$\begin{aligned}
\text{Attention}(PQ, PK, PV) &= \text{SoftMax}\left(\frac{PQK^TP^T}{\sqrt{d_k}}\right)PV \\
&= P\ \text{SoftMax}\left(\frac{QK^T}{\sqrt{d_k}}\right)P^TPV \\
&= P\ \text{SoftMax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \\
&= P\ \text{Attention}(Q, K, V)
\end{aligned} \tag{A.4}$$

We have demonstrated this by using the softmax function's permutation equivariance and the orthogonality of the permutation matrix P. Since permutation matrices are orthogonal, they maintain the inner product structure between queries and keys, preserving equivariance.

For the case of Particle Transformer, we also include the adjoint interaction tensor A, which, by its construction from edge features, changes under permutation as PAP^T [143]. Therefore, when adding the interaction tensor A of dimension (h, N, N) for each jet to the scaled-dot-product attention each interaction 'mask' U_h of dimension (N, N), simplified as U in equation A.5, contributes to a single head as follows:

$$\begin{aligned}
\text{Attention}(PQ, PK, PV, PUP^T) &= \text{SoftMax}\left(\frac{PQK^TP^T}{\sqrt{d_k}} + PUP^T\right)PV \\
&= \text{SoftMax}\left(P(\frac{QK^T}{\sqrt{d_k}} + U)P^T\right)PV \\
&= P\ \text{SoftMax}\left(\frac{QK^T}{\sqrt{d_k}} + U\right)P^TPV \\
&= P\ \text{SoftMax}\left(\frac{QK^T}{\sqrt{d_k}} + U\right)V \\
&= P\ \text{Attention}(Q, K, V, U)
\end{aligned} \tag{A.5}$$

Thus, we have demonstrated that all operations of our embedding layers and Transformer blocks are permutation-equivariant. This ensures that the hidden variables produced by our Transformer model create a representation of our jet that respects the Particle Cloud principle as desired.

On Positional encoding and Causality

Positional encodings introduce a notion of order, while causal masks enforce a directional structure on sequences. These operations inherently depend on the input order and thus break permutation equivariance. Hence, positional encoding and causal masks introduce a dependency on the order of inputs, breaking permutation equivariance. By avoiding these components, we maintain the desired permutation properties in our models.

Permutation of the Pooling Mechanisms

Pooling mechanisms such as the class token attention used by Particle Transformer is a permutation-invariant operation. We can ensure this by developing the attention mentioned, using the class token vector as the query and the jet constituents as the query and value

$$
\begin{aligned}
\text{Attention}(Q_{CLS}, PK, PV) &= \text{SoftMax}\left(\frac{Q_{CLS}K^T P^T}{\sqrt{d_k}}\right) PV \\
&= \text{SoftMax}\left(\frac{Q_{CLS}K^T}{\sqrt{d_k}}\right) P^T PV \\
&= \text{SoftMax}\left(\frac{Q_{CLS}K^T}{\sqrt{d_k}}\right) V \\
&= \text{Attention}\left(Q_{CLS}, K, V\right)
\end{aligned}
\tag{A.6}
$$

where Q_{CLS} is the query vector obtained by a linear layer, and K and V are the key and value tensors of the usual jet constituents. We have used the permutation equivariance properties of the softmax function and the orthogonality of P.

For DeepJet Transformer, we similarly develop the attention pooling function and use the same properties of softmax equivariance and the orthogonality of P:

$$
\begin{aligned}
\text{Attention Pooling}(PZ, PX) &= \text{SoftMax}\left(Z^T P^T\right) PX \\
&= \text{SoftMax}\left(Z^T\right) P^T PX \\
&= \text{SoftMax}\left(Z^T\right) X \\
&= \text{Attention Pooling}(Z, X)
\end{aligned}
\tag{A.7}
$$

Thus, we have demonstrated that our pooling operations are permutation-invariant, allowing us to construct, for both of our algorithms, a vector encoding the information from the jet constituents tensor while preserving the properties of the Particle Cloud representation. By adding linear layers for predictions on the vector obtained through pooling, we ensure the total permutation invariance of our models. We have thus demonstrated the permutation invariance of DeepJet Transformer and Particle Transformer, and we can conclude this development.

Appendix B

Training variables

npf feature	**Training 2023**	**Training 2024**
Fraction of the jet momentum carried by the npf	✓	✓
ΔR between the jet axis and the npf	✓	✓
Boolean value identifying the npf as a photon or not	✓	✓
Fraction of energy deposit of the npf in the hadronic calorimeter	✓	✓
ΔR between the npf and the closest secondary vertex	✓	✓
The PUPPI weight of the npf	✓	✓
npf η relative to the jet axis	×	✓
npf ϕ relative to the jet axis	×	✓

Table B.1: List of neutral PF candidates (npf) features

cpf feature	**Training 2023**	**Training 2024**
cpf η relative to the jet axis	✓	✓
cpf p_T relative to the jet axis	✓	✓
Dot product of the jet and cpf momentum	✓	✓
Dot product of the jet and cpf momentum divided by the jet momentum norm	✓	✓
ΔR between the jet axis and the cpf	✓	✓
The cpf 2D impact parameter value	✓	✓
The cpf 2D impact parameter significance	✓	✓
The cpf 3D impact parameter value	✓	✓
The cpf 3D impact parameter significance	✓	✓
The cpf distance to the jet axis	✓	✓
Fraction of the jet momentum carried by the cpf	✓	✓
ΔR between the cpf and the closest secondary vertex	✓	✓
Integer indicating if the cpf was used for the primary vertex fit	✓	✓
The PUPPI weight of the cpf	✓	✓
χ^2 of the charged track fit	✓	✓
Integer indicating the fitting quality of the cpf	✓	✓
The electric charge of the cpf	×	✓
The longitudinal displacement of the cpf	×	✓
The decay length of the cpf	×	✓
The fraction of the energy deposit in the hadronic calorimeter of the cpf	×	✓
The fraction of the energy deposit in the electronic calorimeter of the cpf	×	✓
The particle-flow pdg id of the cpf	×	✓
The number of lost inner hits of the cpf	×	✓
The number of pixel hits of the cpf	×	✓
The number of strip hits of the cpf	×	✓

Table B.2: List of charged PF candidates (cpf) features

lt feature	Training 2023	Training 2024
lt η relative to the jet axis	×	✓
lt p_T relative to the jet axis	×	✓
Dot product of the jet and lt momentum	×	✓
Dot product of the jet and lt momentum divided by the jet momentum norm	×	✓
ΔR between the jet axis and the lt	×	✓
The lt 2D impact parameter value	×	✓
The lt 2D impact parameter significance	×	✓
The lt 3D impact parameter value	×	✓
The lt 3D impact parameter significance	×	✓
The lt distance to the jet axis	×	✓
ΔR between the lt and the closest secondary vertex	×	✓
The PUPPI weight of the lt	×	✓
χ^2 of the charged track fit	×	✓
Integer indicating the fitting quality of the lt	×	✓
The electric charge of the lt	×	✓
The number of lost inner hits of the lt	×	✓
The number of pixel hits of the lt	×	✓
The number of strip hits of the lt	×	✓

Table B.3: List of Lost Tracks (lt) features

sv feature	Training 2023	Training 2024
The sv transverse momentum	✓	✓
ΔR between the jet axis and the sv	✓	✓
The sv mass	✓	✓
Number of tracks associated with the sv	✓	✓
χ^2 value of the sv fit	✓	✓
Reduced χ^2 value of the sv fit	✓	✓
The sv 2D impact parameter value	✓	✓
The sv 2D impact parameter significance	✓	✓
The sv 3D impact parameter value	✓	✓
The sv 3D impact parameter significance	✓	✓
Cosine of the angle between the sv vertex flight direction and sv momentum	✓	✓
Ratio between the sv energy and the jet energy	✓	✓
sv η relative to the jet axis	×	✓
sv ϕ relative to the jet axis	×	✓

Table B.4: List of secondary vertices (sv) features

Appendix C

Training samples

Training 2023

Requested events	TT samples	Samples name
20.000.000	/TTTo2J1L1Nu_CP5_13p6TeV_powheg-pythia8/Run3Winter22MiniAOD-122X_mcRun3_2021_realistic_v9-v2/MINIAODSIM	ttbar semileptonic
Requested events	QCD samples	Samples name
5.000.000	/QCD_Pt_50to80_TuneCP5_13p6TeV_pythia8/Run3Winter22MiniAOD-122X_mcRun3_2021_realistic_v9-v2/MINIAODSIM	QCD p_T 50 to 80 GeV
5.000.000	/QCD_Pt_80to120_TuneCP5_13p6TeV_pythia8/Run3Winter22MiniAOD-122X_mcRun3_2021_realistic_v9-v2/MINIAODSIM	QCD p_T 80 to 120 GeV
5.000.000	/QCD_Pt_120to170_TuneCP5_13p6TeV_pythia8/Run3Winter22MiniAOD-122X_mcRun3_2021_realistic_v9-v2/MINIAODSIM	QCD p_T 120 to 170 GeV
5.000.000	/QCD_Pt_170to300_TuneCP5_13p6TeV_pythia8/Run3Winter22MiniAOD-122X_mcRun3_2021_realistic_v9-v2/MINIAODSIM	QCD p_T 170 to 300 GeV
5.000.000	/QCD_Pt_300to470_TuneCP5_13p6TeV_pythia8/Run3Winter22MiniAOD-122X_mcRun3_2021_realistic_v9-v2/MINIAODSIM	QCD p_T 300 to 470 GeV
5.000.000	/QCD_Pt_470to600_TuneCP5_13p6TeV_pythia8/Run3Winter22MiniAOD-122X_mcRun3_2021_realistic_v9-v2/MINIAODSIM	QCD p_T 470 to 600 GeV
5.000.000	/QCD_Pt_600to800_TuneCP5_13p6TeV_pythia8/Run3Winter22MiniAOD-122X_mcRun3_2021_realistic_v9-v2/MINIAODSIM	QCD p_T 600 to 800 GeV
5.000.000	/QCD_Pt_800to1000_TuneCP5_13p6TeV_pythia8/Run3Winter22MiniAOD-122X_mcRun3_2021_realistic_v9-v2/MINIAODSIM	QCD p_T 800 to 1000 GeV
5.000.000	/QCD_Pt_1000to1400_TuneCP5_13p6TeV_pythia8/Run3Winter22MiniAOD-122X_mcRun3_2021_realistic_v9-v2/MINIAODSIM	QCD p_T 1000 to 1400 GeV
5.000.000	/QCD_Pt_1400to1800_TuneCP5_13p6TeV_pythia8/Run3Winter22MiniAOD-122X_mcRun3_2021_realistic_v9-v2/MINIAODSIM	QCD p_T 1400 to 1800 GeV
5.000.000	/QCD_Pt_1800to2400_TuneCP5_13p6TeV_pythia8/Run3Winter22MiniAOD-122X_mcRun3_2021_realistic_v9-v2/MINIAODSIM	QCD p_T 1800 to 2400 GeV
5.000.000	/QCD_Pt_2400to3200_TuneCP5_13p6TeV_pythia8/Run3Winter22MiniAOD-122X_mcRun3_2021_realistic_v9-v2/MINIAODSIM	QCD p_T 2400 to 3200 GeV
5.000.000	/QCD_Pt_3200toInf_TuneCP5_13p6TeV_pythia8/Run3Winter22MiniAOD-122X_mcRun3_2021_realistic_v9-v2/MINIAODSIM	QCD p_T 3200 to Inf GeV

2024 samples

Requested events	TT samples	Samples name
20.000.000	/TTto4Q-TuneCP5-13p6TeV-powheg-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v3/MINIAODSIM	ntuple-TTto4Q
20.000.000	/TTto2L2Nu-TuneCP5-13p6TeV-powheg-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v3/MINIAODSIM	ntuple-TTto2L2Nu
20.000.000	/TTtoLNu2Q-TuneCP5-13p6TeV-powheg-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v3/MINIAODSIM	ntuple-TTtoLNu2Q
Requested events	HtoTauTau samples	Samples name
5.000.000	/GluGluHto2Tau-M-125-2HDM-II-TuneCP5-13p6TeV-powheg-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v2/MINIAODSIM	ntuple-GluGluHto2Tau-M-125-2HDM
5.000.000	/GluGluHto2Tau-M-180-2HDM-II-TuneCP5-13p6TeV-powheg-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v2/MINIAODSIM	ntuple-GluGluHto2Tau-M-180-2HDM
5.000.000	/GluGluHto2Tau-M-250-2HDM-II-TuneCP5-13p6TeV-powheg-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v2/MINIAODSIM	ntuple-GluGluHto2Tau-M-250-2HDM
5.000.000	/GluGluHto2Tau-M-400-2HDM-II-TuneCP5-13p6TeV-powheg-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v2/MINIAODSIM	ntuple-GluGluHto2Tau-M-400-2HDM
5.000.000	/GluGluHto2Tau-M-600-2HDM-II-TuneCP5-13p6TeV-powheg-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v2/MINIAODSIM	ntuple-GluGluHto2Tau-M-600-2HDM
5.000.000	/GluGluHto2Tau-M-800-2HDM-II-TuneCP5-13p6TeV-powheg-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v2/MINIAODSIM	ntuple-GluGluHto2Tau-M-800-2HDM
5.000.000	/GluGluHto2Tau-M-1000-2HDM-II-TuneCP5-13p6TeV-powheg-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v2/MINIAODSIM	ntuple-GluGluHto2Tau-M-1000-2HDM
5.000.000	/GluGluHto2Tau-M-1200-2HDM-II-TuneCP5-13p6TeV-powheg-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v2/MINIAODSIM	ntuple-GluGluHto2Tau-M-1200-2HDM
5.000.000	/GluGluHto2Tau-M-1400-2HDM-II-TuneCP5-13p6TeV-powheg-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v2/MINIAODSIM	ntuple-GluGluHto2Tau-M-1400-2HDM
5.000.000	/GluGluHto2Tau-M-1600-2HDM-II-TuneCP5-13p6TeV-powheg-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v2/MINIAODSIM	ntuple-GluGluHto2Tau-M-1600-2HDM
5.000.000	/GluGluHto2Tau-M-1800-2HDM-II-TuneCP5-13p6TeV-powheg-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v2/MINIAODSIM	ntuple-GluGluHto2Tau-M-1800-2HDM
5.000.000	/GluGluHto2Tau-M-2000-2HDM-II-TuneCP5-13p6TeV-powheg-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v2/MINIAODSIM	ntuple-GluGluHto2Tau-M-2000-2HDM
5.000.000	/GluGluHto2Tau-M-2300-2HDM-II-TuneCP5-13p6TeV-powheg-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v2/MINIAODSIM	ntuple-GluGluHto2Tau-M-2300-2HDM
5.000.000	/GluGluHto2Tau-M-2600-2HDM-II-TuneCP5-13p6TeV-powheg-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v2/MINIAODSIM	ntuple-GluGluHto2Tau-M-2600-2HDM
5.000.000	/GluGluHto2Tau-M-2900-2HDM-II-TuneCP5-13p6TeV-powheg-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v2/MINIAODSIM	ntuple-GluGluHto2Tau-M-2900-2HDM

Requested events	DY samples	Samples name
5.000.000	/DYto2L-2Jets-MLL-50-0J-TuneCP5-13p6TeV-amcatnloFXFX-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v3/MINIAODSIM	ntuple-DYJetsToLL-0J
5.000.000	/DYto2L-2Jets-MLL-50-1J-TuneCP5-13p6TeV-amcatnloFXFX-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v3/MINIAODSIM	ntuple-DYJetsToLL-1J
5.000.000	/DYto2L-2Jets-MLL-50-2J-TuneCP5-13p6TeV-amcatnloFXFX-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v3/MINIAODSIM	ntuple-DYJetsToLL-2J
5.000.000	/DYto2L-2Jets-MLL-50-PTLL-100to200-2J-TuneCP5-13p6TeV-amcatnloFXFX-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v1/MINIAODSIM	ntuple-DYJetsToLL-100to200
5.000.000	/DYto2L-2Jets-MLL-50-PTLL-200to400-2J-TuneCP5-13p6TeV-amcatnloFXFX-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v1/MINIAODSIM	ntuple-DYJetsToLL-200to400
5.000.000	/DYto2L-2Jets-MLL-50-PTLL-400to600-2J-TuneCP5-13p6TeV-amcatnloFXFX-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v1/MINIAODSIM	ntuple-DYJetsToLL-400to600
5.000.000	/DYto2L-2Jets-MLL-50-PTLL-600-2J-TuneCP5-13p6TeV-amcatnloFXFX-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v1/MINIAODSIM	ntuple-DYJetsToLL-600
5.000.000	/DYto2TautoMuTauh-M-50-TuneCP5-13p6TeV-madgraphMLM-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v1/MINIAODSIM	ntuple-DYto2TautoMuTauh
Requested events	W+jets samples	Samples name
5.000.000	/WtoLNu-4Jets-TuneCP5-13p6TeV-madgraphMLM-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v3/MINIAODSIM	ntuple-WJetsToLNu-0J
5.000.000	/WtoLNu-4Jets-1J-TuneCP5-13p6TeV-madgraphMLM-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v3/MINIAODSIM	ntuple-WJetsToLNu-1J
5.000.000	/WtoLNu-4Jets-2J-TuneCP5-13p6TeV-madgraphMLM-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v3/MINIAODSIM	ntuple-WJetsToLNu-2J
5.000.000	/WtoLNu-4Jets-3J-TuneCP5-13p6TeV-madgraphMLM-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v3/MINIAODSIM	ntuple-WJetsToLNu-3J
5.000.000	/WtoLNu-4Jets-4J-TuneCP5-13p6TeV-madgraphMLM-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v3/MINIAODSIM	ntuple-WJetsToLNu-4J
5.000.000	/Wto2Q-3Jets-HT-200to400-TuneCP5-13p6TeV-madgraphMLM-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v2/MINIAODSIM	ntuple-Wto2Q-3Jets-HT-200to400
5.000.000	/Wto2Q-3Jets-HT-400to600-TuneCP5-13p6TeV-madgraphMLM-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v2/MINIAODSIM	ntuple-Wto2Q-3Jets-HT-400to600
5.000.000	/Wto2Q-3Jets-HT-600to800-TuneCP5-13p6TeV-madgraphMLM-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v2/MINIAODSIM	ntuple-Wto2Q-3Jets-HT-600to800
5.000.000	/Wto2Q-3Jets-HT-800-TuneCP5-13p6TeV-madgraphMLM-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v2/MINIAODSIM	ntuple-Wto2Q-3Jets-HT-800
Requested events	QCD samples	Samples name
5.000.000	/QCD-PT-15to30-TuneCP5-13p6TeV-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v2/MINIAODSIM	ntuple-QCD-15to30
5.000.000	/QCD-PT-30to50-TuneCP5-13p6TeV-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v2/MINIAODSIM	ntuple-QCD-30to50
5.000.000	/QCD-PT-50to80-TuneCP5-13p6TeV-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v2/MINIAODSIM	ntuple-QCD-50to80
5.000.000	/QCD-PT-80to120-TuneCP5-13p6TeV-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v2/MINIAODSIM	ntuple-QCD-80to120
5.000.000	/QCD-PT-120to170-TuneCP5-13p6TeV-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v2/MINIAODSIM	ntuple-QCD-120to170
5.000.000	/QCD-PT-170to300-TuneCP5-13p6TeV-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v2/MINIAODSIM	ntuple-QCD-170to300
5.000.000	/QCD-PT-300to470-TuneCP5-13p6TeV-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v2/MINIAODSIM	ntuple-QCD-300to470
5.000.000	/QCD-PT-470to600-TuneCP5-13p6TeV-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v2/MINIAODSIM	ntuple-QCD-470to600
5.000.000	/QCD-PT-600to800-TuneCP5-13p6TeV-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v2/MINIAODSIM	ntuple-QCD-600to800
5.000.000	/QCD-PT-800to1000-TuneCP5-13p6TeV-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v2/MINIAODSIM	ntuple-QCD-800to1000
5.000.000	/QCD-PT-1000to1400-TuneCP5-13p6TeV-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v2/MINIAODSIM	ntuple-QCD-1000to1400
5.000.000	/QCD-PT-1400to1800-TuneCP5-13p6TeV-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v2/MINIAODSIM	ntuple-QCD-1400to1800
5.000.000	/QCD-PT-1800to2400-TuneCP5-13p6TeV-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v2/MINIAODSIM	ntuple-QCD-1800to2400
5.000.000	/QCD-PT-2400to3200-TuneCP5-13p6TeV-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v2/MINIAODSIM	ntuple-QCD-2400to3200
5.000.000	/QCD-PT-3200-TuneCP5-13p6TeV-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v2/MINIAODSIM	ntuple-QCD-3200

Requested events	H samples	Samples name
5.000.000	/GluGluHToBB-M-125-TuneCP5-13p6TeV-powheg-pythia8/Run3Summer23BPix MiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v2/MINIAODSIM	ntuple-GluGluHToBB-M-125
5.000.000	/GluGluHToTauTau-M-125-TuneCP5-13p6TeV-powheg-pythia8/Run3Summer23BPix MiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v2/MINIAODSIM	ntuple-GluGluHToTauTau-M-125
5.000.000	/VBFHto2B-M-125-TuneCP5-13p6TeV-powheg-pythia8/Run3Summer23BPix MiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v3/MINIAODSIM	ntuple-VBFHToBB-M-125
5.000.000	/VBFHToTauTau-M125-TuneCP5-13p6TeV-powheg-pythia8/Run3Summer23BPix MiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v2/MINIAODSIM	ntuple-VBFHToTauTau-M125
5.000.000	/VBFHToCC-M-125-TuneCP5-13p6TeV-powheg-pythia8/Run3Summer23BPix MiniAODv4-130X-mcRun3-2023-realistic-postBPix-v6-v2/MINIAODSIM	ntuple-VBFHToCC-M-125
5.000.000	/WminusH-Hto2C-WtoLNu-M-125-TuneCP5-13p6TeV-powheg-pythia8/Run3Summer23 BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v2/MINIAODSIM	ntuple-WminusH-HTo2C-WToLNu-M-125
5.000.000	/WminusH-Hto2B-WtoLNu-M-125-TuneCP5-13p6TeV-powheg-pythia8/Run3Summer23 BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v2/MINIAODSIM	ntuple-WminusH-HTo2B-WToLNu-M-125
5.000.000	/WplusH-Hto2C-WtoLNu-M-125-TuneCP5-13p6TeV-powheg-pythia8/Run3Summer23BPix MiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v2/MINIAODSIM	ntuple-WplusH-HTo2C-WToLNu-M-125
5.000.000	/WplusH-Hto2B-Wto2Q-M-125-TuneCP5-13p6TeV-powheg-pythia8/Run3Summer23BPix MiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v2/MINIAODSIM	ntuple-WplusH-HTo2B-WTo2Q-M-125
5.000.000	/ggZH-Hto2C-Zto2L-M-125-TuneCP5-13p6TeV-powheg-pythia8/Run3Summer23BPix MiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v3/MINIAODSIM	ntuple-ggZH-HTo2C-ZTo2L-M-125
5.000.000	/ggZH-Hto2B-Zto2L-M-125-TuneCP5-13p6TeV-powheg-pythia8/Run3Summer23BPix MiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v3/MINIAODSIM	ntuple-ggZH-HTo2B-ZTo2L-M-125
5.000.000	/ggZH-Hto2C-Zto2Q-M-125-TuneCP5-13p6TeV-powheg-pythia8/Run3Summer23BPix MiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v3/MINIAODSIM	ntuple-ggZH-HTo2C-ZTo2Q-M-125
5.000.000	/ggZH-Hto2B-Zto2Q-M-125-TuneCP5-13p6TeV-powheg-pythia8/Run3Summer23BPix MiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v3/MINIAODSIM	ntuple-ggZH-HTo2B-ZTo2Q-M-125
5.000.000	/GluGlutoHHto2B2Tau-kl-0p00-kt-1p00-c2-0p00-TuneCP5-13p6TeV-powheg-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v6-v2/MINIAODSIM	ntuple-GluGlutoHHto2B2Tau-1
5.000.000	/GluGlutoHHto2B2Tau-kl-0p00-kt-1p00-c2-1p00-TuneCP5-13p6TeV-powheg-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v6-v2/MINIAODSIM	ntuple-GluGlutoHHto2B2Tau-2
5.000.000	/GluGlutoHHto2B2Tau-kl-1p00-kt-1p00-c2-0p00-TuneCP5-13p6TeV-powheg-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v6-v2/MINIAODSIM	ntuple-GluGlutoHHto2B2Tau-3
5.000.000	/GluGlutoHHto2B2Tau-kl-1p00-kt-1p00-c2-0p10-TuneCP5-13p6TeV-powheg-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v6-v2/MINIAODSIM	ntuple-GluGlutoHHto2B2Tau-4
5.000.000	/GluGlutoHHto2B2Tau-kl-1p00-kt-1p00-c2-0p35-TuneCP5-13p6TeV-powheg-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v6-v2/MINIAODSIM	ntuple-GluGlutoHHto2B2Tau-5
5.000.000	/GluGlutoHHto2B2Tau-kl-1p00-kt-1p00-c2-3p00-TuneCP5-13p6TeV-powheg-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v6-v2/MINIAODSIM	ntuple-GluGlutoHHto2B2Tau-6
5.000.000	/GluGlutoHHto2B2Tau-kl-1p00-kt-1p00-c2-m2p00-TuneCP5-13p6TeV-powheg-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v6-v2/MINIAODSIM	ntuple-GluGlutoHHto2B2Tau-7
5.000.000	/GluGlutoHHto2B2Tau-kl-2p45-kt-1p00-c2-0p00-TuneCP5-13p6TeV-powheg-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v6-v2/MINIAODSIM	ntuple-GluGlutoHHto2B2Tau-8
5.000.000	/GluGlutoHHto2B2Tau-kl-5p00-kt-1p00-c2-0p00-TuneCP5-13p6TeV-powheg-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v6-v2/MINIAODSIM	ntuple-GluGlutoHHto2B2Tau-9
5.000.000	/GluGlutoHHto2B2WtoLNu2Q-kl-1p00-kt-1p00-c2-0p00-TuneCP5-13p6TeV-powheg-pythia8/Run3Summer23BPixMiniAODv4-130X-mcRun3-2023-realistic-postBPix-v2-v2/MINIAODSIM	ntuple-GluGlutoHHto2B2WtoLNu2Q

Appendix D

Charm tagging robustness performance

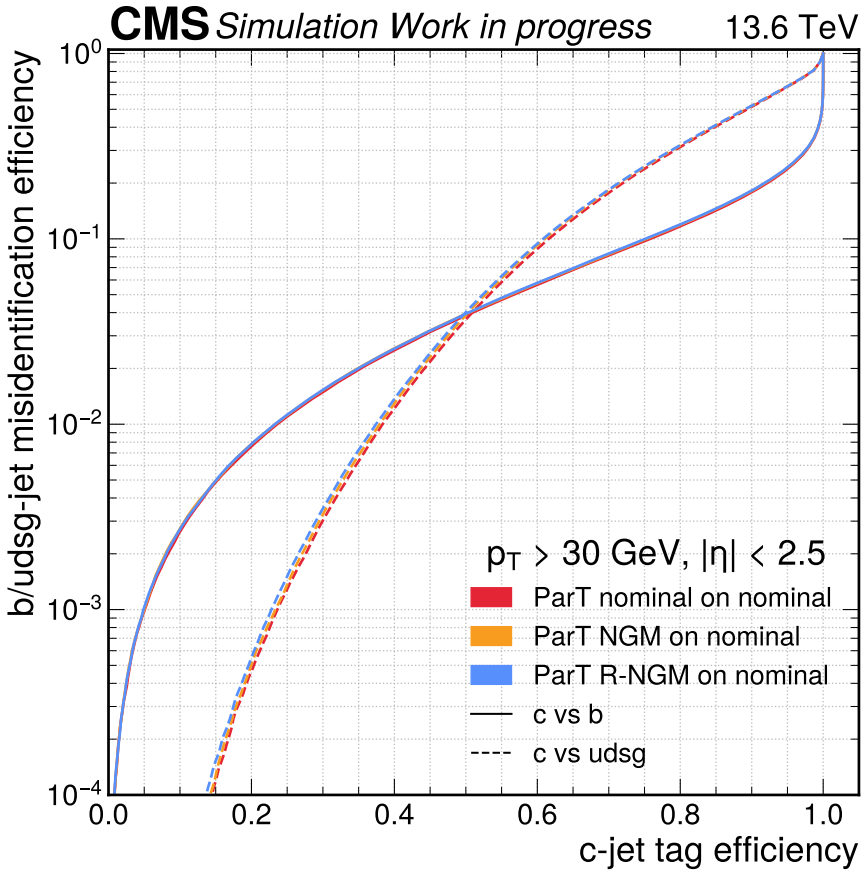

Figure D.1: ROC curves performance on nominal samples of the b vs udsg (dashed lines) and b vs c (solid lines) rejection of the Particle Transformer algorithm trained in nominal mode (red) or with NGM training (orange) and R-NGM (blue). We can observe that the three models perform similarly when evaluated on nominal samples.

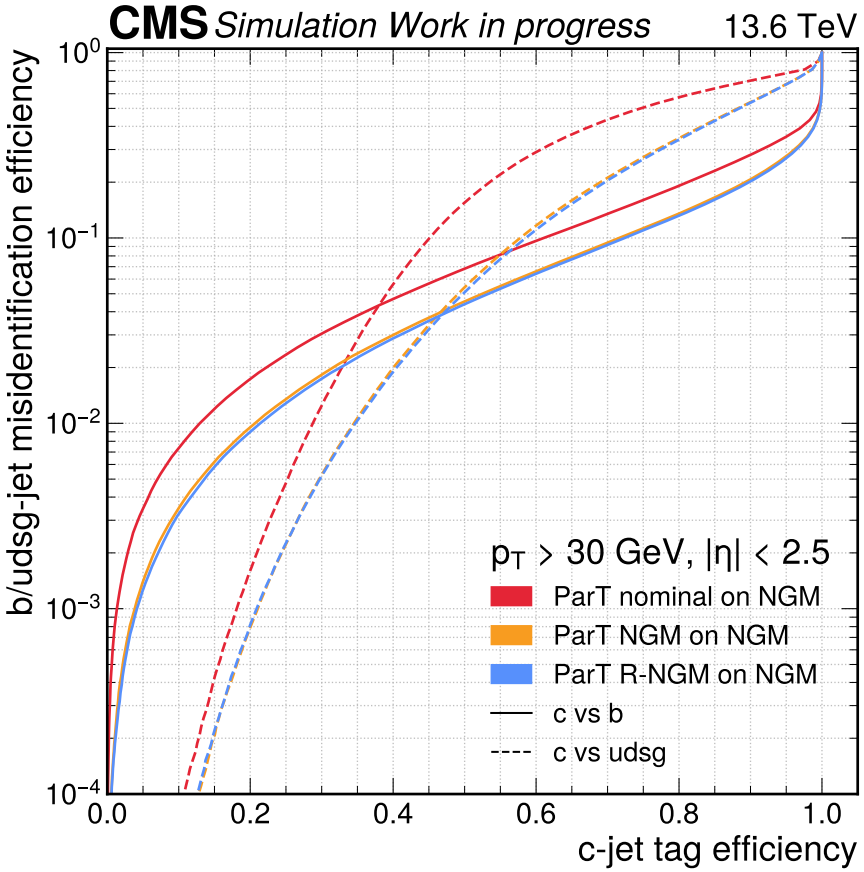

Figure D.2: ROC curves performance on NGM samples of the b vs udsg (dashed lines) and b vs c (solid lines) rejection of the Particle Transformer algorithm trained in nominal mode (red) or with NGM training (orange) and R-NGM (blue). We can observe that the two adversarially trained models perform better than the nominal one. The R-NGM training achieves similar robustness against NGM attacks than the NGM training, even slightly better for the b vs udsg rejection.

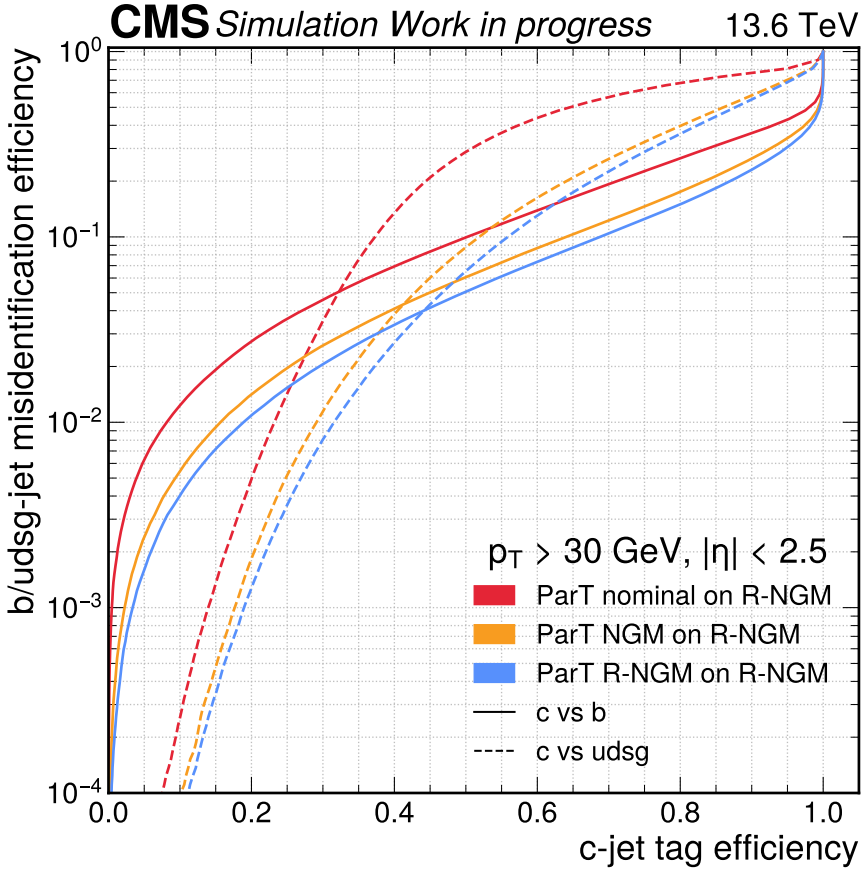

Figure D.3: ROC curves performance on R-NGM samples of the b vs udsg (dashed lines) and b vs c (solid lines) rejection of the Particle Transformer algorithm trained in nominal mode (red) or with NGM training (orange) and R-NGM (blue). We can observe that the two adversarially trained models perform better than the nominal one. The R-NGM training achieves better robustness than the NGM training.

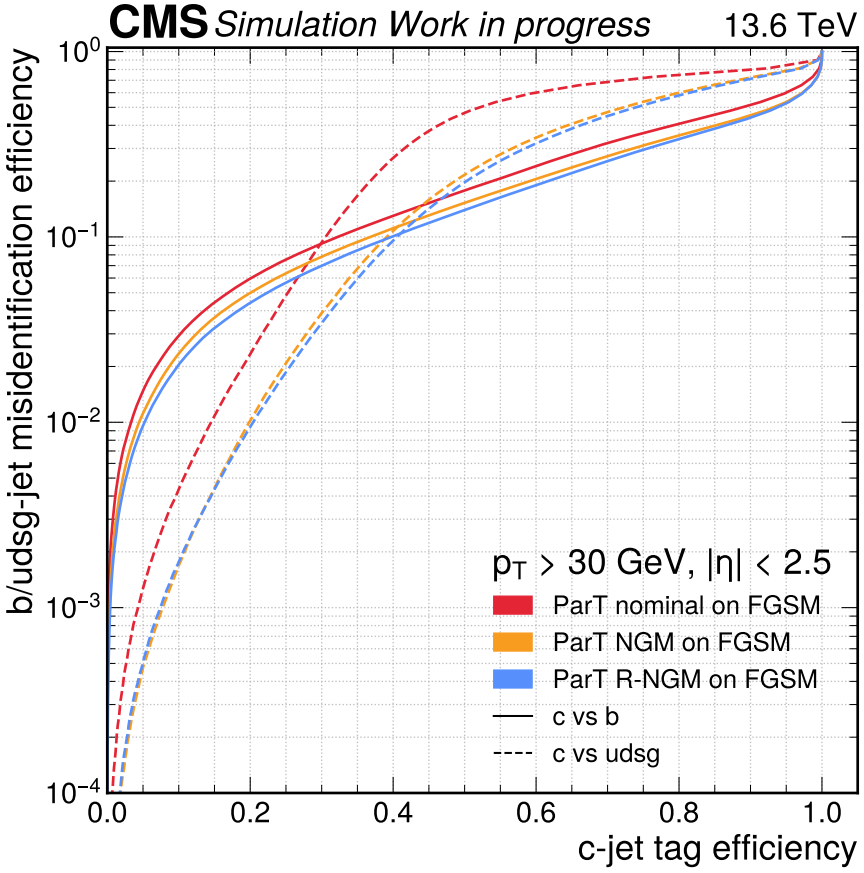

Figure D.4: ROC curves performance on FGSM samples of the b vs udsg (dashed lines) and b vs c (solid lines) rejection of the Particle Transformer algorithm trained in nominal mode (red) or with NGM training (orange) and R-NGM (blue). We can observe that the two adversarially trained models perform better than the nominal one. The R-NGM training achieves better robustness than the NGM training.

Author contributions

This appendix aims to highlight the author's contribution to the research discussed in this thesis and additional research that has not been mentioned.

The DeepJet Transformer algorithm introduced in Chapter 3 is initially developed for flavour tagging at the CMS experiment, of which I am the author. An adapted version of this algorithm was also employed in flavour tagging research for the FCC-ee, published in Ref. [138]. In this research, I contributed by creating and implementing DeepJet Transformer, assisting with model training and writing the publication. I also contributed to the introduction of the Particle Transformer algorithm to CMS in collaboration with its authors Huilin Qu, Congqiao Li, and Sitian Qian, along with Denise Müller.

Among the adversarial training methods in Chapter 4, the development of the NGM adversarial training method was done in collaboration with Annika Stein. At the same time, I am the sole author of the R-NGM method development at CMS.

Among the current CMS jet algorithms mentioned in Chapter 5, I co-developed the Run 3 version of DeepJet with Denise Müller. In collaboration with Annika Stein, I developed and trained RobustParticleTransformerAK4. Together with Stephane Cooperstein and Raffaele Gerosa, we developed and trained the UnifiedParticleTransformerAK4 model. For this last model, I contributed by adapting the ntupler, creating a version of the current b-hive framework adapted for inclusive training, and developing the extension for s-tagging. I was also responsible for the model training and its implementation

in CMSSW.

In parallel to this research work, I also had the opportunity to take on responsibilities within the b-tagging and vertexing (BTV) group at CMS. I served as the L3 Software & Algorithm convener from 2021 to 2023. I have also been an offline validator for flavour tagging algorithms since 2021. Since 2023, I have also served as the machine learning contact person for the BTV group and the jet and missing ET group (JME). I also contributed to organising the 2023 BTV workshop, 'To b or not to b,' at the Vrije Universiteit Brussel as a member of the local committee. Since 2021, I have been the principal contributor and maintainer of the ntupler used for training jet algorithms by BTV, DeepNTuples. I also contributed to the development of the DeepJet/DeepJetCore training framework, notably introducing the PyTorch training pipeline and replacing the KERAS version. I am also one of the main authors of the new deep learning framework, b-hive.

I have also undertaken and completed several deep learning projects not mentioned in my thesis. Primarily, I contributed to improving the performance of heavy-flavour tagging algorithms by introducing new geometric variables as attention bias and high-energy jet tagging through new variables related to tracker hits. I also studied the impact that different reweighting methods can have on performance, as well as potential training biases that some reweighting approximations might introduce. Additionally, I contributed to implementing new optimisers for training and introducing the knowledge transfer method known as 'knowledge distillation' to reduce the size of the final models.

Finally, I have also fulfilled teaching responsibilities by serving as an assistant for the 'Statistical Treatment of Experimental Data' course at the Bachelor's level since the 2021-2022 academic year. I also co-supervised Claire Dooms's bachelor's thesis, 'Feature Evaluation for Jet Tagging Using Machine Learning.'

Bibliography

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

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

[3] F. Englert and R. Brout. Broken Symmetry and the Mass of Gauge Vector Mesons. *Phys. Rev. Lett.*, 13:321–323, Aug 1964.

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

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

[6] Rumelhart, David E and Hinton, Geoffrey E and Williams, Ronald J. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[7] J J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982.

[8] Steven Weinberg. A Model of Leptons. *Phys. Rev. Lett.*, 19:1264–1266, Nov 1967.

[9] S. L. Glashow, J. Iliopoulos, and L. Maiani. Weak Interactions with Lepton-Hadron Symmetry. *Phys. Rev. D*, 2:1285–1292, Oct 1970.

[10] Scott Willenbrock. Symmetries of the standard model. In *Theoretical Advanced Study Institute in Elementary Particle Physics: Physics in $D \geqq 4$*, pages 3–38, 10 2004.

[11] Steven Weinberg. *The Quantum Theory of Fields, Volume 1: Foundations*. Cambridge University Press, 2005.

[12] Steven Weinberg. *The quantum theory of fields. Vol. 2: Modern applications*. Cambridge University Press, 1996. Cambridge, UK: Univ. Pr. (1996) 489 p.

[13] Matic Lubej. Standard Model, 2015. `https://www.physik.uzh.ch/groups/serra/StandardModel.html` [Last access : 24 Sept 2024)].

[14] S. Navas et al. Review of particle physics. *Phys. Rev. D*, 110(3):030001, 2024.

[15] Armen Tumasyan et al. Measurement of the top quark mass using a profile likelihood approach with the lepton + jets final states in proton–proton collisions at $\sqrt{s} =$ 13 TeV. *Eur. Phys. J. C*, 83(10):963, 2023.

[16] Georges Aad et al. Measurement of the top-quark mass using a leptonic invariant mass in pp collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector. *JHEP*, 06:019, 2023.

[17] Javier Aparisi et al. mb at mH: The Running Bottom Quark Mass and the Higgs Boson. *Phys. Rev. Lett.*, 128(12):122001, 2022.

[18] D. Hatton, C. T. H. Davies, J. Koponen, G. P. Lepage, and A. T. Lytle. Determination of $\overline{m}_b/\overline{m}_c$ and $\overline{m}_b$ from $n_f = 4$ lattice QCD+QED. *Phys. Rev. D*, 103(11):114508, 2021.

[19] Valery A Rubakov. *Classical theory of gauge fields*. Princeton Univ., Princeton, NJ, 2002.

[20] Yuri L. Dokshitzer. Calculation of the Structure Functions for Deep Inelastic Scattering and e+ e- Annihilation by Perturbation Theory in Quantum Chromodynamics. *Sov. Phys. JETP*, 46:641–653, 1977.

[21] V. N. Gribov and L. N. Lipatov. Deep inelastic e p scattering in perturbation theory. *Sov. J. Nucl. Phys.*, 15:438–450, 1972.

[22] Guido Altarelli and G. Parisi. Asymptotic Freedom in Parton Language. *Nucl. Phys. B*, 126:298–318, 1977.

[23] B. Andersson, G. Gustafson, G. Ingelman, and T. Sjöstrand. Parton fragmentation and string dynamics. *Physics Reports*, 97(2):31–145, 1983.

[24] Bo Andersson. *The Lund Model*. Cambridge Monographs on Particle Physics, Nuclear Physics and Cosmology. Cambridge University Press, 1998.

[25] Torbjorn Sjöstrand, Stephen Mrenna, and Peter Z. Skands. An Introduction to PYTHIA 8.2. *Comput. Phys. Commun.*, 191:159, 2015.

[26] Tina Potter. Particle and Nuclear physics lectures. `https://www.hep.phy.cam.ac.uk/~chpotter/particleandnuclearphysics/Lecture_07_QCD.pdf` [Last access : 24 Sept 2024)].

[27] E. Noether. Invariante Variationsprobleme. *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-Physikalische Klasse*, 1918:235–257, 1918.

[28] John Ellis. Higgs Physics. In *2013 European School of High-Energy Physics*, pages 117–168, 2015.

[29] Lyndon Evans and Philip Bryant. LHC Machine. *Journal of Instrumentation*, 3(08):S08001, aug 2008.

[30] LEP Phase 1 and Cost Breakdown. Phase 1 du LEP et analyse des coûts. 180th Meeting of Finance Committee. *CERN/SPC/0472*, 1981.

[31] Maurizio Vretenar, J Vollaire, R Scrivens, C Rossi, F Roncarolo, S Ramberger, U Raich, B Puccio, D Nisbet, R Mompo, S Mathot, C Martin, L A Lopez-Hernandez, A Lombardi, J Lettry, J B Lallement, I Kozsar, J Hansen, F Gerigk, A Funken, J F Fuchs, N Dos Santos, M Calviani, M Buzio, O Brunner, Y Body, P Baudrenghien, J Bauche, and T Zickler. *Linac4 design report*, volume 6 of *CERN Yellow Reports: Monographs*. CERN, Geneva, 2020.

[32] Ewa Lopienska. The CERN accelerator complex. Complexe des accélérateurs du CERN, 2022. General Photo.

[33] Public CMS Luminosity Information, 2024. `https://twiki.cern.ch/twiki/bin/view/CMSPublic/LumiPublicResults`.

[34] The CMS Collaboration. The CMS experiment at the CERN LHC. *Journal of Instrumentation*, 3(08):S08004, aug 2008.

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

[36] CMS coordinates system. `https://wiki.physik.uzh.ch/cms/_detail/latex:cms_coordinate_system.png?id=latex%3Aexample_spherical_coordinates`. Accessed: 2024-09-15.

[37] V Karimäki et al. The CMS tracker system project: Technical Design Report. *CMS-TDR-5*, 1997.

[38] The CMS collaboration. The CMS tracker: addendum to the Technical Design Report. *CMS-TDR-5-add-1*, 2000.

[39] W. Adam et al. The CMS Phase-1 Pixel Detector Upgrade. *JINST*, 16(02):P02027, 2021.

[40] CMS Tracker Detector Performance Results, 2024. `https://twiki.cern.ch/twiki/bin/view/CMSPublic/DPGResultsTRK`.

[41] Benedikt Roland Vormwald. The CMS inner tracker – transition from LHC Run I to Run II and first experience of Run II. *CMS-CR-2015-213*, 2015.

[42] J.-L. Agram. CMS Silicon Strip Tracker Performance. *Physics Procedia*, 37:844–850, 2012. Proceedings of the 2nd International Conference on Technology and Instrumentation in Particle Physics (TIPP 2011).

[43] The CMS Collaboration. Description and performance of track and primary-vertex reconstruction with the CMS tracker. *Journal of Instrumentation*, 9(10):P10009, oct 2014.

[44] The CMS collaboration. The CMS electromagnetic calorimeter project: Technical Design Report. *CMS-TDR-4*, 1997.

[45] Philippe Bloch, Robert Brown, Paul Lecoq, and Hans Rykaczewski. Changes to CMS ECAL electronics: addendum to the Technical Design Report. *CMS-TDR-4-add-1*, 2002.

[46] The CMS collaboration. CMS ECAL performance with 2017 data. *CMS-DP-2019-029*, 2019.

[47] Badder Marzocchi. Simulation of the CMS electromagnetic calorimeter response at the energy and intensity frontier. *Journal of Physics: Conference Series*, 1162:012007, 01 2019.

[48] The CMS collaboration. The CMS hadron calorimeter project: Technical Design Report. *CMS-TDR-2*, 1997.

[49] J Mans, J Anderson, B Dahmes, P de Barbaro, J Freeman, T Grassi, E Hazen, J Mans, R Ruchti, I Schimdt, T Shaw, C Tully, J Whitmore, and T Yetkin. CMS Technical Design Report for the Phase 1 Upgrade of the Hadron Calorimeter. *CMS-TDR-010*, 2012. Additional contact persons: Jeffrey Spalding, Fermilab, spalding@cern.ch, Didier Contardo, Universite Claude Bernard-Lyon I, contardo@cern.ch.

[50] Albert M Sirunyan et al. Calibration of the CMS hadron calorimeters using proton-proton collision data at $\sqrt{s} = 13$ TeV. *JINST*, 15:P05002, 2020.

[51] J. G. Layter. *The CMS muon project: Technical Design Report.* Technical design report. CMS. CERN, Geneva, 1997.

[52] The CMS collaboration. Performance of CMS muon reconstruction in pp collision events at $\sqrt{s} = 7$ TeV. *Journal of Instrumentation*, 7(10):P10002, oct 2012.

[53] M. A. Akl et al. CMS Technical Design Report for the Muon Endcap GEM Upgrade. 6 2015.

[54] Aram Hayrapetyan et al. Enriching the Physics Program of the CMS Experiment via Data Scouting and Data Parking. 3 2024.

[55] S. Dasu et al. CMS. The TriDAS project. Technical design report, vol. 1: The trigger systems. *CMS-TDR-6*, 12 2000.

[56] Sergio Cittolin, Attila Rácz, and Paris Sphicas. CMS The TriDAS Project: Technical Design Report, Volume 2: Data Acquisition and High-Level Trigger. CMS trigger and data-acquisition project. *CMS-TDR-6*, 2002.

[57] The CMS collaboration. Performance of Run 3 track reconstruction with the mkFit algorithm. *CMS-DP-2022-018*, 2022.

[58] Wolfgang Adam, Boris Mangano, Thomas Speer, and Teddy Todorov. Track Reconstruction in the CMS tracker. *CMS-NOTE-2006-041*, 2006.

[59] The CMS collaboration. Performance of the track selection DNN in Run 3. *CMS-DP-2023-009*, 2023.

[60] Thomas Speer, Kirill Prokofiev, R Frühwirth, Wolfgang Waltenberger, and Pascal Vanlaer. Vertex Fitting in the CMS Tracker. *CMS-NOTE-2006-032*, 2006.

[61] A.M. Sirunyan et al. Identification of heavy-flavour jets with the CMS detector in pp collisions at 13 TeV. *Journal of Instrumentation*, 13(05):P05011, may 2018.

[62] A.M. Sirunyan et al. Performance of the CMS muon detector and muon reconstruction with proton-proton collisions at $\sqrt{s} = 13$ TeV. *Journal of Instrumentation*, 13(06):P06015, jun 2018.

[63] W Adam, R Frühwirth, A Strandlie, and T Todorov. Reconstruction of electrons with the Gaussian-sum filter in the CMS tracker at the LHC, journal = Journal of Physics G: Nuclear and Particle Physics. 31(9):N9, jul 2005.

[64] The CMS collaboration. CMS software.

[65] Eric Metodiev. The Fractal Lives of Jets, 2020. `https://www.ericmetodiev.com/post/jetformation/` [Last access : 19 Sept 2024)].

[66] Matteo Cacciari, Gavin P. Salam, and Gregory Soyez. The anti-k_T jet clustering algorithm. *JHEP*, 04:063, 2008.

[67] Daniele Bertolini, Philip Harris, Matthew Low, and Nhan Tran. Pileup per particle identification. *Journal of High Energy Physics*, 2014(10), oct 2014.

[68] Yu.L Dokshitzer, G.D Leder, S Moretti, and B.R Webber. Better jet clustering algorithms. *Journal of High Energy Physics*, 1997(08):001–001, August 1997.

[69] M. Wobisch and T. Wengler. Hadronization Corrections to Jet Cross Sections in Deep-Inelastic Scattering, 1999.

[70] V. Khachatryan et al. Jet energy scale and resolution in the CMS experiment in pp collisions at 8 TeV. *Journal of Instrumentation*, 12(02):P02014, feb 2017.

[71] Garvita Agarwal. Jet Energy Scale and Resolution Measurements in CMS. In *Proceedings of 41st International Conference on High Energy physics — PoS(ICHEP2022)*, page 652. Sissa Medialab, December 2022.

[72] Mohiuddin Ahmed, Raihan Seraj, and Syed Mohammed Shamsul Islam. The k-means Algorithm: A Comprehensive Survey and Performance Evaluation. *Electronics*, 9(8), 2020.

[73] Yves Grandvalet and Yoshua Bengio. Semi-supervised Learning by Entropy Minimization. In L. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 17. MIT Press, 2004.

[74] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*. OpenReview.net, 2021.

[75] Mingyu Ding, Bin Xiao, Noel Codella, Ping Luo, Jingdong Wang, and Lu Yuan. Davit: Dual attention vision transformers. In Shai Avidan, Gabriel J. Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *ECCV (24)*, volume 13684 of *Lecture Notes in Computer Science*, pages 74–92. Springer, 2022.

[76] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings*

of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[77] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. LLaMA: Open and Efficient Foundation Language Models, 2023.

[78] Albert Q. Jiang et al. Mixtral of Experts, 2024.

[79] Yingyan Li, Lue Fan, Jiawei He, Yuqi Wang, Yuntao Chen, Zhaoxiang Zhang, and Tieniu Tan. Enhancing end-to-end autonomous driving with latent world model. *CoRR*, abs/2406.08481, 2024.

[80] Felix Wong et al. Discovery of a structural class of antibiotics with explainable deep learning. *Nature*, 626:1–9, 12 2023.

[81] G. Aad et al. Light-quark and gluon jet discrimination in *pp* p p collisions at $\sqrt{s} =$ 7 TeV s = 7 TeV with the ATLAS detector. *The European Physical Journal C*, 74(8), August 2014.

[82] Leif Lönnblad, Carsten Peterson, and Thorsteinn Rögnvaldsson. Finding gluon jets with a neural trigger. *Phys. Rev. Lett.*, 65:1321–1324, Sep 1990.

[83] Leif Lönnblad, Carsten Peterson, and Thorsteinn Rögnvaldsson. Using neural networks to identify jets. *Nuclear Physics B*, 349(3):675–702, 1991.

[84] Byron P. Roe, Hai-Jun Yang, Ji Zhu, Yong Liu, Ion Stancu, and Gordon McGregor. Boosted decision trees as an alternative to artificial neural networks for particle identification. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 543(2–3):577–584, May 2005.

[85] Joosep Pata, Javier Duarte, Jean-Roch Vlimant, Maurizio Pierini, and Maria Spiropulu. MLPF: efficient machine-learned particle-flow reconstruction using graph neural networks. *The European Physical Journal C*, 81(5), May 2021.

[86] Vardan Khachatryan et al. Search for the associated production of the Higgs boson with a top-quark pair. *JHEP*, 1409:087, 2014. Replaced with published version. Added journal reference and DOI.

[87] Supriya Jain. Evidence for Single Top Production at the Tevatron. In *43rd Rencontres de Moriond on QCD and High Energy Interactions*, pages 137–140, 5 2008.

[88] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[89] Andrew Ng. *Machine Learning Yearning*. Online Draft, 2017.

[90] A. B. Novikoff. On Convergence Proofs on Perceptrons. In *Proceedings of the Symposium on the Mathematical Theory of Automata*, volume 12, pages 615–622, New York, NY, USA, 1962. Polytechnic Institute of Brooklyn.

[91] Marvin Minsky and Seymour Papert. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge, MA, USA, 1969.

[92] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML 2010*, pages 807–814, 2010.

[93] Dan Hendrycks and Kevin Gimpel. Gaussian Error Linear Units (GELUs), 2023.

[94] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107:3–11, 2018. Special issue on deep reinforcement learning.

[95] S.I. Gallant. Perceptron-based learning algorithms. *IEEE Transactions on Neural Networks*, 1(2):179–191, 1990.

[96] Bernard Widrow and Marcian E. Hoff. Adaptive Switching Circuits. In *1960 IRE WESCON Convention Record, Part 4*, pages 96–104, New York, 1960. IRE.

[97] Ke-Lin Du, Chi-Sing Leung, Wai Ho Mow, and M. N. S. Swamy. Perceptron: Learning, Generalization, Model Selection, Fault Tolerance, and Role in the Deep Learning Era. *Mathematics*, 10(24), 2022.

[98] Y. Lecun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Handwritten Digit Recognition with a Back-Propagation Network. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, pages 396–404. Morgan Kaufmann, 1990.

[99] Chase Shimmin. Particle Convolution for High Energy Physics. 7 2021.

[100] Hamza Kheddar, Yassine Himeur, Abbes Amira, and Rachik Soualah. Image Classification in High-Energy Physics: A Comprehensive Survey of Applications to Jet Analysis. 3 2024.

[101] Y. LeCun, L. D. Jackel, L. Bottou, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, U. A. Muller, E. Sackinger, P. Simard, and V. Vapnik. 'Learning Algorithms For Classification: A Comparison On Handwritten Digit Recognition'. In J. H. Oh, C. Kwon, and S. Cho, editors, *Neural Networks: The Statistical Mechanics Perspective*, pages 261–276. World Scientific, 1995.

[102] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition, 2015.

[103] Hongwei Dong, Lamei Zhang, and Bin Zou. PolSAR Image Classification with Lightweight 3D Convolutional Networks. *Remote Sensing*, 12:396, 01 2020.

[104] Aditya Amberkar, Parikshit Awasarmol, Gaurav Deshmukh, and Piyush Dave. Speech Recognition using Recurrent Neural Networks. In *2018 International Conference on Current Trends towards Converging Technologies (ICCTCT)*, pages 1–4, 2018.

[105] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger, editors, *NIPS*, pages 3104–3112, 2014.

[106] STEPHEN G. BRUSH. History of the Lenz-Ising Model. *Rev. Mod. Phys.*, 39:883–893, Oct 1967.

[107] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

[108] Michael Phi. Illustrated Guide to LSTM's and GRU's: A step by step explanation, 2018. Accessed: (06/06/2024).

[109] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997.

[110] E. Bols, J. Kieseler, M. Verzetti, M. Stoye, and A. Stakia. Jet flavour classification using DeepJet. *Journal of Instrumentation*, 15(12):P12012–P12012, December 2020.

[111] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, page 5998–6008. Curran Associates, Inc., 2017.

[112] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2014. cite arxiv:1409.0473Comment: Accepted at ICLR 2015 as oral presentation.

[113] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective Approaches to Attention-based Neural Machine Translation, 2015.

[114] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis R. Bach and David M. Blei, editors, *ICML*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 448–456. JMLR.org, 2015.

[115] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer Normalization, 2016.

[116] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[117] Daesoo Lee and Seung Lee. Motion predictive control for DPS using predicted drifted ship position based on deep learning and replay buffer. *International Journal of Naval Architecture and Ocean Engineering*, 12:768–783, 01 2020.

[118] H. Robbins and S. Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.

[119] J. Kiefer and J. Wolfowitz. Stochastic Estimation of the Maximum of a Regression Function. *The Annals of Mathematical Statistics*, 23(3):462 – 466, 1952.

[120] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. 'Gradient-Based Learning Applied to Document Recognition'. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.

[121] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[122] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013.

[123] Yurii Nesterov. A method for solving the convex programming problem with convergence rate $O(1/k^2)$. *Proceedings of the USSR Academy of Sciences*, 269:543–547, 1983.

[124] Yanli Liu, Yuan Gao, and Wotao Yin. An improved analysis of stochastic gradient descent with momentum. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *NeurIPS*, 2020.

[125] Ehsan Adeli Fei-Fei Li. CS231n: Convolutional Neural Networks for Visual Recognition, February 2024.

[126] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[127] Yann N. Dauphin, Harm de Vries, Junyoung Chung, and Yoshua Bengio. RMSProp and equilibrated adaptive learning rates for non-convex optimization. *CoRR*, abs/1502.04390, 2015.

[128] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, San Diega, CA, USA, 2015.

[129] Huilin Qu and Loukas Gouskos. Jet tagging via particle clouds. *Physical Review D*, 101(5), March 2020.

[130] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. In *ICLR*. OpenReview.net, 2020.

[131] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR (Poster)*. OpenReview.net, 2019.

[132] Michael R. Zhang, James Lucas, Jimmy Ba, and Geoffrey E. Hinton. Lookahead optimizer: k steps forward, 1 step back. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché Buc, Emily B. Fox, and Roman Garnett, editors, *NeurIPS*, pages 9593–9604, 2019.

[133] Less Wright and Nestor Demeure. Ranger21: a synergistic deep learning optimizer. *CoRR*, abs/2106.13731, 2021.

[134] The CMS collaboration. Identification of b-quark jets with the CMS experiment. *Journal of Instrumentation*, 8(04):P04013, apr 2013.

[135] The ATLAS collaboration. Identification of Jets Containing *b*-Hadrons with Recurrent Neural Networks at the ATLAS Experiment. Technical report, CERN, Geneva, 2017. All figures including auxiliary figures are available at `https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/PUBNOTES/ATL-PHYS-PUB-2017-003`.

[136] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan Salakhutdinov, and Alexander Smola. Deep sets. 2017. cite arxiv:1703.06114Comment: NIPS 2017.

[137] The ATLAS collaboration. Deep Sets based Neural Networks for Impact Parameter Flavour Tagging in ATLAS. Technical report, CERN, Geneva, 2020. All figures including auxiliary figures are available at `https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/PUBNOTES/ATL-PHYS-PUB-2020-014`.

[138] Freya Blekman, Florencia Canelli, Alexandre De Moor, Kunal Gautam, Armin Ilg, Anna Macchiolo, and Eduardo Ploerer. Jet Flavour Tagging at FCC-ee with a Transformer-based Neural Network: DeepJetTransformer. 6 2024.

[139] Huilin Qu, Congqiao Li, and Sitian Qian. Particle transformer for jet tagging. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *ICML*, volume 162 of *Proceedings of Machine Learning Research*, pages 18281–18292. PMLR, 2022.

[140] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R. Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *ICML*, 2018.

[141] Masanari Kimura, Ryotaro Shimizu, Yuki Hirakawa, Ryosuke Goto, and Yuki Saito. On permutation-invariant neural networks. *CoRR*, abs/2403.17410, 2024.

[142] Frédéric A. Dreyer and Huilin Qu. Jet tagging in the Lund plane with graph networks. *JHEP*, 03:052, 2021.

[143] Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges, 2021.

[144] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019.

[145] Francois Chollet et al. Keras, 2015.

[146] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing

Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[147] The CMS collaboration. Jet energy scale and resolution of jets with ParticleNet p_{T} regression using Run 3 data collected by the CMS experiment in 2022 and 2023 at 13.6 TeV. *CMS-DP-2024-064*, 2024.

[148] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *ICLR*. OpenReview.net, 2022.

[149] Albert Gu and Tri Dao. Mamba: Linear-Time Sequence Modeling with Selective State Spaces, 2024.

[150] Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. In *ICML*. OpenReview.net, 2024.

[151] Bo Peng, Daniel Goldstein, Quentin Anthony, Alon Albalak, Eric Alcaide, Stella Biderman, Eugene Cheah, Xingjian Du, Teddy Ferdinan, Haowen Hou, Przemyslaw Kazienko, Kranthi Kiran GV, Jan Kocon, Bartlomiej Koptyra, Satyapriya Krishna, Ronald McClelland Jr., Niklas Muennighoff, Fares Obeid, Atsushi Saito, Guangyu Song, Haoqin Tu, Stanislaw Wozniak, Ruichong Zhang, Bingchen Zhao, Qihang Zhao, Peng Zhou, Jian Zhu, and Rui-Jie Zhu. Eagle and finch: Rwkv with matrix-valued states and dynamic recurrence. *CoRR*, abs/2404.05892, 2024.

[152] Roger Waleffe et al. An Empirical Study of Mamba-based Language Models, 2024.

[153] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR, 2019.

[154] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *5th International Conference on Learning Representations*, 2016.

[155] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *ICLR 2018*, 2017.

[156] Ze Yu Zhao, Zheng Zhu, Guilin Li, Wenhan Wang, and Bo Wang. Generative Pre-training at Scale: Transformer-Based Encoding of Transactional Behavior for Fraud Detection, 2023.

[157] William J. Dally, Stephen W. Keckler, and David B. Kirk. Evolution of the Graphics Processing Unit (GPU). *IEEE Micro*, 41(6):42–51, 2021.

[158] The CMS Collaboration. Transformer models for heavy flavor jet identification. *CMS-DP-2022-050*, 2022.

[159] Vijay Prakash Dwivedi and Xavier Bresson. A Generalization of Transformer Networks to Graphs, 2021.

[160] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. In *ICCV*, pages 32–42. IEEE, 2021.

[161] John M. Campbell, R. Keith Ellis, Paolo Nason, and Emanuele Re. Top-pair production and decay at NLO matched with parton showers. *Journal of High Energy Physics*, 2015(4), apr 2015.

[162] S. Agostinelli et al. *GEANT4* — a simulation toolkit. *Nucl. Instrum. Meth. A*, 506:250, 2003.

[163] S. Chatrchyan et al. The CMS experiment at the CERN LHC. *JINST*, 3:S08004, 2008.

[164] A Dominguez et al. CMS Technical Design Report for the Pixel Detector Upgrade. Technical report, CERN-LHCC-2012-016, 2012.

[165] The CMS collaboration. Particle-flow reconstruction and global event description with the CMS detector. *Journal of Instrumentation*, 12(10):P10003, oct 2017.

[166] Jan Kieseler, Emil Bols, Mauro Verzetti, Caterina Vernieri, Devdatta Majumder, Loukas Gouskos, Markus Stoye, Seth Moortgat, Huilin Qu, Anna Stakia, and Henning Kirschenmann. Deepntuples, February 2020.

[167] Alexandre De Moor, Niclas Eich, Mate Farkas, Alexander Jung, Pavlo Kashko, and Ulrich Willemsen. b-hive. `https://gitlab.cern.ch/cms-btv/b-hive`, September 2023.

[168] Rene Brun et al. root-project/root: v6.18/02, June 2020.

[169] G Petrucciani, A Rizzi, and C Vuosalo. Mini-AOD: A New Analysis Data Format for CMS. *Journal of Physics: Conference Series*, 664(7):072052, December 2015.

[170] Jan Kieseler, Markus Stoye, Mauro Verzetti, Emil Bols, Anna Stakia, Henning KIRSCHENMANN, Huilin QU, Loukas GOUSKOS, and Swapneel Sundeep MEHTA. DeepJet, February 2020.

[171] Jan Kieseler, Markus Stoye, Mauro Verzetti, Pedro Silva, Swapneel Sundeep MEHTA, Anna Stakia, Yutaro IIYAMA, Emil Bols, Shah Rukh QASIM, Henning KIRSCHENMANN, Huilin Qu, Marcel Rieger, and Loukas Gouskos. DeepJetCore, February 2020.

[172] Marcel Rieger et al. riga/law: v0.1.18, February 2024.

[173] Marco Peruzzi, Giovanni Petrucciani, Andrea Rizzi, and for the CMS Collaboration. The NanoAOD event data format in CMS. *Journal of Physics: Conference Series*, 1525(1):012038, apr 2020.

[174] Adam Paszke et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[175] Jason Ansel et al. PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation. In *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24)*. ACM, April 2024.

[176] Matteo Cacciari and Gavin P. Salam. Pileup subtraction using jet areas. *Physics Letters B*, 659(1–2):119–126, January 2008.

[177] Lindsey Gray, Nicholas Smith, Andrzej Novak, Peter Fackeldey, Benjamin Tovar, Yi-Mu Chen, Gordon Watts, and Iason Krommydas. coffea (v2024.6.1), June 2024.

[178] Charles R. Harris et al. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[179] Yann Collet. The LZ4 compression algorithm, 2011.

[180] Jack Choquette, Wishwesh Gandhi, Olivier Giroux, Nick Stam, and Ronny Krashinsky. NVIDIA A100 Tensor Core GPU: Performance and Innovation. *IEEE Micro*, 41(2):29–35, 2021.

[181] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *ICLR (Poster)*. OpenReview.net, 2017.

[182] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

[183] Junjie Bai, Fang Lu, Ke Zhang, et al. ONNX: Open Neural Network Exchange. `https://github.com/onnx/onnx`, 2019.

[184] ONNX Runtime developers. ONNX Runtime. `https://onnxruntime.ai/`, 2021. Version: x.y.z.

[185] The CMS collaboration. Adversarial training for b-tagging algorithms in CMS. *CMS-DP-2022-049*, 2022.

[186] The CMS collaboration. A unified approach for jet tagging in Run 3 at $\sqrt{s}$=13.6 TeV in CMS. *CMS-DP-2024-066*, 2024.

[187] The CMS collaboration. Run 3 commissioning results of heavy-flavor jet tagging at $\sqrt{s}$ =13.6 TeV with CMS data using a modern framework for data processing. *CMS-DP-2024-024*, 2024.

[188] Armen Tumasyan et al. A new calibration method for charm jet identification validated with proton-proton collision events at $\sqrt{s}$ = 13 TeV. *Journal of Instrumentation*, 17(03):P03014, March 2022.

[189] G. Aad et al. Performance and calibration of quark/gluon jet taggers using 140 fb^{-1} of pp collisions at 13 TeV with the ATLAS detector. *Chinese Physics C*, 48(2):023001, February 2024.

[190] The CMS collaboration. Performance summary of AK4 jet b tagging with data from 2022 proton-proton collisions at 13.6 TeV with the CMS detector. *CMS-DP-2024-025*, 2024.

[191] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *ICLR*. OpenReview.net, 2017.

[192] Binghui Li, Jikai Jin, Han Zhong, John E. Hopcroft, and Liwei Wang. Why robust generalization in deep learning is difficult: Perspective of expressive power. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *NeurIPS*, 2022.

[193] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *ICLR*. OpenReview.net, 2021.

[194] Tao Li, Pan Zhou, Zhengbao He, Xinwen Cheng, and Xiaolin Huang. Friendly sharpness-aware minimization. In *CVPR*, pages 5631–5640. IEEE, 2024.

[195] Qihuang Zhong, Liang Ding, Li Shen, Peng Mi, Juhua Liu, Bo Du, and Dacheng Tao. Improving sharpness-aware minimization with fisher mask for better generalization on language models. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *EMNLP (Findings)*, pages 4064–4085. Association for Computational Linguistics, 2022.

[196] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv 1412.6572*, 12 2014.

[197] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial machine learning at scale. In *ICLR (Poster)*. OpenReview.net, 2017.

[198] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P. Xing, Laurent El Ghaoui, and Michael I. Jordan. Theoretically principled trade-off between robustness and accuracy. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 7472–7482. PMLR, 2019.

[199] Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom B. Brown, Dawn Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. Extracting training data from large language models. In Michael D. Bailey and Rachel Greenstadt, editors, *USENIX Security Symposium*, pages 2633–2650. USENIX Association, 2021.

[200] Nicholas Carlini, Matthew Jagielski, Christopher A. Choquette-Choo, Daniel Paleka, Will Pearce, Hyrum S. Anderson, Andreas Terzis, Kurt Thomas, and Florian Tramèr. Poisoning web-scale training datasets is practical. In *SP*, pages 407–425. IEEE, 2024.

[201] Sandy H. Huang, Nicolas Papernot, Ian J. Goodfellow, Yan Duan, and Pieter Abbeel. Adversarial attacks on neural network policies. In *ICLR (Workshop)*. OpenReview.net, 2017.

[202] Ezgi Korkmaz. Deep reinforcement learning policies learn shared adversarial features across mdps. In *AAAI*, pages 7229–7238. AAAI Press, 2022.

[203] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. In *ICLR (Poster)*. OpenReview.net, 2017.

[204] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial machine learning at scale. In *ICLR (Poster)*. OpenReview.net, 2017.

[205] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *ICLR (Poster)*. OpenReview.net, 2018.

[206] Nicolas Papernot, Patrick D. McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *EuroSP*, pages 372–387. IEEE, 2016.

[207] Yann LeCun, Corinna Cortes, and CJ Burges. MNIST handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

[208] Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy*, pages 39–57. IEEE Computer Society, 2017.

[209] Annika Stein, Xavier Coubez, Spandan Mondal, Andrzej Novak, and Alexander Schmidt. Improving Robustness of Jet Tagging Algorithms with Adversarial Training. *Computing and Software for Big Science*, 6(1), September 2022.

[210] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *ICLR (Poster)*. OpenReview.net, 2018.

[211] William Villegas, Angel Jaramillo-Alcázar, and Sergio Luján-Mora. Evaluating the Robustness of Deep Learning Models against Adversarial Attacks: An Analysis with FGSM, PGD and CW. *Big Data and Cognitive Computing*, 8:8, 01 2024.

[212] Hoki Kim, Woojin Lee, and Jaewook Lee. Understanding catastrophic overfitting in single-step adversarial training. In *AAAI*, pages 8119–8127. AAAI Press, 2021.

[213] Guillermo Ortiz-Jimenez, Pau de Jorge, Amartya Sanyal, Adel Bibi, Puneet K. Dokania, Pascal Frossard, Grégory Rogez, and Philip Torr. Catastrophic overfitting is a bug but it is caused by features, 2023.

[214] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 3319–3328. JMLR.org, 2017.

[215] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In Yoshua Bengio and Yann LeCun, editors, *ICLR (Workshop Poster)*, 2014.

[216] Scott M Lundberg and Su-In Lee. A Unified Approach to Interpreting Model Predictions. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[217] Nicolas Papernot, Patrick D. McDaniel, Ian J. Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In Ramesh Karri, Ozgur Sinanoglu, Ahmad-Reza Sadeghi, and Xun Yi, editors, *AsiaCCS*, pages 506–519. ACM, 2017.

[218] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael Wellman. Towards the Science of Security and Privacy in Machine Learning, 2016.

[219] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian J. Goodfellow, Dan Boneh, and Patrick D. McDaniel. Ensemble adversarial training: Attacks and defenses. In *ICLR (Poster)*. OpenReview.net, 2018.

[220] Anish Athalye, Nicholas Carlini, and David A. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In Jennifer G. Dy and Andreas Krause, editors, *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 274–283. PMLR, 2018.

[221] Theodoros Tsiligkaridis and Jay Roberts. Second Order Optimization for Adversarial Robustness and Interpretability, 2020.

[222] Yaguan Qian, Yuqi Wang, Bin Wang, Zhaoquan Gu, Yuhan Guo, and Wassim Swaileh. Hessian-Free Second-Order Adversarial Examples for Adversarial Learning, 2022.

[223] A.M. Sirunyan et al. Observation of the Higgs boson decay to a pair of τ leptons with the CMS detector. *Physics Letters B*, 779:283–316, April 2018.

[224] M. Aaboud et al. Cross-section measurements of the Higgs boson decaying into a pair of τ-leptons in proton-proton collisions at $\sqrt{s} = 13TeV$ with the ATLAS detector. *Physical Review D*, 99(7), April 2019.

[225] A.M. Sirunyan et al. Search for Higgs boson pair production in events with two bottom quarks and two tau leptons in proton–proton collisions at $\sqrt{s} = 13TeV$. *Physics Letters B*, 778:101–127, March 2018.

[226] M. Aaboud et al. Search for Resonant and Nonresonant Higgs Boson Pair Production in the $b\bar{b}\tau^+\tau^-$ Decay Channel in pp Collisions at $\sqrt{s} = 13TeV$ with the ATLAS Detector. *Physical Review Letters*, 121(19), November 2018.

[227] A. M. Sirunyan et al. Search for direct pair production of supersymmetric partners to the τ lepton in proton–proton collisions at $\sqrt{s} = 13$ TeV. *The European Physical Journal C*, 80(3), March 2020.

[228] A. M. Sirunyan et al. Search for heavy neutrinos and third-generation leptoquarks in hadronic states of two τ leptons and two jets in proton-proton collisions at $\sqrt{s} = 13$ TeV. *Journal of High Energy Physics*, 2019(3), March 2019.

[229] A. Tumasyan et al. Identification of hadronic tau lepton decays using a deep neural network. *Journal of Instrumentation*, 17(07):P07023, jul 2022.

[230] CMS Collaboration. Performance of τ-lepton reconstruction and identification in CMS. *Journal of Instrumentation*, 7(01):P01001–P01001, January 2012.

[231] A.M. Sirunyan et al. Reconstruction and identification of τ lepton decays to hadrons and $\nu\tau$ at CMS. *Journal of Instrumentation*, 11(01):P01019, jan 2016.

[232] A.M. Sirunyan et al. Performance of reconstruction and identification of τ leptons decaying to hadrons and $\nu\tau$ in pp collisions at $\sqrt{s}$=13 TeV. *Journal of Instrumentation*, 13(10):P10005, oct 2018.

[233] B Todd Huffman, Charles Jackson, and Jeff Tseng. Tagging b quarks at extreme energies without tracks. *Journal of Physics G: Nuclear and Particle Physics*, 43(8):085001, July 2016.

[234] Yuichiro Nakai, David Shih, and Scott Thomas. Strange Jet Tagging. 3 2020.

[235] J. Erdmann, O. Nackenhorst, and S.V. Zeißner. Maximum performance of strange-jet tagging at hadron colliders. *Journal of Instrumentation*, 16(08):P08039, August 2021.

[236] P. Abreu et al. Measurement of the strange quark forward backward asymmetry around the Z0 peak. *Eur. Phys. J. C*, 14:613–631, 2000.

[237] Nicola Cabibbo. Unitary Symmetry and Leptonic Decays. *Phys. Rev. Lett.*, 10:531–533, Jun 1963.

[238] Makoto Kobayashi and Toshihide Maskawa. CP-Violation in the Renormalizable Theory of Weak Interaction. *Progress of Theoretical Physics*, 49(2):652–657, 02 1973.

[239] Ahmed Ali, Fernando Barreiro, and Theodota Lagouri. Prospects of measuring the CKM matrix element $|V_{ts}|$ at the LHC. *Physics Letters B*, 693(1):44–51, September 2010.

[240] J. Duarte-Campderros, G. Perez, M. Schlaffer, and A. Soffer. Probing the Higgs–strange-quark coupling at e^+e^- colliders using light-jet flavor tagging. *Physical Review D*, 101(11), June 2020.

[241] The CMS collaboration. Performance of quark/gluon discrimination in 8 TeV pp data. *CMS-PAS-JME-13-002*, 2013.

[242] J. Erdmann. A tagger for strange jets based on tracking information using long short-term memory. *Journal of Instrumentation*, 15(01):P01021–P01021, January 2020.

[243] A.M. Sirunyan et al. Measurement of charged pion, kaon, and proton production in proton-proton collisions at $\sqrt{s}$=13 TeV. *Physical Review D*, 96(11), December 2017.

[244] The CMS collaboration. Optimizing the pileup per particle identification algorithm in the context of τ_h lepton identification in Run 3. *CMS-DP-2024-043*, 2024.

[245] The ATLAS collaboration. Graph Neural Network Jet Flavour Tagging with the ATLAS Detector. *ATL-PHYS-PUB-2022-027*, 2022. All figures including auxiliary figures are available at https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/PUBNOTES/ATL-PHYS-PUB-2022-027.

[246] The ATLAS collaboration. Jet Flavour Tagging With GN1 and DL1d. Generator dependence, Run 2 and Run 3 data agreement studies. *ATL-PLOT-FTAG–01*, 2023.

[247] The CMS Collaboration. A deep neural network to search for new long-lived particles decaying to jets. *Machine Learning: Science and Technology*, 1(3):035012, aug 2020.

[248] Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y. Hou, and Max Tegmark. KAN: Kolmogorov–arnold networks. In *Submitted to The Thirteenth International Conference on Learning Representations*, 2024. under review.

[249] Stano Tokar. Jet charge determination at the LHC. In *Parton radiation and fragmentation from LHC to FCC-ee*, pages 79–84, 2 2017.