



DEPARTMENT OF
**COMPUTER
SCIENCE**

UNIVERSITY OF OXFORD

MSC IN COMPUTER SCIENCE
PROJECT DISSERTATION

Machine Learning in Particle Physics Quark versus Gluon Jet Discrimination

Author:

Maxence DRAGUET

Supervisors:

Prof. Alan BARR

Dr. Anisoara CALINESCU

Trinity Term
2019-2020



Abstract

This is a multidisciplinary project, adapting, applying, and assessing modern data analysis techniques and advanced machine learning in the field of Particle Physics. The objective is to derive a new technique to perform quark versus gluon jet discrimination with the ATLAS detector, one of the main experiments on the Large Hadron Collider at CERN. Different machine learning architectures targeting generated level simulations (not accounting for detector effects) have recently been explored in the literature with promising results. Among them, the Junipr approach learns a probabilistic model scaffolded on a leading-order model of Physics (jet factorisation algorithm) by recurrent learning and deep neural network mapping. It was proven in 2019 to lead to state-of-the-art quark versus gluon jet discrimination on generated level simulations, with the added quality that the model thus learnt is interpretable. The project aims to implement and adapt this method into an efficient discriminator for the ATLAS collaboration, using the particular conditions of its detector and simulations comparable to the data collected (the reconstructed level). An important constraint added to these restrictions is to work with low-level data to pave the way for future research on a possible trigger application. The results show that a Junipr approach given low-level information succeeds in outperforming boosted decision tree and deep neural network models using high-level information, in a specific energy region. The adapted framework is also shown to obtain similar performance to that of the original paper while restraining to low-level variables and fully reconstructed simulations. Finally, Junipr stays globally competitive compared to the benchmarking models in all energy regions of interest, suggesting a trigger implementation should be considered for the ATLAS experiment.

Word count: 27,854 (without the Appendices and Acknowledgements)

Acknowledgements

I would like to start by deeply thanking the Wiener-Anspach foundation for providing me with the scholarship that made this Oxford experience a possibility.

I am deeply grateful for the continuous and precious support that both my supervisors provided all along this project. Each weekly meeting offered invaluable opportunities to learn from their insightful and interesting remarks and advice. Thank you Dr. Calinescu for accepting to embark with me on this interdisciplinary project and for your manifest interest in delving into the field of Particle Physics. Thank your Professor Barr for adapting this project on such short notice, as well as for giving me access to your team's weekly meetings, an extraordinary environment of world-wide specialists, and first-class infrastructure.

I wish to express my sincere gratitude for the key help Aaron O'Neill provided in this project. His helpful and methodological guidance, his availability, and his profound understanding of Particle Physics have been crucial components of the project. I particularly thank him for his help in gathering the data and correcting the algorithm to process it with the ATLAS software. I would also like to express my unbounded gratitude for introducing me to UpRoot, giving the invaluable option to work in Python and avoid ROOT as much as possible.

Finally, I would like to extend my deepest gratitude and appreciation to both my family and my friends. For encouraging me during this project and before, for supporting me during the strange months of confinement, and for helping me to get settle in Oxford.

Contents

1	Introduction	7
2	Jet Tagging in High Energy Physics	11
2.1	High Energy Physics, the Standard Model, and the Experimental Setup . . .	11
2.2	Motivation and Open Questions	20
2.3	Objectives and Organisation	23
3	Methodology	26
3.1	Data	26
3.1.1	Data Collection	28
3.1.2	Data Processing	29
3.1.3	Analysing the Data	40
3.2	Machine Learning Models	45
3.2.1	Boosted Decision Trees	45
3.2.2	Deep Neural Networks	45
3.2.3	Junipr	47
3.2.3.1	Mathematical Basis of Junipr	47
3.2.3.2	Machine Learning Implementation of Junipr	49
3.2.3.3	Applications of Junipr	54
3.2.3.4	Optimising Junipr	57
4	Results and Analyses	63
4.1	Limitations of Boosted Decision Trees and Deep Neural Networks	63
4.2	Unary Junipr	70
4.3	Binary Junipr	82
4.4	Comparison of Binary Junipr to the BDT and DNN Benchmarks	85
5	Conclusions and Future Work	104
	Bibliography	112
	Appendices	119
	Appendix A Global Project Information	A. 1
	A.1 Overview of the Project Codebase	A. 1
	Appendix B Additional Plots	B. 5
	B.1 Diagnostic Plots of Sets 4 and 6	B. 5
	Appendix C Additional information on the Junipr Model	C. 10
	C.1 Additional Diagrams of Junipr Analysis	C. 10
	C.2 Information on the Trained Junipr Models	C. 12
	C.3 Scaling Function for the Junipr model	C. 18
	C.4 Additional Junipr Trees	C. 22

List of Figures

2.1.1	Particles in the SM	12
2.1.2	A multijet event in the ATLAS detector	14
2.1.3	Accelerator complex of CERN in 2019	15
2.1.4	Schema of the ATLAS detector	16
2.1.5	Coordinate system used by the ATLAS experiment	18
3.1.1	Example of a tree produced by jet clustering	31
3.1.2	Two different sequential recombination algorithm on the same jet	33
3.1.3	From one anti- k_T jet to two C-A jets	35
3.1.4	Histogram of miss-reconstructed jets	35
3.1.5	Jet energy distributions in sets 4 and 6	41
3.1.6	Jet energy distributions in set 4 in log scale	41
3.1.7	Jet energy distributions in sets 4 and 6 matched	42
3.1.8	Jet p_T distributions in sets 4 and 6	42
3.1.9	Jet number of constituents distributions in sets 4 and 6	43
3.1.10	Jet width distribution in sets 4 and 6	43
3.1.11	EMTopoJet constituent mass and ϕ distributions in set 4	44
3.2.1	Deep neural network structure	46
3.2.2	Diagrammatic representation of Junipr	51
3.2.3	Scaling function for Junipr	59
4.1.1	ROC curves BDT and DNN on set 2	67
4.1.2	ROC curves BDT and DNN on set 3	67
4.1.3	Feature importance of BDT model on set 3	68
4.2.1	Learning curves of 6 epochs unary models on set 4	72
4.2.2	Learning curves of 16 epochs unary Junipr on set 4	73
4.2.3	Quark jets from the 16 epochs unary model on set 4	75
4.2.4	Gluon jets from the 16 epochs unary model on set 4	75
4.2.5	Quark and gluon unary Junipr ending and mother index probability distributions	79
4.2.6	Quark and gluon unary Junipr θ and δ probability distributions	80
4.2.7	Quark and gluon unary Junipr z and ϕ probability distributions	81
4.3.1	Learning curves of 19 epochs binary Junipr on set 4	83
4.3.2	Score distribution for the 19 epochs binary Junipr model on set 4	84
4.3.3	Two jet trees from set 4 analysed by binary Junipr <i>BJ 1</i>	85
4.4.1	ROC curves of binary Junipr vs BDT and DNN on set 4	86
4.4.2	Score distribution of the BDT and DNN models on set 4	86
4.4.3	Ratio of normalised score distribution DNN/binary Junipr for set 4	87
4.4.4	ROC curves of <i>BJ 2</i> vs <i>BJ 1</i> the BDT, and the DNN	88
4.4.5	ROC curves of <i>BJ 2</i> for different energy bins	90
4.4.6	Evolution of <i>BJ 2</i> score across energy bins of set 5	91

4.4.7	ROC curves of <i>BJ 2</i> versus <i>BJ 3</i> on the 100 to 200 GeV bin of set 5	92
4.4.8	AUC of <i>BJ 1</i> , <i>BJ 2</i> , and <i>BJ 4</i> versus the benchmarking BDT and DNN as a function of energy	93
4.4.9	Feature importance of the BDT model on set 6	95
4.4.10	AUC of <i>BJ 1</i> , <i>BJ 2</i> , and <i>BJ 5</i> as a function of energy	97
4.4.11	2D histogram of miss-labelled nodes: jet energy vs fraction of nodes (<i>BJ 2</i> on set 5)	98
4.4.12	2D histograms of miss-labelled nodes: relative depth of the nodes vs score (<i>BJ 2</i> on set 5)	99
4.4.13	2D histograms of miss-labelled nodes: mother energy vs score (<i>BJ 2</i> on set 5)	100
4.4.14	Two jet trees from set 5 in the above 500 GeV region analysed by binary Junipr <i>BJ 2</i>	103
A.1.1	The architecture of the main codebase	A. 2
B.1.1	Number of constituents distribution before cuts in sets 4 and 6	B. 6
B.1.2	Distribution of the sum of track momentum above 500 GeV	B. 6
B.1.3	Distribution jet η in sets 4 and 6	B. 7
B.1.4	Distribution jet ϕ in sets 4 and 6	B. 7
B.1.5	Distribution of jet mass in sets 4 and 6	B. 8
B.1.6	Distribution jet energy collected in ECAL in sets 4 and 6	B. 8
B.1.7	Distribution fraction of p_T associated to charge particles in sets 4 and 6 .	B. 8
B.1.8	Distribution number of tracks with p_T above 500 GeV in sets 4 and 6 . .	B. 9
B.1.9	Distribution number of tracks with p_T above 1000 GeV in sets 4 and 6 . .	B. 9
C.1.1	Diagrammatic representation of Junipr at first step	C. 11
C.1.2	Diagrammatic representation of Junipr at last step	C. 11
C.2.1	Learning curves of 16 epochs unary Junipr on set 5	C. 14
C.2.2	Learning curves of 26 epochs binary Junipr on set 5	C. 14
C.2.3	Learning curves of 50 epochs unary Junipr on the 100 to 200 GeV energy bin of set 5	C. 15
C.2.4	Learning curves of 14 epochs binary Junipr on the 100 to 200 GeV energy bin of set 5	C. 15
C.2.5	Learning curves of the 16 epochs large unary Junipr models on set 5 . .	C. 16
C.2.6	Learning curves of 26 epochs large binary Junipr model on set 5	C. 16
C.2.7	Learning curves of the 16 epochs large unary Junipr models on set 5b . .	C. 17
C.2.8	Learning curves of 26 epochs large binary Junipr model on set 5b	C. 17
C.3.1	Energy scaling function	C. 19
C.3.2	Mass scaling function	C. 20
C.3.3	z scaling function	C. 20
C.3.4	θ scaling function	C. 21
C.3.5	δ scaling function	C. 21
C.4.1	Unary Junipr trees from models <i>UJ Q1</i> and <i>UJ G1</i> on set 4	C. 22
C.4.2	Binary Junipr trees from model <i>BJ 2</i> from several energy bins of set 5 .	C. 23

List of Tables

3.1.1	Data collected for Junipr	37
3.1.2	Datasets collected and produced	38
3.1.3	Cuts for set 1, 2, and 3	39
3.1.4	Cuts for set 4, 5, and 6	39
4.1.1	Variables from sets 1, 2, and 3 for BDT's and DNN's	65
4.1.2	Variables from set 6 for BDT and DNN models	70
4.2.1	Training schedule for unary Junipr	71
4.2.2	Extension of the training schedule for unary Junipr	73
4.3.1	Training schedule for binary Junipr	82
4.4.1	Long extended training schedule for unary Junipr	91
C.2.1	Junipr models trained	C. 13
C.3.1	Scaling parameters	C. 19

List of Abbreviations

AdaBoost Adaptive Boosting	MC Monte Carlo
AUC Area Under the Curve	ML Machine Learning
BDT Boosted Decision Tree	MVA Multivariate Analysis
C-A Cambridge-Aachen	NLP Natural Language Processing
CNN Convolutional Neural Network	PU Pile-Up
DNN Deep Neural Network	PV Primary Vertex
GNN Graph Neural Network	QCD Quantum Chromodynamics
GRU Gated Recurrent Unit	QFT Quantum Field Theory
HEP High Energy Physics	RNN Recurrent Neural Network
IP Interaction Point	ROC Receiver Operating Characteristic
LHC Large Hadron Collider	SGD Stochastic Gradient Descent
LO Leading Order	SM Standard Model
LSTM Long Short Term Memory	WD Weight Decay

CHAPTER 1

Introduction

Particle Physics is the branch of Physics investigating the nature of the constituents of our Universe. In this quest, it aims to uncover every aspect of elementary particles, the smallest irreducible subset of elements, and the way they fundamentally interact. This effort has spanned centuries, with many important theoretical and experimental discoveries realised in the last decades. Quantum mechanics, quantum field theory, atomic and molecular physics, astrophysics, accelerator complexes, radiotherapy, and, in some sense, the Web and touchscreen technology are all born of this active field of investigation.

The current landscape of Particle Physics is a deeply puzzling one. The 2012 observation of the Higgs boson [1, 2, 3] marked the latest resounding success of the Standard Model. A patchwork of interconnected quantum field theories developed in the second part of the 20th century, this model is currently our best description of the constituents of matter and the way they interact under the strong, weak, and electromagnetic interactions [4]. While the extent and precision of this theoretical edifice is unparalleled in the history of science, to this day, it still lacks an explanation for some essential aspects of our Universe: from gravity, to dark matter and energy, the origin of the mass of neutrinos, the fine-tuned hierarchy of effects, and the matter-antimatter asymmetry. The main challenge of contemporary Physics is to extend its reach to address these important elements, a fertile

ground that has seen a plethora of ideas grow into theories like Supersymmetry, String Theory, and Grand Unified Theory.

The objective of experimental Particle Physics is to unlock the current situation by either uncovering evidence of beyond the Standard Model Physics or situations where its predictions diverge from observations. To achieve this goal, the world's largest machine and most powerful accelerator was built in 2008 at CERN: the Large Hadron Collider. Indeed, physicists explore the subatomic world by colliding large numbers of protons and analysing the particles emerging from the subsequent interactions. A wide range of information from these events is recorded by giant detectors built around the collision point. Analysing the samples thus gathered, the validity of theoretical predictions can be tested against observations. This experimental setup requires a large number of computational methods to process the huge volume of collected data into a usable sample. Note that, on a typical day of operation, the different detectors of the Large Hadron Collider generate about 1 PB of collision data per second and collect about 100 TB daily [5, 6]. For a point of comparison, Facebook's daily traffic averages about 4 PB [7]. Computational efficiency is thus a vital part of modern experimental Physics.

This work presents an exploratory search for an efficient quark versus gluon jet classifier using interpretable machine learning methods fed low-level information. The broad objective is to study whether a new recurrent-learning based algorithm should be added to the toolbox of experimental physicists of the ATLAS collaboration at CERN. While boosted decision trees and simple deep neural networks using highly-engineered variables are still prevalent in the field, thanks to their easy out-of-the-box deployment, several techniques recently proposed in the literature for this particular topic have shown promising results (as discussed in section 2.2). The specific method studied in this work is the Junipr approach, introduced in [8] and extended in [9] into the current state-of-the-art on a specific set of simulated quark and gluon jets using high-level variables at the generated level (no detector simulation introduced). Deriving an efficient and fast algorithm to discriminate quark and gluon jets in the ATLAS experiment could prove extremely useful in its quest to unravel the laws of our Universe at its smallest scale.

The present project concerns a specific part of an extremely large edifice and, along with this limited perspective, the background introduction of chapter 2, section 2.1, is restricted to the “heart” of the matter: what quarks, gluons, and jets are and how we can observe them. Following these physics questions, section 2.2 moves on to the main research areas and methods to discriminate these objects as well as some recent results, thereby providing a brief state-of-the-art. Section 2.3 aims to reframe these points into a concrete and coherent set of research questions investigated in this project.

To assess the effectiveness and properties of the methods studied, a dataset was collected and processed such as to satisfy strict criteria. The data samples must be physically sound, match the specific properties of the ATLAS detector, and have a format compatible with the models targeted. Boosted decision trees and deep neural network models are chosen to represent benchmarking methods, as they are widely used in the field. They also represent high-level models since they rely on engineered-variables to function. Junipr is the specialised model explored in this project to try to improve the efficiency on the discrimination task. It also serves the purpose of the low-level model, by adapting it to work with calorimeter information. This specific technique was selected among a larger set of options as it led to state-of-the-art results on generated level simulation in 2019. These different models need to be carefully implemented. This is particularly important for Junipr since the bases of its predictions are interpretable when founded upon a correct physical model. These elements are discussed in depth in chapter 3, dedicated to the global methodology of the project. The first part of the chapter focuses on the steps taken to collect the data (section 3.1.1), process it (section 3.1.2), and analyse the datasets produced (section 3.1.3). The methods implemented to serve as benchmarking models are then presented in section 3.2.1, for boosted decision trees, and in section 3.2.2, for deep neural networks. The presentation of the Junipr approach is organised in four stages. After presenting the mathematical basis of the model in section 3.2.3.1, the architecture of the machine learning implementation is revealed in section 3.2.3.2. The two final sections are dedicated to the range of applications of Junipr (section 3.2.3.3) and some personal optimisation tricks and methods to improve the efficiency of the model (section 3.2.3.4).

The different pieces of the puzzle gathered, chapter 4 next presents the exploration of the subjects selected. It starts, with section 4.1, by introducing some results of the benchmarking models on a dataset of common processes generating quark and gluon jets. The objective of this first discussion is to experimentally motivate why boosted decision trees and deep neural networks are not enough. These limitations demonstrated, the capabilities of unary Junipr models are presented in section 4.2. The methodology adopted to train, construct, and analyse these models is extensively discussed and supported by different experiments. Binary Junipr models are the subject of the next section (section 4.3), adopting a similar approach to uncover some general points. The core of the project can then be unveiled in section 4.4, comparing the performance of binary Junipr to the benchmarking models for the task of quark versus gluon jet discrimination. Several experiments, sequentially conducted to address questions raised by previous observations, are presented in the same causal order to investigate the subtle effects uncovered. Finally, the models are extensively analysed to clarify their respective efficiency, weaknesses, and strengths.

Chapter 5 concludes this dissertation by summarising the important conclusions drawn from each step of the work: from the collection of data to the final analyses of the models. The main personal contribution is the adaptation of Junipr to work at the reconstructed level of the ATLAS experiment with low-level calorimeter information. Building upon these results, it presents ideas for extensions of the work. The most important points argued suggest implementing the studied Junipr model as an online trigger for the ATLAS experiment, and further adapting the model to the specific structure of the detector.

Throughout the project, a strong emphasis is placed on the interdisciplinarity between Particle Physics and Machine Learning. This work was, after all, pursued as a collaboration with Professor Alan Barr and DPhil student Aaron O'Neill of the Particle Physics group of the University of Oxford. This dissertation aims to discuss elements of pertinence to both fields and introduce the physics thematics in the depth required to fully motivate the importance of an efficient quark versus gluon jet discriminator for the ATLAS experiment.

Jet Tagging in High Energy Physics

The scope of the project requires an interdisciplinary approach combining elements of High Energy Physics (HEP) with computer science methods. This chapter aims to offer an overview of the physics context by introducing the main ideas, the relevant theoretical knowledge, the experiment setup, and the different techniques employed in the task of quark versus gluon jet discrimination.

2.1 High Energy Physics, the Standard Model, and the Experimental Setup

To date, the Standard Model (SM) has been the most successful theory in describing the constituents and the dynamics of matter [4]. It can boast 55 Nobel prizes and is often hailed as the most successful theory of science [10], with the ability to predict and model some mechanisms of the Universe to a staggering degree of precision. For example, the anomalous magnetic dipole moment predictions agree with measurements to more than 10 significant figures [11]. Without entering any mathematical details, the SM is a set of pieces called fields and their dynamics: some describing matter itself (*fermions*, such as the electron) and others (*bosons*, such as the W and the famous Higgs boson H) how

this matter interacts under the electromagnetic, weak, and strong interactions with other pieces of matter. To each particle corresponds an *antiparticle*, in all ways similar except for having the opposite electrical charge (e.g., an electron e^- has a charge -1, in electron units, and the antielectron, called positron e^+ , a charge +1). Figure 2.1.1 displays these different components. Note that the masses listed in figure 2.1.1 are in a specific unit of energy called the GeV (10^9 eV where 1 eV is the energy of an electrical charge equal to that of the electron accelerating from rest in a potential of 1 volt). The mass m and energy in the rest frame E are connected by the famous equation $E = mc^2$, with c the speed of light in vacuum. Both the momentum and the mass are often indicated in units of GeV, with the appropriate factor of c to be placed back when necessary (for example, a mass in GeV can be re-expressed in the right unit with GeV/c^2). It should be restated, as mentioned in the introduction, that the SM is only but a partial model, lacking an explanation for some important aspects of the Universe, such as gravity but also dark matter and dark energy.

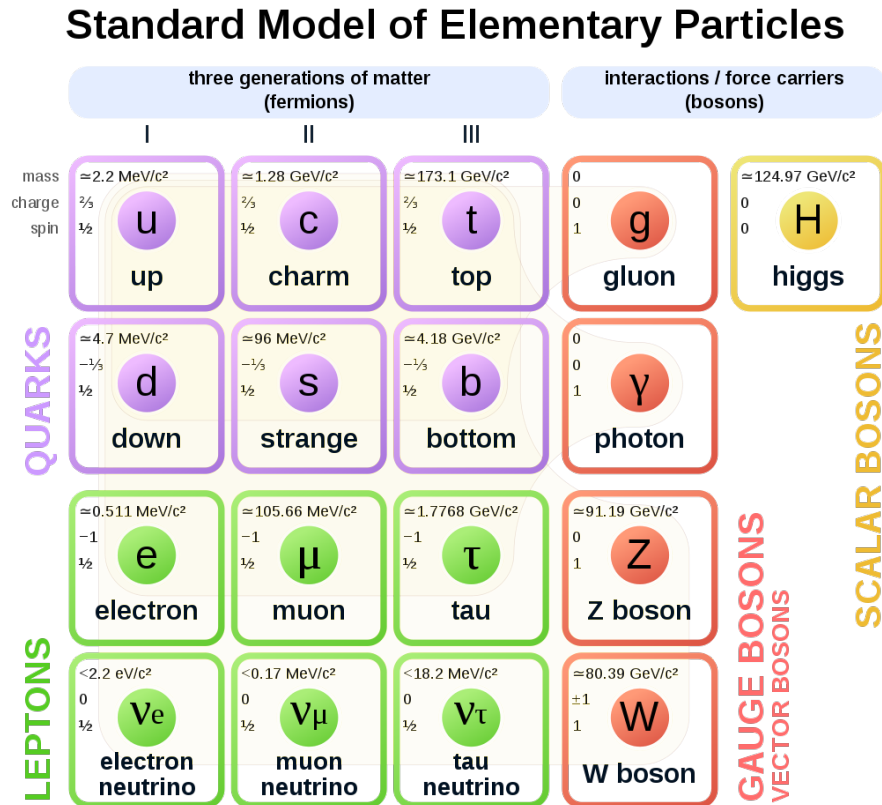


Figure 2.1.1 – Elementary particles of the Standard Model [12]. Elementary fermions are listed in the three left columns and elementary bosons in the two right ones.

In figure 2.1.1, the two top left purple rows list *quarks* (declined into different *flavours*, such as the up u and the down d), the only elementary fermions sensitive to the strong interaction. This mechanism is described by the Quantum Field Theory (QFT) of Quantum Chromodynamics (QCD). *Colour* stands for the elementary strong charge and comes in three variants: blue, red, and green. The mediator of this interaction, called the *gluon* (g), is itself colour-charged (with two charges in fact) and massless, which has a deep implication on the behaviour of quarks and gluons called *colour-confinement*. This effect can be (approximately) understood as the strength of the colour-bound increasing with the distance of separation, forcing coloured-components to aggregate. The image of a string is often mentioned to describe this particular type of bond. Because opposite colours or the combination of all colours neutralise the charge, they combine to form *colourless* objects called *hadrons* (Greek for “massive”, used as an opposition to *leptons*, meaning “light” and corresponding to the two bottom left rows of figure 2.1.1). These aggregated states consist in *mesons* (a quark and an antiquark, such as the pion - Greek for “intermediate”), *baryons* (a set of three quarks, such as the proton - Greek for “heavy”), and more exotic compounds, such as *tetraquarks* (a set of four quarks). The process forcing the combination of coloured elementary particles into colourless objects is called *hadronisation*. A complete theoretical description of this mechanism has not yet been fully derived, with several phenomenological approaches widely adopted [13].

Given that both quarks and gluons have a colour charge, colour-confinement forces them to combine with other coloured matter to form colourless aggregates. Note that where that matter comes from precisely is still an active subject of investigation. In the SM, these external particles are presumed to emanate from quark-antiquark pairs spontaneously created from the vacuum. The result of their hadronisation is a cascade of combinations and decays that form a tight cone of particles around the trajectory of the hadronising particle. Indeed, by the law of energy-momentum conservation, the momentum $\vec{p}$ of the initial particle is shared between the final state constituents, hence the general global orientation. Such a conic structure of decays is called a *jet*. A real ATLAS event with multiple jets is represented in figure 2.1.2. The process of gathering final state constituents to reconstruct the global jet is called *clustering* [13]. It is performed by *clustering algorithms* with the

whole history of decays organised into a *factorisation tree*, as discussed in more details in section 3.1.2.

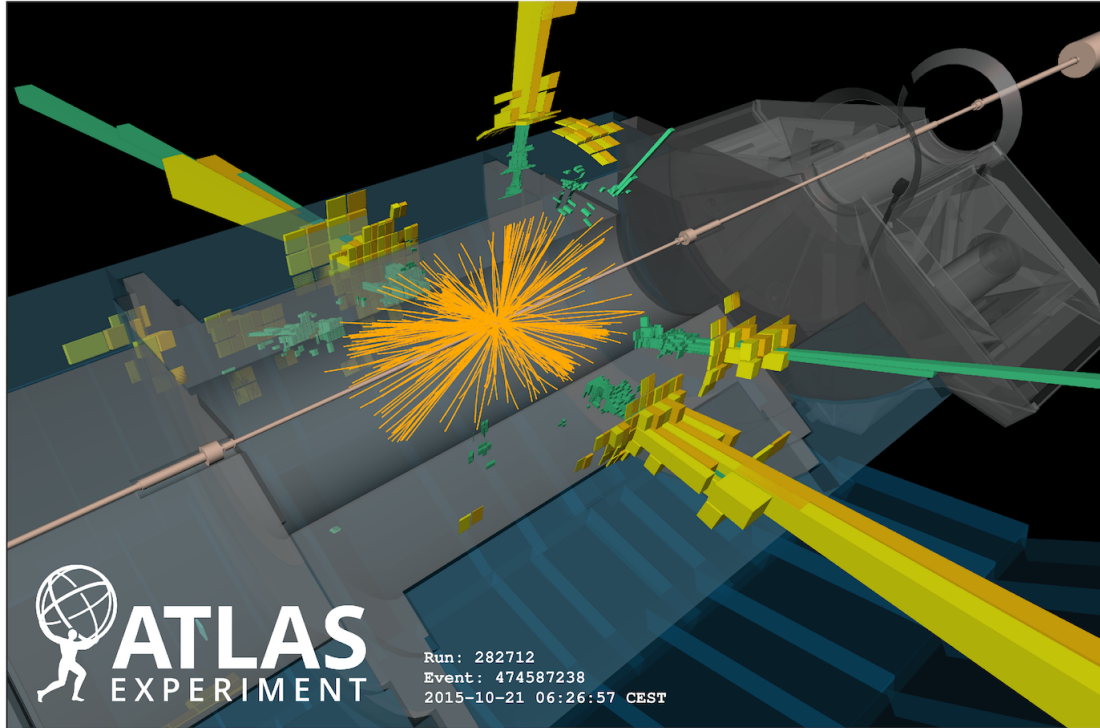


Figure 2.1.2 – A multijet event in the ATLAS detector [14]. The green and yellow volumes represent energy deposits in specific parts of the detector (ECAL and HCAL). The small orange curves in the centre are individual particle tracks. Note how many of them are grouped behind each large energy deposit, an evident sign of the presence of a jet.

To observe quark and gluon jets as well as any other animal of this particle zoo, massive accelerator complexes and detector systems were assembled. This short introduction focuses on the ATLAS Experiment at CERN, since this project is calibrated to their data input. In Physics, energy and (inverse) length are connected, as suggested by the famous Planck-Einstein relation $E = \frac{hc}{\lambda}$, relating the energy E to Planck’s constant h , the speed of light in vacuum c and the wavelength λ . To observe or, equivalently, to measure requires to reach a wavelength of the size of the investigated objects and therefor the right energy. The smaller the object, the larger the energy required. The interest of accelerators is precisely to increase the kinetic energy of components before “releasing” their content through collisions but, more generally speaking, interactions. New elements are produced from the decay of the initial particles that unveil the processes ruling Physics at the energy investigated. Most accelerators, such as the CERN’s Large Hadron Collider (LHC), exploit

a circular shape to keep the beams of charged particles, typically gathered into bunches, in a closed trajectory while their speed (and hence kinetic energy) is increased with each revolution by the use of radiofrequency cavities. This device generates an oscillating electromagnetic field synchronised with the evolving revolution frequency of the bunches to accelerate them. The beams are constrained to their expected trajectory by powerful magnetic systems. Different geometries, such as dipoles and quadrupoles are used to monitor, control, and shape the extent of the bunches. The LHC is only the last ring in a complex chain of accelerators, as suggested in the diagram of the CERN operation presented in figure 2.1.3. It operates with pre-accelerated protons or *ions* (atoms or molecules with a net electrical charge).

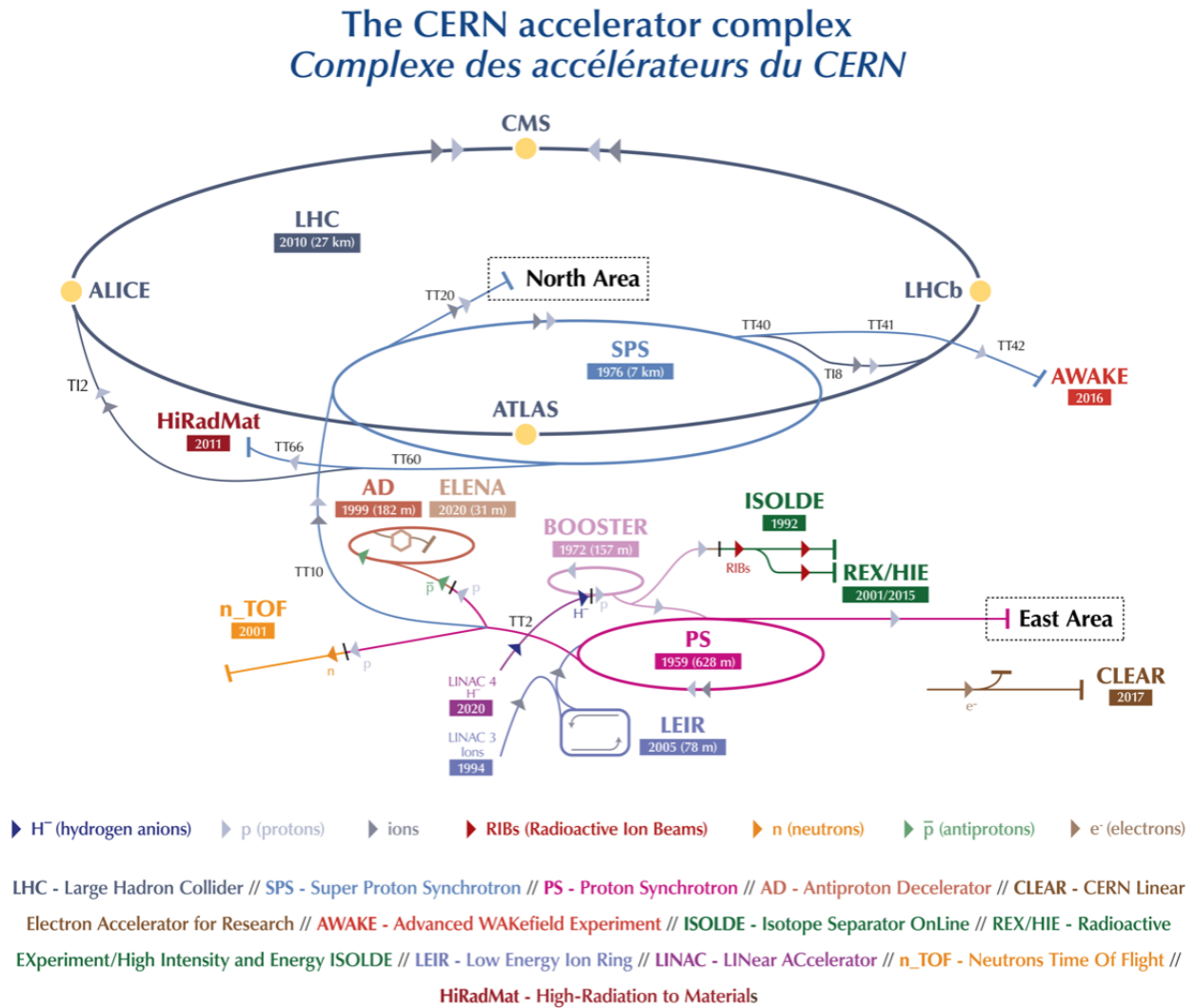


Figure 2.1.3 – The accelerator complex of CERN in 2019 [15]. The LHC is the top dark blue ring.

Along their trajectory, two different beams can collide at a controlled *Interaction Point (IP)*. The word *event* is used to encompass all information and processes emanating from the collision of two bunches (all collisions within a 25 ns period). For the purpose of this project, the collided elements in the two beams are protons at a centre of mass energy of 13 TeV, where $1 \text{ TeV} = 1000 \text{ GeV}$. At these energies, the constituents of protons, as perceived by observing their decay, do not match the often presented and simplified three quarks picture ($2 u$ and $1 d$). They are made of a probabilistic set of constituents called *partons*. *Parton distribution functions* were phenomenologically derived to quantify the probability distribution of observing a given parton at a given momentum transfer due to the interaction undergone [16]. In a sense, the content of a proton is governed by a probability distribution depending on the energy at which the proton is observed. Physicists analyse the constituents of matter arising from the collision by building giant detectors around this particular location. The ATLAS collaboration uses a cylindrically shaped multi-layered detector that is 45 m long with a diameter of 26 m and sitting in a cavern 100 m below ground, as presented in figure 2.1.4.

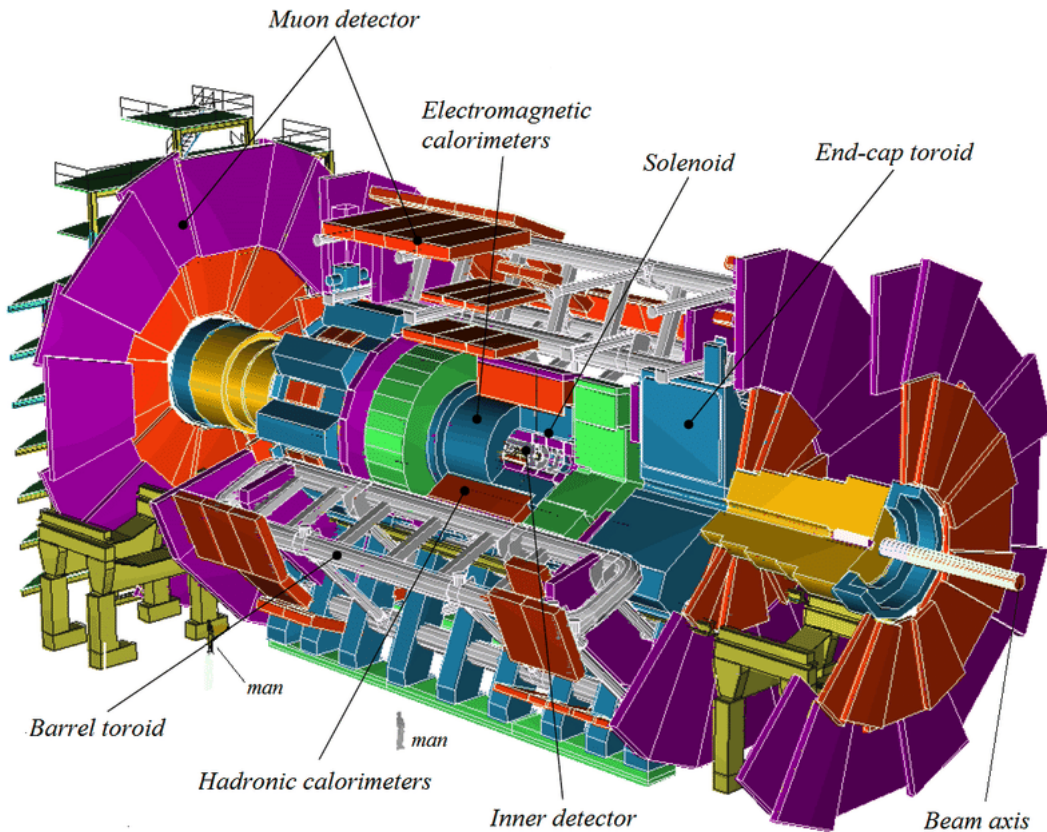


Figure 2.1.4 – Schema of the ATLAS detector [17].

Such a detector is far more versatile than needed for the present project. This short presentation focuses on its essential components. Following a radius from the IP (in the centre of the detector) towards the exterior, the main elements encountered are [18]:

- An inner detector: different technologies are deployed to be sensitive to the presence of charged particles in active cells. It allows the reconstruction of a *track* (a path) for electrically charged particles going through its layers, lending to the device the name of *track detector* or *tracker*. A variety of algorithms are employed in the ATLAS experiment to fit a track to a set of hits. A *hit* is a localised deposit of energy, usually in a tracker or a calorimeter cell. A point from which new tracks emerge is called a *vertex*, with the *Primary Vertex (PV)* being the main vertex of the event, from which most of the interesting tracks come.
- A solenoid: a large cylindrical magnet releasing a magnetic field parallel to the axis of the cylinder, along the beamline. It has the effect of curving the trajectory of charged particles, as dictated by the famous Lorentz force $\vec{F} = q\vec{v} \times \vec{B} + q\vec{E}$, q being the charge of the particle, $\vec{v}$ its velocity, and where $\vec{B}$, the magnetic flux density, and $\vec{E}$, the electrical field, are taken at the position of the particle. A curved trajectory in this fixed magnetic field reveals the electric charge of a particle (since a flip of q changes the direction of rotation for a given $\vec{v}$). Another advantage of curving the trajectory of a charged particle with a fixed magnetic field comes from the ability to estimate its momentum $\vec{p}$ by measuring the radius of curvature ρ with the relation $p = qB\rho$. Note that for a given p , one can control ρ by adapting B : this is exactly what the magnetic system of the accelerator does to constrain the bunches to their expected trajectories.
- Calorimeters: electromagnetic (ECAL, for electrons, positrons, and photons) and hadronic (HCAL, for hadrons) calorimeters are large active volumes which purpose is to collect (most of) the energy of a passing particle. The calorimeters are structured into cells (separated into a forward region, called *endcaps*, and a central region, called the *barrel*). A cell collecting an amount of energy over a certain threshold is recorded as a hit. In the ATLAS experiment, groups of cells thus triggered are clustered into larger structures called *topoclusters*, for topological clusters.

The reference frame adopted in the present work is introduced in figure 2.1.5, with the two angles being the polar angle θ and the azimuthal angle ϕ . The x -axis points towards the centre of the LHC ring, the z -axis is taken along the beamline, and the y -axis closes the basis, so that the system $x - y - z$ is right-handed. Variables living in the $x - y$ plane are referred to as *transversal* and added the subscript “ T ”. For example, the energy projected in the transverse plane is called E_T and the projection of momentum $\vec{p}$ in the $x - y$ plane is $\vec{p}_T$ with a magnitude of p_T , the latter being a particularly important variable in HEP. Instead of the polar angle θ , one often uses the pseudo-rapidity $\eta = -\log\left(\tan\frac{\theta}{2}\right)$ that maps the θ range of $[0, \pi]$ to $[-\infty, \infty]$. This transformation is motivated in ultra-relativistic Physics by the closeness of the pseudo-rapidity to the notion of rapidity. The two notions are in fact equal for massless particles. Note that there is a natural limit to the range of η to which the detector is sensitive since, at values of θ closed to 0 or π , the particle is still inside the beam pipe and cannot be detected. The geometrical (η) and dynamical (p_T) limitation of events observable due to the limits of the detector is called the *acceptance*. It depends on the object being studied, as different subdetectors may be involved in gathering information to identify it. A typical acceptance range for η is the range $[-3, 3]$.

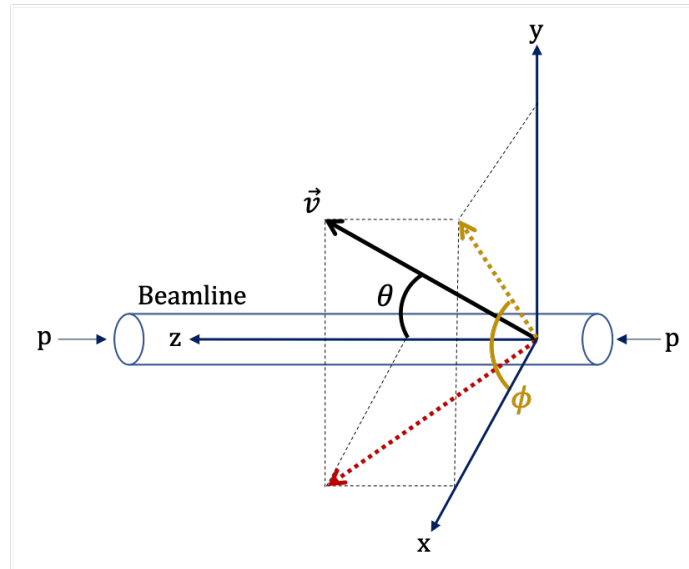


Figure 2.1.5 – The coordinate system used by the ATLAS experiment. The x -axis points towards the centre of the LHC and the z -axis is along the beamline (p symbolises a proton). For a particle of velocity $\vec{v}$, θ is the polar angle (between $\vec{v}$ and the z -axis) and ϕ the azimuthal angle (between the projection of $\vec{v}$ on the $x - y$ plane and the x -axis).

In operation, the enormous rate of collisions occurring in the centre of the detector (close to 40 MHz) causes a data transfer bottleneck. A set of filters must be employed to restrict data processing to events deemed of interest. These filters are called *triggers* and are implemented by a hierarchical setup of electronic devices, for the lower level, and numerical methods, for the higher one, to progressively reduce the rate of events. The raw data collected that passes all steps of the triggers can then be stored in giant databases. Several layers of processing are applied to reconstruct higher-level variables, such as the energy, the mass of a particle, the tracks, and, most importantly here, the identity or *tag* of the particle (e.g., gluon or quark), from low-level information, such as hits in the tracker and deposit in the calorimeter cells. Assigning a label to a particle is called *tagging*. A hierarchy of increasingly processed databases for different signal regions (regions with a predominance of a specific sort of events or particles, depending on the trigger setup used when collecting the data) is then accessible to physicists.

To analyse the data and extract the above relevant information, several techniques are employed. As the probabilistic nature of quantum physics forbids exact event-by-event analysis, one is forced to consider distributions of observables (for example the invariant mass, equal to the energy in the centre of mass of the collision). Given this stochastic aspect, Monte Carlo (MC) methods are the natural simulation tool to produce physical events as dictated by the relevant theory: this is called the *generated level* [19]. MC methods were in fact initially developed for this purpose, although in a more military context. The simulated events can then be passed through another layer of simulation to include the act of detection, bringing them to the *reconstructed level*. For example, when two bunches collide, the detector records a set of simultaneous and overlapping events, an effect known as *Pile-Up (PU)* that needs to be added to simulations. Many reconstructed events thus produced are agglomerated into distributions, to be compared to the measured ones. Consequently, theoretical predictions can be tested against the recorded information, to search for either agreement or deviation. This is a scientifically highly efficient albeit computationally intensive process that has favoured the common pulling of resources into a worldwide computing grid and, most notably, the Web, created at CERN in 1989 and made public in 1993.

2.2 Motivation and Open Questions

The previous sections introduced the landscape of the Physics experiment. The intense efforts devoted to these gigantic experiments come from the need to perform precise tests of the Standard Model and uncover traces of new Physics. For many reasons, the SM cannot be the final theory. For example, it already fails to include gravity. Joining this interaction with the other ones is challenging, given it is extremely weak at the quantum level and only becomes somewhat significant at larger scales, due to the positive aggregation of matter. The ambitious research programme of the Particle Physics community requires several characteristics [20]:

- Many events, to reduce the statistical incertitude.
- High energies, to increase the range of processes that can occur in a collision. Currently, the average centre of mass energy of proton-proton events in the LHC is of about 13 TeV.
- Fine granularity and precise methods, to improve the quality of the collected data, reconstruct more events, implement better selections, reduce the data transfer bottleneck, improve the accuracy by finer track reconstruction, and have better-aggregated information.

The first two points are regularly improved by upgrades of the components of the intricate experimental system of the CERN accelerator complex and the ATLAS detector. The final point can greatly benefit from machine learning. This has attracted a community-wide interest for these methods, with many applications already derived, particularly in classification and reconstruction (such as track fitting and pile-up removal) [21].

One such effort that is of paramount importance to the advancement of research in Particle Physics is the ability to identify and discriminate jets, and in particular those associated with quarks and gluons. These objects are indeed a by-product of many different physical processes. They leave very noticeable patterns in the different subdetectors, turning them into natural and evident clues to hunt specific signals. Better QCD and SM measurements as well as improved and new searches for exotic physics would be the prize of a significant

increase in efficiency [22]. Specifically related to the task of quark versus gluon jet discrimination, two promising analyses that would benefit greatly from an improved classifier are [23]:

1. Searches for mono-jet of dark matter in a gluon-dominated signal.
2. Searches for dijet resonances in a quark-dominated signal.

The history of machine learning in the context of High Energy Physics is already long, beginning with Multivariate Analysis (MVA) in the early 2000s [24] to recent deep learning applications. Most importantly, in the context of jet tagging, recently explored techniques include:

- Boosted Decision Tree (BDT): the reference method, most commonly used given its flexibility and ease of training for a relatively good performance when fed “highly-engineered” physical inputs [23, 25].
- Deep Neural Network (DNN): their performance is usually comparable to BDT’s, or marginally better [26]. It also requires engineered inputs, for competitive efficiency.
- Image analysis: the calorimeter surrounding the interaction point offers a natural 3D image of the jet energy deposits. This can and has been exploited in several Convolutional Neural Network (CNN) models. The performance obtained is however not state-of-the-art for quark versus gluon discrimination, even when combined with inside information from the tracker, but it does perform slightly better than basic BDT’s and DNN’s using typical jet information [27]. The significant advantage of these methods resides in the possibility to deploy them in triggers, as they only use low-level information (calorimeter hits).
- Junipr: a framework for unsupervised learning deploying a Recurrent Neural Network (RNN) around a *Leading Order (LO)* model of Physics to learn the probability distribution underlying the data [8]. Most observables in QFT are derived through perturbation theory to a certain degree or order, with the leading order being the first term. The Junipr framework offers the possibility of performing a likelihood ratio test to discriminate between different learnt distributions (from simulations)

corresponding to different types of jets. Furthermore, the probabilistic model learnt when scaffolded around a factorisation tree (a tree giving a successive order to decays leading to the final state particles observed) is interpretable. In 2019, the method was proven state-of-the-art for quark versus gluon jet discrimination on a set of simulations at the generated level [9]. Broadly, the method relies on RNN to learn from the succession of decays characteristic of a specific type of jet [28]. Given these qualities, the Junipr approach is the chosen paradigm for this project and is described thoroughly in section 3.2.3.

- Particle cloud: an adaption of point cloud considering a jet as an unordered list of final state particles. When combined with Dynamic Graph CNN to perform complex convolution operations on the graph of final state particles, the method can achieve good performance for jet tagging problems [29]. This approach exploits recent progresses in Graph Neural Network (GNN) [30].
- Energy Flow Network: this method produces a set of highly-engineered features called *energy flow polynomials*. They constitute a complete set of jet substructure observables, forming a discrete basis [31]. This information can be passed to any of the previously mentioned methods (linear classifier, BDT, DNN, CNN, ...) to improve its performance in the tagging and discrimination tasks [32, 33].
- Weakly supervised learning: several papers have suggested that this learning paradigm could prove “more physical” [26]. Indeed, the assignment of a jet as emanating from a quark or a gluon parton is an inherently phenomenological one, as it does not correspond to any real physical process. The quantum world is far more complicated than the simple image this may suggest, as the contents of a proton themselves are a probabilistic mix and jets emanate from interactions within these distributions (the previously introduced parton distribution functions). Jets are assigned such a label when their orientation corresponds to that of a *mother* parton momentum. What is typically called a quark or gluon jet is a specific object possessing its own properties and emanating from distinct scenarios. Therefore, rather than learning with exact labels, it has been proposed to pursue discrimination between quark- or gluon-rich distributions collected directly from the data, made of impure samples [34, 35, 36, 37].

These various methods have been applied with different levels of success to the task of jet tagging using information from physical simulations. However, most of them are not considered in the context of a specific measuring device, as they are restricted to *generated* level simulations. Measuring distorts to a certain degree (depending on the measured variables, the state of the detector at the time, ...) the underlying distribution to bring it to the *reconstructed* level. It is an unavoidable consequence of interacting with the Universe at our scale. The extra-layer of simulation required to mimic the act of detection, introducing variance and blurring the descriptive power of observables, has to be accounted for when considering the level of a certain experiment, confronted to a specific way of collecting data. The aim of this project, described in the next section, is to confront a selection of these methods to the reality of the ATLAS experiment at CERN.

2.3 Objectives and Organisation

From the previous section, one can conclude that there are three main ways of representing a jet:

1. As a physical object, with observables describing it, such as its width, and number of constituent particles. This leads to common methods, such as DNN and BDT.
2. As an image: whether solely from the calorimeter or combined with track and other information, offering a natural ground for CNN.
3. As a succession of branchings, whether ordered (factorisation tree) or unordered (particle cloud). This gives a relational point-of-view adapted to RNN and GNN.

This project implements DNN and BDT models for benchmark and the Junipr approach from the paradigm of recurrent learning to the specific case of the ATLAS experiment. Two main research questions are explored with this subset of models:

1. Is it possible to efficiently discriminate quark and gluon jets based on the low-level information delivered by the calorimeters of the ATLAS experiment? For this purpose, Junipr models are adapted to work with topocluster momenta. Junipr is chosen as it obtains state-of-the-art result on generated level simulations.

2. Is a Junipr approach competitive compared to commonly used methods when working at the reconstructed level? For this purpose, DNN and BDT models on high-level variable of reconstructed simulations are implemented. These models were chosen as they are quite common in the field and exploit engineered variables. They thus provide a natural benchmark to analyse the results of Junipr.

The main novel contribution of this work is to implement the Junipr model for fully *reconstructed* simulations, which are intrinsically noisier than generated-level data. Indeed, reconstructed-level data suffers from the limited precision of the detecting device, pile-up, and competing processes (both reducible, if they possess different final states, and irreducible, if they do not). The second important contribution is to develop an efficient quark versus gluon jet classifier restricting to *low-level* information from the detector. This final constraint, vastly complicating the task given the lower information content, is motivated by the following points:

- This kind of information is that directly gathered by the detector and, therefore, the first one accessible: exploiting it efficiently is vital in the context of triggering. A precise algorithm, trained offline, could quickly perform an intelligent online selection of interesting events.
- Low-level information has a geometrical nature (such as the energy deposit and track hits) due to the extent and localisation of detector cells and, to a certain degree, a recurrent one (track combination and vertices, points where new tracks emanate, radiate in a tree structure from their initial point, the PV). Many advanced Machine Learning (ML) techniques from the previous section can exploit these patterns.

This project is an interdisciplinary effort between the Computer Science and Particle Physics departments of the University of Oxford. The objective of this work is to serve as a concrete basis for future physics analyses by assessing the potential of new specific computational tools based on advanced ML. Interesting steps such as interpreting the distributions obtained (except when assessing the quality of results) or searching for specific signals involving quark and/or gluon jets are not included in the present dissertation and are left for future research. Given the COVID-19 pandemic in 2020, the entire work was

carried out remotely and contacts with supervisors were restricted to mostly weekly Teams meetings (sometimes at a higher frequency with Aaron O'Neill during the stages concerned with collecting the data). This has not caused any significant issues. Indeed, the CERN community, being spread all around the globe, is used to deal with such communication restrictions and have thus developed an organisational structure appropriate for remote access.

Summary of the Chapter

This chapter introduces the physical context of the project. Quarks and gluons are elementary particles from the Standard Model of Particle Physics. Having a colour charge, they are sensitive to the effects described by QCD and hadronise to form colour-less objects. This process generates large structures of successive decays called *jets* and leaves a remarkable pattern in the detector of the ATLAS experiment, based on the LHC at CERN. Jets provide a useful tool to search for and isolate specific signals in the collected data to compare to theoretical predictions. A review of current methods introduces different machine learning-based techniques to discriminate quark from gluon jets. Among them, the Junipr framework is the most promising option, given it leads to state-of-the-art performance at the generated level and is interpretable when founded upon a physical factorisation model. This project studies an adaptation of the Junipr approach to use low-level calorimeter information to improve the accuracy at the reconstructed level of the quark versus gluon jet discrimination task. BDT and DNN models serve as benchmarks, given their popularity in the field and reliance on high-level variables.

The pipeline of the project is explored in this chapter: from data collection, processing, and analysis, to the implementation of the models. As in any data science project, there is a strong interaction between the dataset produced and the quality of the analysis carried out on the trained models. Many iterations and adjustments were made to take these subtleties into account; this is a recurring theme throughout this chapter. A diagrammatic representation of the softwares implementing the methodology is presented in figure [A.1.1](#).

3.1 Data

Particle Physics is a potent field to implement machine learning methods. The data collected in its endeavours is well understood theoretically, abundant, and reliable. It can easily be simulated to both the generated level and the reconstructed level. Monte Carlo methods are most commonly chosen to perform the variety of tasks required to simulate data production in the conditions of a specific experiment, though, recently, deep generative models have also been considered (e.g., variational autoencoders and adversarial networks [\[38\]](#)). At the generated level, a large sample of individual random events is produced in a fashion prescribed by theoretical concerns. The interactions of the various final state particles with the detector are then simulated to bring the generated events

to the reconstructed level. The objective of this second step is for the reconstructed simulations to be comparable to the data collected. Note that in HEP the term *data* is usually reserved for collected data (from real events in the detector), while simulated data is usually referred to as *simulation* or *MC*. A typical proton-proton interaction is simulated with the following steps [39]:

- A hard scattering process: partons of each proton collide. The probability of the selected momenta of these constituents are returned by the parton distribution functions. The adjective *hard* is used to describe the part of the event with the largest momentum transfer, while *soft* typically refers to further processes which generally have smaller such transfers.
- Around the hard scattering event, parton showering in the initial and final states can occur through radiation of quarks and gluons.
- The underlying events: the parts of the protons not concerned with the hard scattering can interact or decay.
- Hadronisation: low energy quarks and gluons (below the typical 1 GeV) cannot be assumed to be free and they have to recombine into colourless hadrons.
- Particles with a short lifetime, such as taus, have their decay simulated.
- Pile-up is added to reproduce the multiple interactions per bunch crossing in the output data. Mainly, soft inelastic collisions are added to the interesting hard scattering event. This procedure is quite difficult to calibrate. The problem is corrected by reweighting the MC datasets to match the PU distributions observed in the data.

Various MC generators are available, such as Pythia [40] MadGraph [41] and Powheg [42]. Pythia has the advantage of being a general-purpose program describing both the hard process and the parton showering. It is however limited to LO computation. The events are then submitted to a dedicated toolkit called GEANT4 [39, 43] to simulate the passage of particles through the different ATLAS subdetectors and bring the simulation to the reconstructed level. The AnalysisBase library is a C++ dedicated software for the ATLAS experiment that contains knowledge on the detector and methods for physics analyses

[44]. It also offers the possibility of submitting jobs to the GRID, a shared computing network, and of efficiently accessing files produced by the ATLAS collaboration, thanks to the RUCIO library [45].

3.1.1 Data Collection

To carry out the analyses pursued in this project, different simulated datasets were produced, each generating two sets: one for quark jets and another one for gluon jets. The quark jets are produced from $t\bar{t}$ events, the decay of a pair of particles made of a top quark t and an antitop quark $\bar{t}$. Indeed, most of the jets produced in this particular decay are known to be quark-like. There is however also a portion of gluon jets generated by this process, such as those coming from initial state radiation, so the set is not pure in quarks. The gluon jet samples come from the dijet process. These are events where two jets are produced from the collision, with a higher proportion of gluon jets than quark jets expected. Two different types of ATLAS-produced simulations based on Pythia are used throughout this project. They are described in the following two paragraphs.

MultiJet Data: the `qgTaggingSystematics` from `xAOD MultiJetsNTuples20200506` (MC simulations based on the 2016 state of the detector), gathered by Aaron O’Neill of the Physics Department. It collects quark and gluon jets from a larger variety of processes than previously described. These include the dijet and $t\bar{t}$, the two most important, but also ZZ and WW decaying with some quarks in the final state, for example. The larger set of processes considered makes this dataset an “average” representation of quark and gluon jets encountered in the detector. This dataset only contains high-level information (engineered variables) and is thus not suitable for Junipr.

In Physics, each process has a certain probability of occurring, depending on the energy of the collision and other observables. This probability is represented by the *cross-section*, in unit of area (typically, the barn $b = 10^{-28} \text{ m}^2$). The idea comes from a geometrical analogy: the probability that two particles collide is proportional to the cross-sectional area of each. The notion is easily extended to talk of the probability of specific events to happen, not just generic collisions. The samples in this dataset are not weighted by cross-section,

meaning that the physical probability distribution of having events from each process is not respected (for example, the $t\bar{t}$ process is far more likely than the ZZ , yet the number of ZZ events is not scaled down to the right fraction). There is some interest in avoiding such weighting: it forces one to extract general quark versus gluon jet information and not to rely on the underlying probability distribution of the dominating processes. This equates to making the network blind to the processes, with the only usable information contained in the jet dynamics itself. The data can however easily be weighted appropriately to match the expected cross-section by importance sampling (this was implemented in the project but not used).

DAOD_JetM6 Data: these samples were obtained from the ATLAS central database. They are taken in a raw format called *DAOD_JetM6* where low-level information is present (jets reconstructed from calorimeter information) but that also contains higher-level jet variables. Finding a dataset possessing both aspects is not trivial, as low-level information is usually filtered out when higher-level variables are produced since the former requires a lot of memory to store. The set thus produced is suitable to compare the personal Junipr implementation to the benchmarking models, the latter working only on engineered-variables. For this purpose, 102 GB of dijet and 84 GB of $t\bar{t}$ were downloaded to the Oxford Particle Physics computing server using RUCIO. The set of dijet samples is larger, as these simulations are sliced in p_T and more files are thus required to cover the whole p_T range considered compared to the $t\bar{t}$ (that is not sliced in p_T).

3.1.2 Data Processing

The raw data collected has to be processed into a usable format for the ML methods considered here. The first step of the processing for the MultiJet data had already been carried out and the data was stored as ROOT files on the Particle Physics server. ROOT is a C++ object-oriented data analysis framework developed by CERN to easily store and access histograms and data organised into tree structures [46].

The DAOD_JetM6 sample however required a first step of processing. An EventLoop-based algorithm, applying the Athena program of the AnalysisBase library [44], was im-

plemented with the required tools of the ATLAS software distribution to process the raw data and extract relevant information on jets and their constituents. Writing such an algorithm is a complex task that requires the fine-tuning of many elements, for data access and quality, and interconnecting several tools of the ATLAS software package (it typically takes 2-3 months for a Physics DPhil student to familiarise oneself with it). Thanks to the help of Aaron O'Neill, about three weeks were required to have a complete and running version. The program also includes a working GRID submission protocol to perform the job on the CERN computing network, and only download the processed files locally. The GRID protocol was eventually not used, because the ATLAS collaboration was at the time in the process of modifying the DAOD_JetM6 distributions, causing an unfixable error when trying to access the files through the GRID. They were instead downloaded locally to the Physics server, as the estimated required storage volume to have large enough final datasets after processing was of a reasonable size (below 200 GB). The algorithm receives a list of targeted low-level ROOT files, processes each event in them (applying a set of selection criteria and reconstructing variables based on SUSY tools, a set of methods coded by the ATLAS subgroup working on Supersymmetry searches), and gathers the transformed data into ROOT trees.

The next stage of processing depends on the model targeted. In all cases, the quark and gluon samples are purified in order to maintain comparability to the results of the Junipr papers.

- The datasets for the BDT's and DNN's are Pandas dataframes, having a matched number of quark and gluon jets and specific global jet variables, stored in an HDF5 file for storage and access efficiency. These dataframes were produced using two different processing routines (for MultiJet and DAOD_JetM6) loading the ROOT trees using UpRoot [47], a python package to work on ROOT files. Some cuts have been added to the pipeline to further restrict the data, such as a p_T cut removing jets with less than 20 GeV. Tables 3.1.3 and 3.1.4 list the cuts applied for the MultiJet and DAOD_JetM6 samples, respectively.
- For the Junipr model, the final state constituents of the jets with their momenta and

the truth (meaning as reconstructed by the ATLAS tools on the MC simulations) label are collected. Since the project focuses on low-level information, constituents, in this case, are topoclusters (group of calorimeter cells) and not actual particles. The momentum of a calorimeter cell is derived by a short calculation: knowing the energy deposit and the location of the cell, a four-momentum $\mathbf{p} = (E, p_x, p_y, p_z)$ is constructed by forcing the entire energy to come from momentum $\vec{p}$ and setting the mass to zero [48]. The momentum of a topocluster is then simply obtained by summing the momenta of its cells. This is the base information used to cluster the constituents of the jet into a tree and derive the inputs of the Junipr network, as described next.

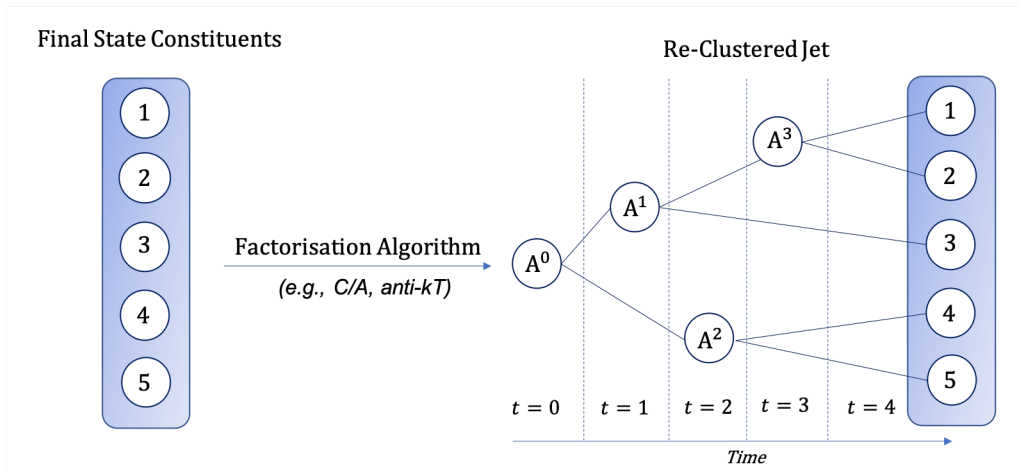


Figure 3.1.1 – Example of a tree produced by jet clustering. Circles with a number are final state constituents. They are sequentially combined into intermediary states described with A^t , where t is the timestep of a decay. A^0 is the seed-momentum (the first particle to decay): finding its label (quark or gluon?) is the objective of the game.

A dedicated program is tasked with processing the Junipr raw material into a set of suitable *json* datasets. Each resulting sample is stored as a dictionary of several keys describing the data. The process relies on the PyJet library [49], wrapping the C++ FastJet package [50] for Python, to implement different clustering algorithms. Indeed, the final state constituents of the jets have to be reclustered following a specified factorisation algorithm, combining pairs of constituents until a single global element is obtained, referred to as the *seed* or *root* (its momentum is called the *seed-momentum*). The clustering history builds a tree-like data structure, as presented in figure 3.1.1, that is the input to Junipr.

The choice of clustering algorithm is not restricted by Junipr since, as explored in the original paper [8] and in section 3.2.3.2, the model is capable of performing on any clustering scheme that adopts the sequential recombination approach (inverting the successive parton branchings to recombine constituents into a single one). In HEP, a family of methods, referred to as *generalised- k_T algorithms*, relies on the following iterative procedure [13, 50].

Generalised- k_T algorithms

1. List the available particles/constituents at the given timestep t , where the initial list is made of the n final state elements. Note that, in this algorithm, time flows in the opposite direction than in figure 3.1.1 (here, from the final state to the root).
 2. Build two distinct sets of distances for the elements of the list.
 - An inter-particle distance: $d_{ij} = \min(p_{t,i}^{2q}, p_{t,j}^{2q}) \Delta R_{i,j}^2$, where q is a free parameter (choosing its value restricts the family to a specific model), $p_{t,i}$ is the momentum of the i^{th} element of the list at step t , and $\Delta R_{i,j}^2$ is the geometric (angular) distance in the $\eta - \phi$ plane, $\Delta R_{i,j}^2 = (\eta_i - \eta_j)^2 + (\phi_i - \phi_j)^2$.
 - A beam distance: $d_{iB} = p_{t,i}^{2q} \Delta R$, where ΔR is another parameter called the jet radius (a typical measure of the jet width).
 3. Iterate over all possible combinations, to find the smallest distance:
 - If the smallest distance is a d_{ij} , the two particles i and j are clustered together at this timestep. They are removed from the list and a new element k is added with momenta $\vec{p}_{t+1,k} = \vec{p}_{t,i} + \vec{p}_{t,j}$ to the next list.
 - If the smallest distance is a d_{iB} , then i is returned as being a jet and is removed from the list.
 4. If the list is not empty, go back to step 1, with t incremented for all remaining particles/constituents.
-

- The anti- k_T algorithm: the value of q in the generalised- k_T algorithm is set to -1. In LHC Physics, jets are almost always reconstructed with this algorithm, as it favours hard particles to cluster first (the minimum distance will likely correspond to a hard particle, given the value $q = -1$ introduces $1/p_{t,i}^2$ factors in the distance).
- The Cambridge-Aachen (C-A) algorithm: the value of q in the generalised- k_T algorithm is set to 0. The distance is then purely geometrical, with no momentum magnitude.

Figure 1 consists of two plots, (a) and (b), showing the evolution of the jet structure as a function of the recombination scale Q . The color bar at the bottom of each plot indicates the scale Q from 1 GeV (red) to 100 GeV (blue). The nodes in the plots are labeled with their energy values in GeV.

(a) Jet reclustered with anti- k_T .

(b) Jet reclustered with C-A.

The anti- k_T algorithm grows the jet by successively adding soft radiations to the hard centre, making it more resilient against the contamination induced by soft background radiations. On the contrary, the C-A algorithm structures the jet into several branches depending on geometrical proximity. The first algorithm sequentially builds the jet final structure, while the C-A method grows the final structure from parallel branches of decay

[8]. Note that the intermediate states of both methods and the order of decays are different. Section 3.2.3.2, dedicated to the ML implementation of the Junipr model, explores the important consequences of these different patterns.

The clustering steps in Junipr offers an interesting experiment to compare the factorisation algorithms. All the raw data processed for Junipr jets comes from EMTopoJets reclustered on the ATLAS side by using the anti- k_T algorithm with an angular radius of 0.4. However, for reasons uncovered in section 3.2.3, the Junipr jet processing reclusters the final constituents of the EMTopoJets, which are calorimeter topoclusters and not particles, with the C-A algorithm and a radius of 0.5. The larger radius is used as there is already a good level of confidence that the final constituents match a single jet. When using the methods directly on a list of particles observed in the detector, a smaller radius is used to reject contamination coming from competing processes, such as the PU. Here, the constituents are already coming from a given jet and the large value of the radius serves as a “wider net” to include all flagged elements. However, some jets are reconstructed by C-A as two different objects, in opposition to the result of anti- k_T . This is extremely rare given the larger radius chosen for C-A, occuring to only ~ 220 jets in a sample of 3,600,000 jets, or about 0.006 % of the whole set. Two extreme examples of such differently reconstructed jets are presented in figure 3.1.3 by showing the $\eta - \phi$ maps, where constituents of different C-A reconstructed jets are indicated by different markers and colour reveals their energy (the black marker corresponds to the C-A clustered jets $\eta - \phi$ values).

A histogram of the size of the two jets reconstructed by C-A from single anti- k_T jets is presented in figure 3.1.4. Clearly, most of these exceptions arise from a few constituents living on the border of the jet and of low energy. As suggested in its definition, the C-A algorithm tends to exclude such radiations, given it is geometrically centred. No special precautions were taken against these differently reconstructed jets: they were simply removed from the data (given their rarity, this is not expected to have any significant impact). However, in regular operation, when a smaller radius (below 0.4, instead of 0.5) has to be chosen for the clustering method to avoid contamination from PU, far more jets are expected to undergo this process and its impact would have to be carefully considered.

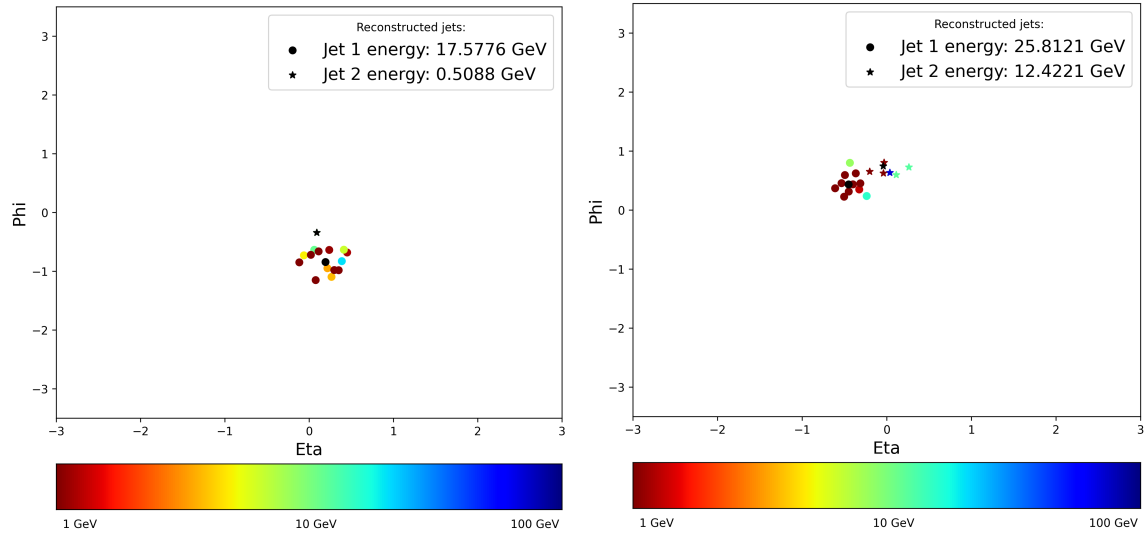


Figure 3.1.3 – Two single anti- k_T (radius = 0.4) $EMTopoJets$, each reconstructed as two jets by C-A (radius = 0.5). Markers indicate to which of the C-A jets the constituent is associated, with colour indicating its energy. The black markers are the reconstructed C-A jets.

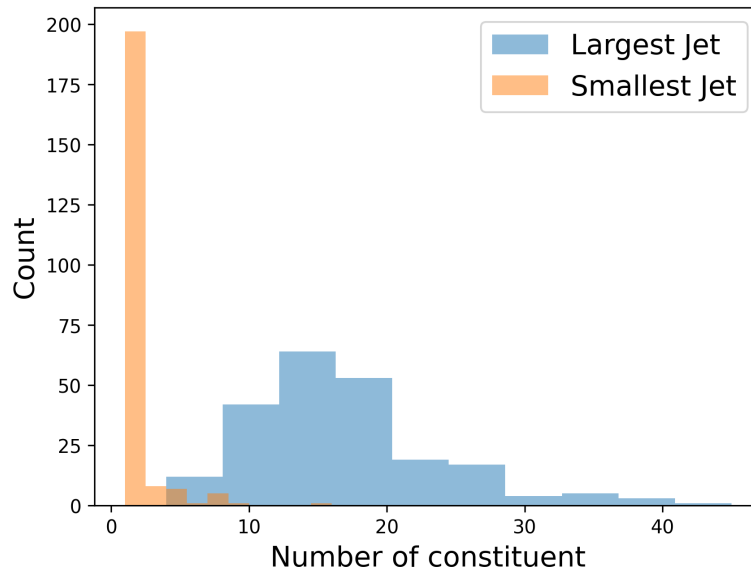


Figure 3.1.4 – Histogram of the number of constituents in the two jets reconstructed by C-A (radius = 0.5) from a single anti- k_T $EMTopoJet$ (radius = 0.4), collecting all such events encountered when processing set 4 (defined in table 3.1.2).

Junipr jets are processed from root (seed-momentum) to leaves (final state constituents). Data on the tree is stored into different entries of a dictionary. Table 3.1.1 lists the different variables accumulated, separating them into three categories: *core variables* are those required as input by Junipr, *visualisation variables* are necessary to construct the jet tree using physical information (as displayed in figure 3.1.2), and *auxiliary variables* offer information useful for debugging. All four-momenta listed in the tables are represented as (E, θ, ϕ, m) , respectively the energy, polar angle, azimuthal angle, and the mass. Angles are taken relative to the seed-momentum (so that the seed-momentum always has $\theta = 0 = \phi$ and the data is centralised by a classical rotation). The branching information is encoded into four variables as $(z, \theta, \phi, \delta)$, where:

- z is the fraction of energy of the soft daughter (the one with lowest p_T) to its mother,
- θ is the angle between the soft daughter and the mother,
- ϕ is the angle of the plane of decay relative to the plane constructed by the direction of the mother and the y -axis of the detector,
- δ is the angle between the hard daughter and the mother.

Core Variables

Label	The truth label of the jet: 0 for gluon, 1 for quark (any flavour), and -1 for others.
N-branchings	The number of branchings (of nodes) in the jet (the tree).
Seed-momentum	The 4-momentum of the seed (initial mother).
Mother-id-energy	At each timestep (node), the index of the mother to decay in a sorted energy list (decreasing order) of available elements at that step (t candidates at timestep t).
Mother-momenta	The 4-momenta of the mothers to decay.
Daughter-momenta	The 4-momenta of the two daughters produced in each decay.
Branching	The encoded information on the branchings mother $\rightarrow$ daughters.

Visualisation Variables

CSJets	A complete list of all 4-momenta in the jet (intermediary and final states).
CS-ID-mothers	Index of the successive mothers in the CSJets list.
CS-ID-daughters	Similar information for daughters.

Auxiliary Variables

Multiplicity	The number of final state constituents.
CS-ID-intermediate-states-ind	Lists each intermediate states by listing the indices (referencing the CSJets list) of the elements available at each timestep.

Table 3.1.1 – The information collected in the json files when processing the analysed ROOT tree into Junipr jets.

Dataset	Data Origin	Use Athena	Targeted Model	Sample Size	General Remarks and Additional Cuts
Set 1	MultiJet $t\bar{t}$	False	BDT, DNN	~ 0.6 M	Already available as a CSV. Used to validate the BDT and DNN implementations. Additional cuts from table 3.1.3.
Set 2	MultiJet	False	BDT, DNN	~ 140 M	Required a processing step to bring 2.2 TB of ROOT data to a 17GB HDF5 file (by <i>UpRootTransformer</i>). Additional cuts from table 3.1.3.
Set 3	MultiJet Completed	False	BDT, DNN	~ 150 M	Processed with <i>UpRootTransformer</i> . Additional cuts from table 3.1.3. A completed version of Set 2 (fixed a problem in data transfer between the grid and the Particle Physics server).
Set 4	DAOD JetM6	True	Junipr	~ 3.6 M	JSON dataset processed with <i>GranularUpRootTransformer</i> . Jets reconstructed using C-A (radius of 0.5). Jets from the DAOD table are EMTopoJets object, reconstructed using anti- k_T with a radius of 0.4. Additional cuts from table 3.1.4. It removes final state constituents with $E < 1$ GeV. The quark and gluon datasets are sampled so that they have the same number of events and energy distribution.
Set 5 and 5b	DAOD JetM6	True	Junipr	~ 3.6 M	Similar to the above one, except that it removes all final state constituents with an energy below 0.5 GeV instead of 1 GeV. Implements the cuts from table 3.1.4. A special set called 5b is processed in the same way but made of Truth jet constituents instead of the EMTopoJets.
Set 6	DAOD JetM6	True	BDT, DNN	~ 2.8 M	Part of the set 4 data collected into a HDF5 jet variable table. Additional cuts from table 3.1.4.

Table 3.1.2 – The datasets collected and produced in the project.

Cut	Explanation
Track Width $p_T500 < 0$ Track Width $p_T1000 < 0$	Removes jets that have an unphysical (negative) width from tracks with p_T above 500 or 1000 GeV.
numPrimaryVertices = 0	Removes jets where no PV exists (a central point from which most tracks should emerge).
PVnumTrk < 2	Removes jets with less than two tracks associated to the primary vertex.
isTruthQuark < 0	Removes jets with a truth label that does not match either quark (1) or gluon (0).
hasBadMuon = 1 hasCosmic = 1	Removes jets where a muon is distorting the data or where a cosmic particle went through the detector.
$p_T < 20$	Removes jets with a p_T below 20 GeV (part of the acceptance cuts).
$ \eta > 2.5$	Removes jets falling out of the geometrical acceptance (part of the acceptance cuts).

Table 3.1.3 – Additional cuts for sets 1, 2, and 3.

Cut	Explanation
E of constituent < 1 or 0.5 GeV	Sets 4 and 6 use the first limit while sets 5 and 5b use the second one.
isTruthQuark < 0	Removes jets with a truth label that does not match either quark (1) or gluon (0).
isNotPVJet = 1	Removes jets where the trigger isNotPVJet is on. This trigger is added to the Athena algorithm and set to 1 when a jet is not coming from the PV. Such bad jets typically come from the PU (this point is further detailed in Appendix section B.1).
$p_T < 20$	Removes jets with a p_T below 20 GeV (part of the acceptance cuts).
Number of jet constituent < 5	Removes jets with less than 5 constituents, as these can in fact correspond to miss-classified taus or muons (this point is further detailed in Appendix section B.1).

Table 3.1.4 – Additional cuts for sets 4, 5, 5b, and 6. Note the absence of high-level variables cuts (for example, that on the track width p_T500) and of the geometric acceptance cut ($|\eta| > 2.5$ in table 3.1.3). The latter is unnecessary given *EMTopoJets* are reconstructed from calorimeter and tracker information, hence events falling out of the acceptance would not trigger any signal (though parts of the jet may fall out).

3.1.3 Analysing the Data

Combining the steps presented in the two last sections, six different datasets were collected, as listed in table 3.1.2. Note that Appendix A offers a global view of the codebase implemented. Modifications and tweaks had to be successively incorporated in the data collection and processing to reach optimal datasets from a physical point of view. This section describes some of the elements uncovered in this process.

To manage such large samples and expensive operations in an optimised way, the processing jobs were carried out from the Particle Physics department server (specifically, pplxint machines 10 and 11 accessed through ssh) and submitted to their HTCondor system. HTCondor is a computing software framework that offers a high degree of parallelisation on a cluster of computers [51]. It is used in the Physics department to schedule jobs for the batch farm [52].

The objective of the work is to develop a generic quark and a gluon jet tagger capable of discriminating between the two with low-level data. It is not for a model to learn the underlying distributions of the physical processes generating those jets. The models must be able to infer the correct label for quark and gluon jets isolated from any underlying distribution, working as a black-box fed the specific jet information. This is an important point: since ML methods are trained on aggregated samples, they can easily extract meta-information (*inter*-sample) to statistically achieve the task demanded, instead of extracting information from the sample data itself (*intra*-sample). The distributions of events in the quark and gluon samples therefore have to agree as much as possible, within physical constraints that must be preserved. By investigating some diagnostic plots collected when processing the data, the gluon samples from sets 4 to 6 were observed to violate this requirement. Indeed, as figure 3.1.5 suggests, the underlying distribution of the gluon sample (left) does not match the quark sample (right). Note in particular how the first and third bins disagree, as well as the longer right tail of the gluon energy distribution. Comparison is easier by opting for a logarithmic scale for the ordinate axis, as displayed in figure 3.1.6.

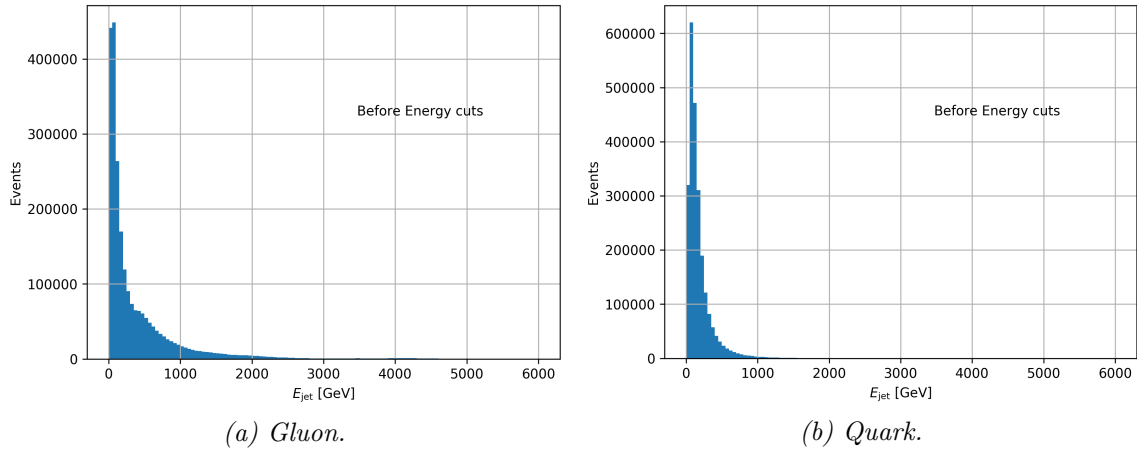


Figure 3.1.5 – The energy distributions of quark and gluon jets after all cuts on sets 4 and 6.

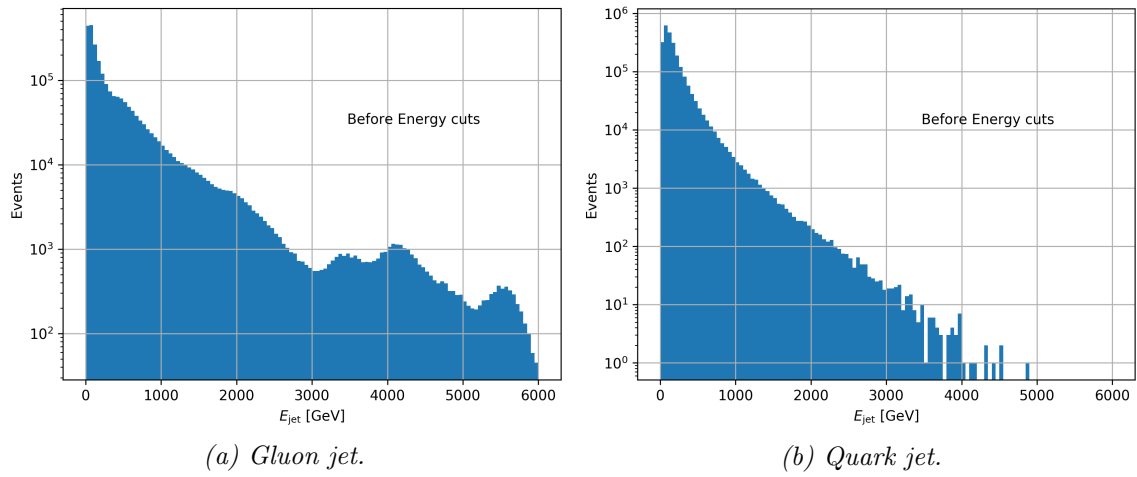


Figure 3.1.6 – The energy distributions of quark and gluon jet after all cuts on sets 4 and 6, semi-log scale.

The strange succession of peaks observed in the gluon sample comes from the process used to generate these jets: the dijet channel. The different peaks in the energy distribution come from the raw data used. Indeed, to ease the effort of simulating the process across all energies, a given number of events is produced in each slice of energy. The number of events per slice is not constrained by physical limitation and a mismatch is introduced when summing the slices. To correct this spurious property, the quark and gluon jet samples are resampled to have the exact same energy distribution. The resulting distribution is shown in figure 3.1.7. The issue having been expected, a larger sample of gluon jets was collected so that it could be trimmed down to match the quark one.

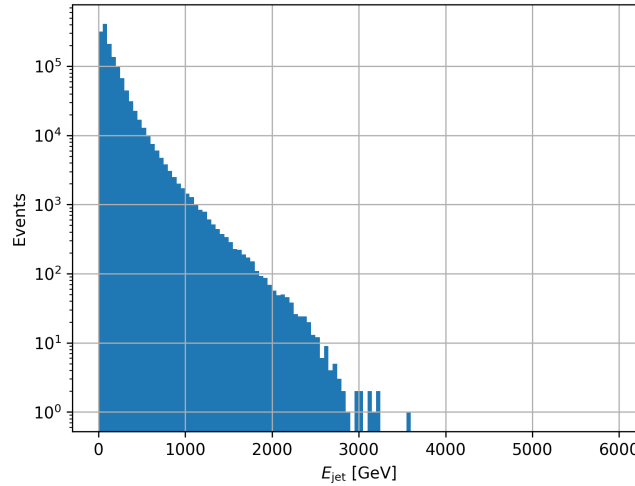


Figure 3.1.7 – The energy distributions of quark and gluon jet after all cuts and matching of sets 4 and 6, semi-log scale.

While this resampling step removes some of the spurious differences between the quark and gluon jet energy distributions, it does not degenerate the two samples into the exact same global distributions. Indeed figure 3.1.8 suggests that the p_T distributions are still significantly different, as well as figure 3.1.9 showing that the distributions of the size of the jets in terms of their number of constituents are different for quark and gluon jets. This conclusion is further reinforced in figure 3.1.10 that unveils a subtle difference between the two objects: gluon jets are on average wider, correlating with the fact they have more constituents. Width in this context is the angular measure of the girth of a jet.

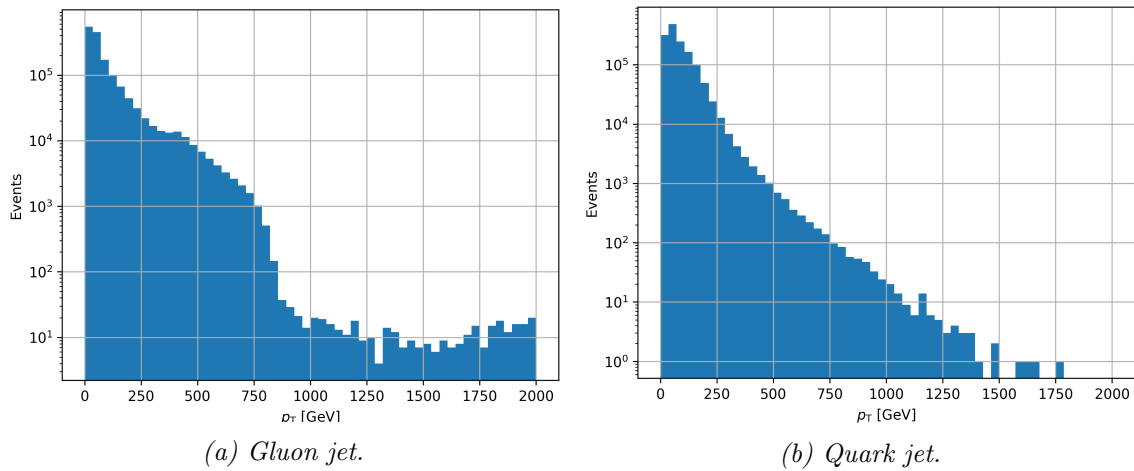


Figure 3.1.8 – The p_T distributions of quark and gluon jets in sets 4 and 6, semi-log scale.

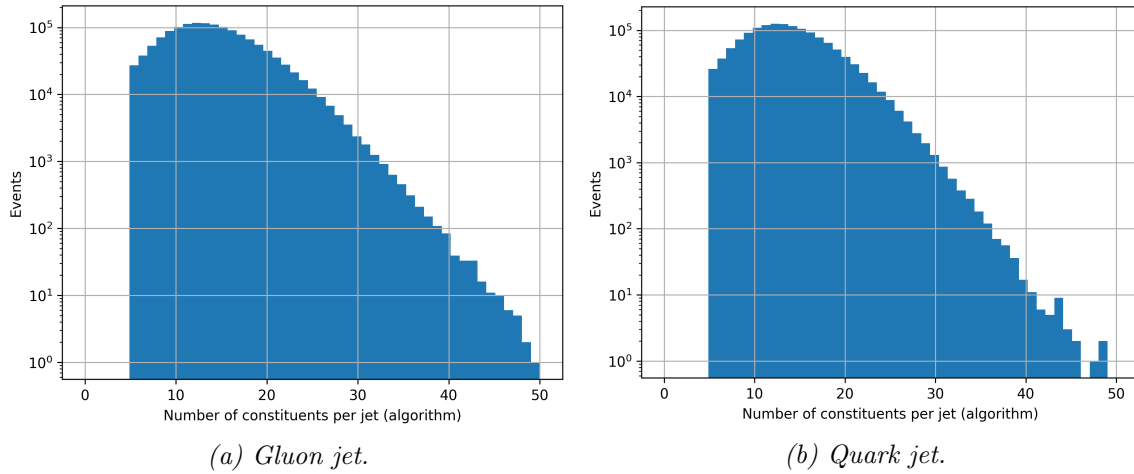


Figure 3.1.9 – The number of constituents in the jets of sets 4 and 6, semi-log scale. Note the effect of the cut removing jets with less than 5 constituents. Figure B.1.1 in Appendix B justifies why the cut removing jets with less than five constituents was adopted.

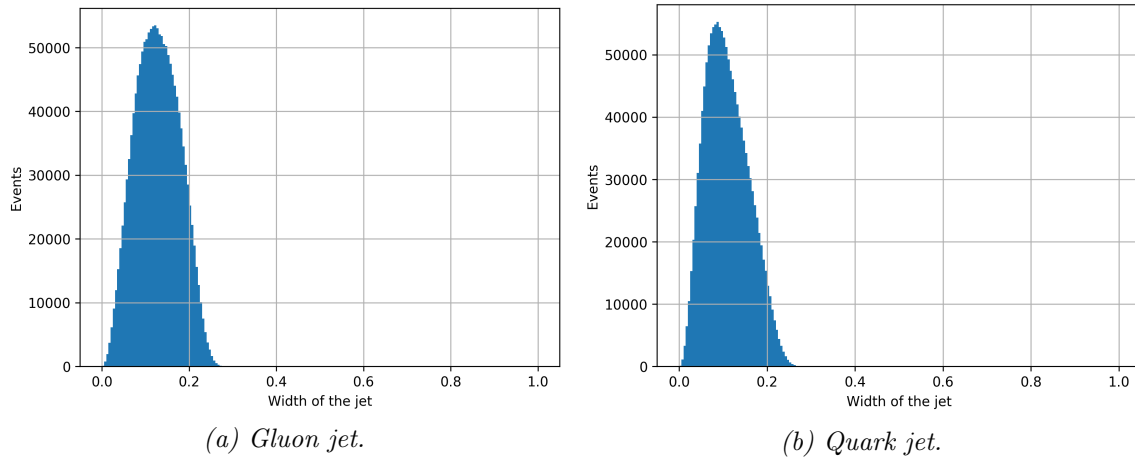


Figure 3.1.10 – The width of the jets in sets 4 and 6, as reconstructed by the ATLAS tools used in the Athena algorithm. The gluon distribution (left) is more tilted forward than the quark one (right).

Such differences have to be expected, as they are internal properties of quark and gluon jets. Indeed, gluon jets tend to be wider and have more constituents due to the properties of QCD (they carry “more charges of colour”). This has two major consequences [23]:

- Radiating a gluon from a hard gluon is favoured by a ratio of colour factors of $\frac{C_A}{C_F} = \frac{9}{4}$ compared to radiating a gluon from a hard quark. This implies that in the history of decays initiated by a hard gluon, more gluons will be present which in turns gives wider jets with more constituents.
- A second technical reason is a difference in the soft limits (low p_T) of the splitting functions $P_{gg}(z)$ and $P_{qq}(z)$. A splitting function $P_{xy}(z)$ gives the probability that a

parton of type x is emitted off a parton of type y with an energy fraction z . They are directly due to the mechanisms described by QCD. This difference is prone to make constituents of a quark jet carry more p_T than those of a gluon jet. Hence, a quark jet tends to be made of fewer constituents since its momentum is on average shared among fewer elements (thus each constituents receives on average a larger share of momentum than gluon jet constituents).

The rest of the jet variable distributions for sets 4 and 6, used in both the BDT and DNN, are presented in Appendix B.1. Figure 3.1.11 presents two constituent-level variables: the mass (left) and the ϕ distributions of constituents. It confirms that a constituent in this case is a topocluster. Indeed, note the absence of mass (all in the null bin) and the particular structure of the ϕ distribution. The latter is due to the geometry of the detector. The calorimeter cells in the barrel and endcaps of the detector are organised along radius of given ϕ . This leads to strong periodical peaks in the distribution, as many cells are aligned behind the first layer of cells thereby forming columns, with lesser numbers of cells interleaved between them in the outside layers to close the geometry of the calorimeter. The peaks, therefore, correspond to the ϕ coordinate of the centre of topoclusters in the first layer of the calorimeter.

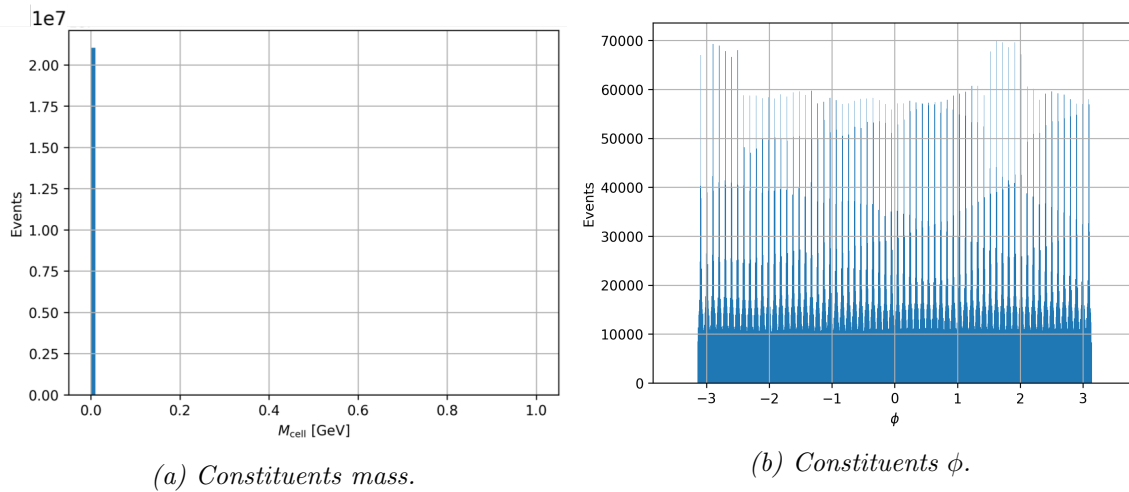


Figure 3.1.11 – The mass (left) and ϕ distributions for constituents of the EMTopoJets used for set 4 (gluon sample), after all cuts.

3.2 Machine Learning Models

This section explores the particular ML architecture implemented in the project. While the BDT and DNN models are fairly standard (one of the reason motivating their use as a benchmark), the Junipr model exploits many particular aspects of the jet structure derived from Physics. The BDT model is implemented as a benchmark because it represents the best out-of-the-box method commonly used in Particle Physics. The DNN model is also chosen for this reason and for its flexibility in training and known efficiency when using high-level variables information, as discussed in section 2.2.

3.2.1 Boosted Decision Trees

Decision trees use as input a list of features to recursively split the data, in order to maximise information or minimise entropy [53]. Each splitting, based on a single variable, is associated with a node and organised into a larger structure forming a tree. The hierarchy of the nodes in the tree and the variables used are decided by selecting the variables having the largest impact to reduce the entropy. The method is said to be *boosted* when several such weak learners are combined to form a strong classifier, based on a forest of decision trees [54]. The parameters of such models are:

1. the number of trees (number of estimators),
2. the depth of each tree,
3. the learning rate to optimise the tree.

The Adaptive Boosting (AdaBoost) type of tree was selected as it is among the most widely used in HEP and therefore provides a coherent benchmark [55]. To implement the model, the scikit-learn library [56] *AdaBoostClassifier* is used with the *DecisionTreeClassifier* (in the implemented class *BDTRunner*).

3.2.2 Deep Neural Networks

Feedforward deep neural networks adopt a structure made of artificial neurons to learn a mapping achieving a specific task. The mapping is adapted during training to optimise

a given metric by gradient steps, using a forward pass to produce an output and compute the value of the objective (typically, a loss function) which is then backpropagated through the network to adapt the parameters [57, 58]. This process is called *learning*. The typical structure of such networks is shown in figure 3.2.1. The left plot presents a network with a single hidden-layer. All artificial neurons of a layer t are connected to each neuron of layer $t + 1$, in this case. Some of these connections could be randomly ignored at times, a mechanism described as *dropout* and helping to avoid *overfitting*, the negative tendency to specialise the parameters of the networks to a specific set of data at the expense of a general one [59]. The right plot shows the particular structure of a given artificial neuron: it linearly combines its inputs, weighting them and adding a bias (these weights and biases are parameters of the networks), before applying a non-linear transformation on the result which is then passed to the next layer or used in the objective function.

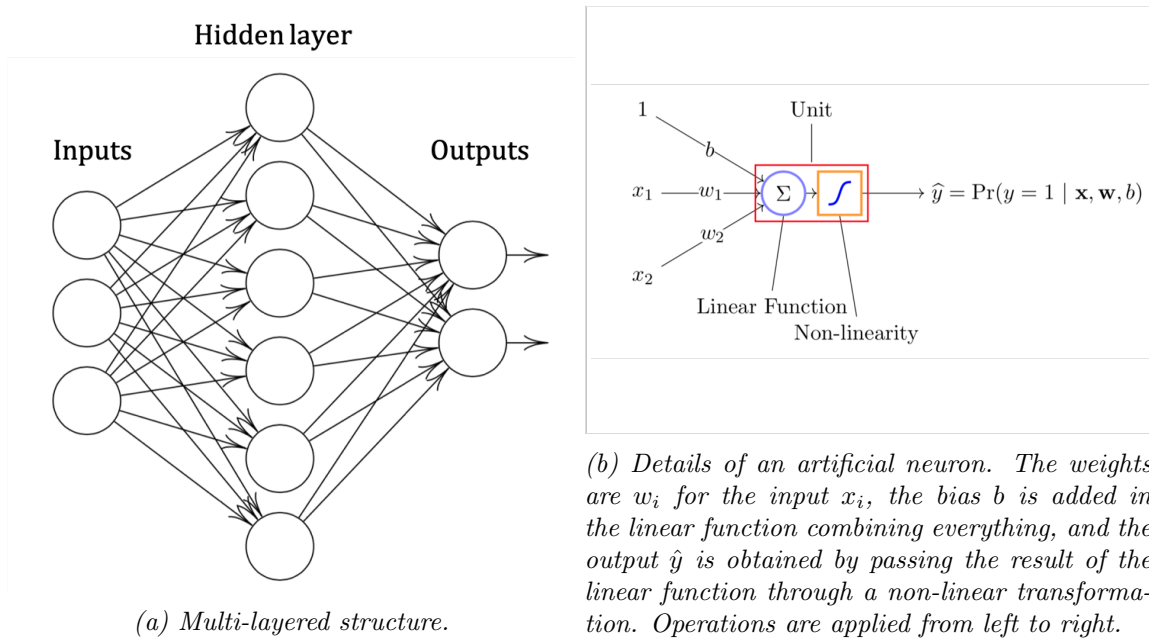


Figure 3.2.1 – The structure of a deep neural network, taken from [59].

All DNN models were implemented with PyTorch [60]. PyTorch was chosen as it is extremely versatile, well-documented, easily moves computation to GPU's, and leaves a lot of control on the training loop. Being written as an object-oriented framework, its classes can be inherited locally to define aggregated modules. It also allows easy interconnections between different components of the codebase, thereby avoiding repetitions. With this last point in mind, the DNN local implementation separates the model definition part

(*NNetwork*) and a *runner* part (*NNRunner*) to interact with the model (initiate, train, load, and test it).

The DNN models implemented in this project have very simple architectures. This is due to the fact the available information for these networks only consists of either 14 (sets 1 to 3) or 11 (set 6) variables, as specified in the next chapter. A DNN quickly reaches saturation on such simple inputs and no improvements were observed by increasing the size or complexity of the networks. The following features are used:

- Adam optimiser with standard parameters (no improvements were obtained by modifying these) [61],
- Weight Decay (WD) with L2 regularisation,
- variable dropout rate on the first layer,
- learning rate schedules.

3.2.3 Junipr

3.2.3.1 Mathematical Basis of Junipr

Junipr stands for *Jets from UNsupervised Interpretable PRobabilistic models*. It learns a probabilistic model scaffolded around the factorisation tree of a jet as reconstructed by any factorisation algorithm, such as C-A. The starting point is a set of n final state four-momenta $\{p_1, \dots, p_n\}$. In the Junipr paper [8], these four-momenta are the final state particles of the jets (the stable products from the set of decays that forms the jet). In this project, the four-momenta are those reconstructed by the calorimeter topoclusters of the ATLAS experiment. At its core, Junipr computes the probability density:

$$P_{\text{jet}}(\{p_1, \dots, p_n\}). \quad (3.2.1)$$

This general probability density is unstructured and does not exploit any prior physics knowledge. The mathematical foundation and motivation behind the Junipr model is the factorisation of the jet likelihood as a tree. Such a structure arises from the approximative sequence of binary splittings ($1 \rightarrow 2$) induced by the factorisation theorem of QCD [62].

This is only but an approximation (a phenomenological approach) of the far more intricate mechanism by which a final state is produced through successive decays in a jet. Scaffolding the model around such a leading order physical model of jet dynamics is expected to ease training and incorporate expert information smoothly into the design. To factorise the probability density, intermediate states are introduced and labelled for each timestep: $k_1^{(t)}, \dots, k_t^{(t)}$, with the superscript indicating the timestep and the subscript enumerating each momentum at that timestep. At each timestep $t = 1, \dots, n$, there are t different states. The final timestep is n and corresponds to the complete set of final state momenta: $\{k_1^{(n)}, \dots, k_n^{(n)}\} = \{p_1, \dots, p_n\}$. The initial state at step 1, corresponds to the global jet momentum called the seed-momentum: $\vec{k}_1^{(1)} = \sum_{i=1}^n \vec{p}_i$.

Between two timesteps, a single $1 \rightarrow 2$ decay, in which a mother decays to give two daughters, brings the set of states from $\{k_1^{(t)}, \dots, k_t^{(t)}\}$ to $\{k_1^{(t+1)}, \dots, k_{t+1}^{(t+1)}\}$ such that, for some unique r in $1, \dots, t$:

$$k_r^{(t)} \rightarrow k_{d_1}^{(t+1)}, k_{d_2}^{(t+1)}, \text{ with } d_1, d_2 \text{ taking a different value in } 1, \dots, t+1.$$

All other momenta are just copied in the new set with their superscript incremented. This factorisation decomposes the jet probability density of equation 3.2.1 into:

$$P_{\text{jet}}(\{p_1, \dots, p_n\}) = \left[\prod_{t=1}^{n-1} P_t(k_1^{(t+1)}, \dots, k_{t+1}^{(t+1)} | k_1^{(t)}, \dots, k_t^{(t)}) \right] \times P_n(\text{stopping} | k_1^{(n)}, \dots, k_n^{(n)}), \quad (3.2.2)$$

where timesteps 1 to $n - 1$ list the decay of a mother of the set of state of the timestep considered. The final term in the product, outside of the main bracket, brings in the probability that the tree does indeed end when reaching the last set of states (which are to be the final states). A further factorisation step of the $P_t(\dots|\dots)$ is necessary to bring forward the $1 \rightarrow 2$ decay dynamics adopted (instead of set-evolution):

$$P_t(k_1^{(t+1)}, \dots | k_1^{(t)}, \dots) = P_{\text{end}}(0 | h^{(t)}) P_{\text{mother}}(m^{(t)} | h^{(t)}) P_{\text{daughter}}(k_{d_1}^{(t+1)}, k_{d_2}^{(t+1)} | k_m^{(t)}, h^{(t)}), \quad (3.2.3)$$

where $m^{(t)}$ is the index of the selected mother in the t -th set, sorted in decreasing-order of energy, and $h^{(t)}$ is a hidden state representing all previous information on the jet (every

timestep $< t$).

The three components separated in this factorisation must be predicted at each timestep $t = 1, \dots, n - 1$:

- $P_{\text{end}}(0 | h^{(t)})$: probability over the set $\mathbb{Z}_2 = \{0, 1\}$, with 0 encoding *not stopping* and 1 encoding *stopping*.
- $P_{\text{mother}}(m^{(t)} | h^{(t)})$: probability over the set $\mathbb{Z}_t = \{1, \dots, t\}$, with $m^{(t)} \in \mathbb{Z}_t$ indicating the candidate mother momenta index in an energy-ordered list (decreasing).
- $P_{\text{daughter}}(k_{d_1}^{(t+1)}, k_{d_2}^{(t+1)} | k_m^{(t)}, h^{(t)})$: probability over the set $\mathbb{R}^4$ of possible branching $k_m^{(t)} \rightarrow k_{d_1}^{(t+1)}, k_{d_2}^{(t+1)}$. Note that this probability is only correctly defined on the subset of $\mathbb{R}^4$ that respects the laws of Physics, in particular energy-momentum conservation (though interestingly the conservation of baryon number is not enforced).

The last two probabilities are also conditioned on *not stopping*, though this is not explicitly stated. Note that *end = stopping* only for timestep n : no decay is to be considered at that moment hence these two branches are not activated (as guaranteed by the limited range of $t = 1, \dots, n - 1$ above). Each P_t , for $t = 1, \dots, n - 1$, is factorised using these three factors, each being conditioned on the past history of the jet encoded in $h^{(t)}$. Crucially, this final term must contain the entire past history of the jet for this approach to be consistent.

3.2.3.2 Machine Learning Implementation of Junipr

The Junipr approach to jet modelling can be implemented by using the recurrent-paradigm of ML, with the PyTorch library. The core part of the model is a recurrent cell, either a plane RNN cell, a Long Short Term Memory (LSTM) [63], or other. It is tasked with generating the hidden state $h^{(t)}$ and to update it with each new node of the tree. At the start of timestep t , the hidden state is updated by taking as input to the recurrent component the previous hidden state $h^{(t-1)}$ and the four-momenta of the pair of daughters that were generated at the previous timestep $k_{d_1}^{(t-1)}, k_{d_2}^{(t-1)}$. Of course, the first node is an exception to this, since no daughters precede it. Therefore, for the first timestep $t = 1$, the hidden state is generated by a DNN mapping from the seed-momentum. Note

that when using an LSTM cell, in addition to the hidden state $h^{(t)}$, the recurrent component also requires a cell state $c^{(t)}$. The latter is not given any specific modelling role in Junipr and is just passed along the sequence. For the initial timestep, it is also obtained by a DNN mapping of the seed-momentum with its own network.

The hidden state information at each node is passed to several different DNN's, combined with input information, to make predictions along the three branches of factorisation presented above. The predictions to make include:

1. Whether there is a decay at the node considered (probability to end). This is encoded by a binary value, 0 being *not stopping* and 1 *stopping*. Achieved by the *end branch*.
2. Which particle, in a decreasing energy-ordered list of available particles, is the next to decay: the index of the next mother. Achieved by the *mother branch*.
3. What dynamical configuration do the daughters have relative to the mother momentum. Achieved by the *daughter branch*.

Following the same order, the input information of these three branches are:

1. The hidden state at the timestep considered: $h^{(t)}$.
2. The hidden state at the timestep considered: $h^{(t)}$.
3. The hidden state + the four-momentum of the mother decaying at the timestep considered: $h^{(t)}, k_m^{(t)}$.

The global Junipr structure when analysing the timestep t is summarised in figure 3.2.2. All DNN hidden layers use ReLu activation and the RNN or LSTM core have a *tanh* activation (DNN's used for the first hidden state also do so). The final layer of each DNN computes a softmax between the discretised class predictions (0/1, the mother index, and the bins of branchings, as explored in the next paragraph). A probability is returned by computing the cross-entropy between the predictions and truth information. Due to the specifics of the PyTorch implementation, it is in fact $-\log(\text{probability}(\dots))$ that is calculated.

Summing these three different log components, the probability of the node is computed:

$$-\log(\text{probability}(\text{node timestep } t)),$$

by the additive property of logarithms. Summing the log-probabilities for each timestep $t = 1, \dots, n$, for a jet made of n final state constituents, gives the probability of the entire jet tree. Timestep $t = 1$ and $t = n$ both adapt the architecture to available information (the full diagrams are presented in Appendix C, figure C.1.1 for $t = 1$ and figure C.1.2 for $t = n$). For timestep $t = 1$, it implies computing the hidden state $h^{(1)}$ by DNN mapping of the seed-momentum. For the final operation, neither the *mother* nor the *daughter branches* are activated.

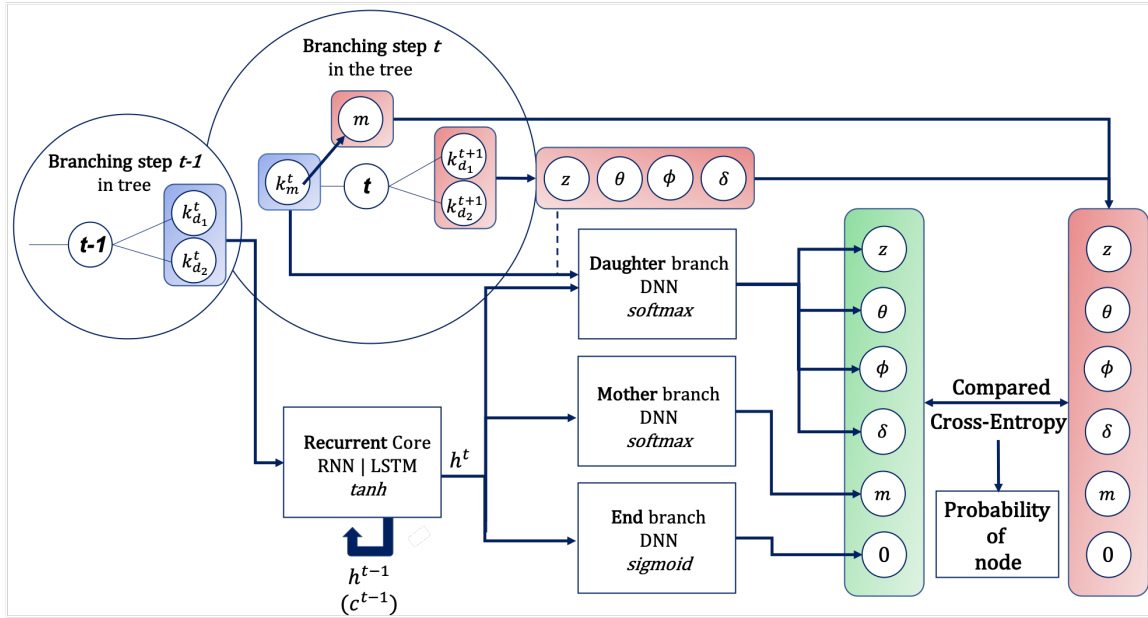


Figure 3.2.2 – Diagrammatic representation of the transfer of information in the Junipr model when processing timestep t (excluding the initial and final timesteps). Inputs to the network are in the blue box, predictions of the network are in the green one, and the input information to which it is compared is in the red box. The two top left circles represent branching nodes in the jet tree, in the order established by the clustering history. These may be located on any branch of the tree (they need not come from the same branch in a sequential order). The blue rectangular frames represent the five components of the Junipr network: the four on the left are the recurrent core, producing the hidden state $h^{(t)}$, and the three DNN mapping to predictions, while the one to the right computes the probability of the node by measuring the cross-entropy between the predictions and truth information. The c^{t-1} inside parenthesis is only used for the LSTM recurrent core (cell state). The dashed line from the branching input information joining with the mother momentum $k_m^{(t)}$ is a special transfer used for the multibranch version of the daughter branch (as suggested by equation 3.2.4).

The last particularity of the Junipr implementation consists in the different approaches explored to model the *daughter branch*. This branch is tasked with describing the decay given the mother momentum involved in the process. A complete physical basis to describe this process consists of the four variables z , θ , ϕ , and δ . As explained at the end of section 3.1.1, these are respectively the fraction of energy of the soft daughter to mother, the angle between the soft daughter and the mother, the angle of the plane of decay in reference to the mother and the y -axis of the detector, and the angle between the hard daughter and the mother, respectively. The *daughter branch* is then modelled as:

$$P_{\text{daughter}} \left(k_{d_1}^{(t+1)}, k_{d_2}^{(t+1)} \mid k_m^{(t)}, h^{(t)} \right) = P_{\text{daughter}} \left(z, \theta, \phi, \delta \mid k_m^{(t)}, h^{(t)} \right).$$

These variables are continuous, hence the predictions of this branch would have to follow a regression approach. To simplify the task, each of these four variables has its values discretised into a given number of bins called *granularity*. The job of the *daughter branch* is then simply to predict the correct bin for each variable, turning the regression task into a classification one (where the right bin equates to the correct class). The authors of Junipr comment in their original paper [8] that no significant decrease in performance is observed with this discretised approach compared to continuous modelling (using an energy model). Two types of DNN are implemented to model this branch:

- *Unique branch*: a single DNN is deployed to map the mother momentum $k_m^{(t)}$ + hidden state $h^{(t)}$ to the four variables. The size of its output layer is the granularity times the number of variables. For example, if the values of each variable are discretised into 10 bins (meaning granularity = 10), the output should be of size 40, with the 10 first corresponding to z , the 10 next to θ , and so on.
- *Multiple branch*: four different DNN's are used to make predictions, one for each variable. Each of these DNN's has an output layer of a size equal to the granularity. A further factorisation step of the *daughter branch* is applied here, with the intention of helping the model analyse branchings. Each successive DNN, when mapping the variables in the order z , θ , δ , to ϕ , receives the truth input information of the previous variables. For example, δ receives the following list of inputs: $\left[\theta, z, k_m^{(t)}, h^{(t)} \right]$.

This corresponds essentially to the following factorisation:

$$P_{\text{branch}}(z, \theta, \phi, \delta | k_m^{(t)}, h^{(t)}) = P_z(z | k_m^{(t)}, h^{(t)}) \times P_\theta(\theta | z, k_m^{(t)}, h^{(t)}) \times P_\delta(\delta | \theta, z, k_m^{(t)}, h^{(t)}) \times P_\phi(\phi | \delta, \theta, z, k_m^{(t)}, h^{(t)}). \quad (3.2.4)$$

An important constraint on the Junipr network is that the hidden state $h^{(t)}$ must be given enough flexibility to model the entire state of the jet at the given timestep. In fact, all global information about the jet state should be accessible through $h^{(t)}$. It must be sufficient to predict whether to end the tree, the next mother to decay, and, combined with the mother momentum, the dynamics of the next decay. In this respect, the clustering algorithm used to construct the factorisation tree, C-A or anti- k_T , plays an important role. As suggested in section 3.1.2, in particular relating to figure 3.1.2, these two algorithms build jets of different structures. Specifically, anti- k_T does not approximate the collinear structure of QCD. Each intermediary branching radiates a final state particle in the right direction and, as a consequence, the global collinear structure of the jet can only be fully appreciated when reaching the final states. By contrast, C-A builds the jet along several branches as dictated by the collinear structure, which thereby emerges from the growing of these multiple branches. The collinear structure is directly present and progressively refined while, for anti- k_T , it only appears towards the end. The task is therefore much more demanding for the recurrent core of a Junipr model analysing anti- k_T jets. It must be able to grasp the global collinear structure that will only be accurately revealed around the last nodes. This discussion motivates the use of the C-A algorithm when analysing the jet [8]. In fact, using C-A for jet analysis and anti- k_T for clustering is a very common practice in Particle Physics [13, 64].

The Junipr networks implemented in the two papers introducing the method [8, 9] and in this project might appear surprisingly light compared to the average Natural Language Processing (NLP) model. The authors of Junipr observed that a simple RNN cell with a hidden state of size 100 is sufficient for the task of tagging. All the DNN's in their models have a single hidden layer of 10 units. The relative simplicity of these networks is heuristically expected to suffice for two particular reasons:

- The ML network is scaffolded around a physics model. It means a lot of the informa-

tion is fed to the model through the structure of the data and supports the learning process.

- The task itself is not as complex as typically encountered in the ML literature. Contrary to NLP data, jets do not have an ordered structure to be learnt. While the meaning of a sentence may depend on the order of its words, no such physically-meaningful order exists among the constituents of a jet. An artificial order is built-in by structuring the information into a tree with a clustering algorithm. The important difference is that the order thus introduced is fixed in the data, not left to the model to learn. As a result, there is a difference in complexity between the jet analysis and sentence processing cases of an approximate $n!$ factor, where n is the size of the sequence analysed (respectively, the number of final state constituents or words in the sentence).

3.2.3.3 Applications of Junipr

The Junipr model has several possible applications. It can be used for discrimination, generation, reweighting of simulations, and even for analysing jet substructures (this last point is discussed in sections 4.2 and 4.3 of the next chapter).

Discrimination: two variants of Junipr can accomplish this task.

1. Train several Junipr models on different datasets. Each one of these models is tasked with learning a correct mapping from its inputs to the probability distribution underlying the jet type dominating in its dataset. Note that this does not require pure samples, the paradigm of weakly supervised learning being naturally adopted in impure situations. Discrimination is achieved by a likelihood ratio test, the most statistically powerful discriminant between two hypotheses under the Neyman-Pearson lemma [65]. Specifically for the present case:
 - Train two *unary* Junipr models: a quark model (respectively, gluon model) on a dataset with a larger fraction of quarks (gluons). Given an input x , the output of each model is the probability of observing x conditioned on the model: $P(x | \text{quark model})$ and $P(x | \text{gluon model})$.

- Given a new *jet*, compute the likelihood ratio test:

$$\frac{P(\text{jet} | \text{quark model})}{P(\text{jet} | \text{gluon model})} > \text{threshold}, \quad (3.2.5)$$

with the *threshold* based on the properties of the Receiver Operating Characteristic (ROC) curve. Values above the threshold are label as quarks (1) and below the threshold as gluons (0).

This tagging method is easily extended to the case of N classes, requiring to train only but N models.

2. An issue with the previous multi-unary approach is that each model is trained on a dataset of jets dominated by a certain label. This implies that each unary model has seen only but a small sample of the other label(s). Given that ML models classify data outside of their distribution poorly, the models learnt in this way are not perfect and, for example, the probability $P(\text{gluon jet} | \text{quark model})$ cannot be accurate. Another issue with the previous approach is that it does not directly train on the task of discrimination and, therefore, does not adapt its weights in the optimal way for the job. To remedy both weaknesses, the authors of Junipr proposed in a secondary paper [9] to combine a pair of *unary* Junipr models into a *binary* Junipr model. The approach is mathematically grounded in Bayes' theorem. The probability of a label (or model) given a jet can be re-expressed as:

$$P(\text{quark} | \text{jet}) = \frac{P(\text{jet} | \text{quark}) P(\text{quark})}{P(\text{jet})}. \quad (3.2.6)$$

In the binary classification case, since $P(\text{jet} | \text{quark}) + P(\text{jet} | \text{gluon})$ must equal 1 (the jet is forced to have a label so that the two labels form a partition), one can adapt the above equation into:

$$P(\text{quark} | \text{jet}) = \frac{P(\text{jet} | \text{quark}) P(\text{quark})}{P(\text{jet} | \text{quark}) P(\text{quark}) + P(\text{jet} | \text{gluon}) P(\text{gluon})}. \quad (3.2.7)$$

In this expression, $P(\text{quark})$ and $P(\text{gluon})$ are respectively the quark and gluon fractions, usually called f_{quark} and f_{gluon} . A similar reasoning holds for $P(\text{gluon} | \text{jet})$.

Combining the log of both results leads to the binary Junipr discrimination objective to maximise:

$$\mathcal{L} = \sum_{\text{quark jets}} \log \frac{P(\text{jet} | \text{quark}) f_{\text{quark}}}{P(\text{jet} | \text{quark}) f_{\text{quark}} + P(\text{jet} | \text{gluon}) f_{\text{gluon}}} + \sum_{\text{gluon jets}} \log \frac{P(\text{jet} | \text{gluon}) f_{\text{gluon}}}{P(\text{jet} | \text{quark}) f_{\text{quark}} + P(\text{jet} | \text{gluon}) f_{\text{gluon}}}. \quad (3.2.8)$$

In the present case, this objective can be further simplified as the quark and gluon fractions are equal ($f_{\text{quark}} = \frac{1}{2} = f_{\text{gluon}}$). The objective $\mathcal{L}$, therefore, reduces to the binary cross-entropy [9]. The precise objective used in this project is in fact slightly different. It is based on the log-likelihood ratio test and given a layer of sigmoid to smoothen learning (adding the sigmoid was observed to speed-up training and reduce the variance). The binary objective (given an equal fraction of quark and gluon jets) used in this project is:

$$\mathcal{L}_{\text{used}} = \sum_{\text{quark jets}} \log [\text{sigmoid}(\log P(\text{jet} | \text{quark}) - \log P(\text{jet} | \text{gluon}))] + \sum_{\text{gluon jets}} \log [1 - \text{sigmoid}(\log P(\text{jet} | \text{quark}) - \log P(\text{jet} | \text{gluon}))] \quad (3.2.9)$$

In this situation, $\text{sigmoid}(\log P(\text{jet} | \text{quark}) - \log P(\text{jet} | \text{gluon}))$ is the probability that the jet observed is a quark jet according to the binary model. Consequently, the objective is also a version of binary cross-entropy. Interestingly, the binary case can easily be extended to the n -ary case by using a softmax operation on the set of log-likelihoods instead of a sigmoid. Note that binary Junipr requires a second leg of training compared to the multi-unity case. However, the binary approach performance was shown to be far superior on generated level data [9].

Generation and Reweighting: generating data and reweighting simulations.

While these two applications are not studied in the present work, they also represent important tasks in the context of Particle Physics. All elements presented in this section come from the main Junipr paper [8]. Interestingly, unary Junipr models can generate new jet trees matching a learnt distribution. This is done by successively sampling from the

unary Junipr probability distribution. First, the *end branch* is sampled to decide whether to add a new node. If the answer is positive, a mother index and decay configuration are sampled from the relevant distributions and the jet and states configuration are updated. These steps are repeated until the first one decides to end the tree. This generative process is quite efficient given the relatively small dimensions of the sampled probability distributions. In the original paper, a good agreement with Pythia generated-level simulations was obtained in the task of generating quark jets [8]. However, some drawbacks were also uncovered:

- Sampling in this way is not as fast as dedicated methods for generation, such as MC and deep generative models.
- The model is inherently sequential: it uses information from the previous nodes to make predictions at the next nodes. This means that any early mishap reverberates all along the jet tree: there is a feedback effect that amplifies any small initial defect. The algorithm is therefore very prone to chaotic effects.

More importantly in HEP, Junipr can be used to reweight distributions of samples. For example, assume a set of data was collected in the ATLAS experiment and a Junipr model was fitted to this signal, giving a probabilistic model P_{ATLAS} . Say a simulator, for example using an MC engine, generated a sample at the generated level and a unary Junipr was fitted to this data, giving a probabilistic model P_{sim} . One can easily reweight the simulated distribution to match the collected data by computing the ratio $P_{\text{ATLAS}}/P_{\text{sim}}$ event-by-event. This implements a form of importance sampling resulting in a reconstructed level simulation. Such an approach has the potential of removing the need for MC fine-tuning, a computationally expensive process.

3.2.3.4 Optimising Junipr

This final section on Junipr explores some implementation tricks and adjustments to improve performance. Several hurdles were encountered when implementing and running Junipr models and solutions had to be designed to meet these challenges.

Scaling Inputs

To reduce the parametric shift in the network, the input data is scaled to the range $[0, 1]$. In the Junipr paper, this is achieved by a transformation taking into account information used to generate the data. Mainly, final state constituents having less energy than a parameter called E_{sub} are removed. Their data is further constrained to a very specific jet energy range of $[450, 550]$ GeV [8] or $[500, 550]$ GeV [9]. Such phase space configurations let them employ a natural scaling to the $[0, 1]$ range, as indicated in the beginning of Appendix C.3. The issue in the present implementation is that no such scaling can be applied as the phase space configuration is not sufficiently restricted: the energy of a jet can take any value in $[20, 13000]$ GeV (first bound is due to the p_T cut, second to the centre of mass energy of collision). When applying the original Junipr scaling to the data of sets 4 and 5, some variables are not mapped to the expected $[0, 1]$ range but to a larger interval, even containing negative values. This was observed to happen for different sets of parameters and to be unavoidable given the non-centred nature of the datasets and the properties of their scaling. Conservative parameters lead to spurious values for energetic jets, while overly optimistic values lead to underflow for low-energy jets. This slight issue in appearance is, in fact, a serious problem when considering the discretisation of the *daughter branch*. The $[0, 1]$ range is indeed discretised into the number of bins prescribed by the demanded granularity. Any underflow (respectively, overflow) is thus trimmed to the bottom (top) bin. This has a blurring effect on the data given to the Junipr model that can only decrease performance. Even choosing the least constraining values of the parameters, many variables are crashing out of the $[0, 1]$ window.

To remediate this problem, a new set of scaling functions and parameters is introduced. It relies on the property of the following family of functions:

$$f_{\text{scaling}}(X; \alpha_X, X_{\text{limit}}) = \frac{\log(1 + \alpha_X \cdot X)}{\log(1 + \alpha_X \cdot X_{\text{limit}})}, \quad (3.2.10)$$

where α_X is a scaling parameter associated with variable X and X_{limit} is the proposed top value of X . This equation applies a mapping $[0, X_{\text{limit}}] \rightarrow [0, 1]$ with no underflow (X must be $\geq -1/\alpha_X$). Interestingly, the top value is easy to adapt (being simply the value of X_{limit})

and α_X offers a way to control the curvature of the function. Increasing α_X , the curve has a steeper initial slope which makes it possible to focus the bin sensitivity (controlling the spread in X values between bins) to the low values of the signal, as suggested in figure 3.2.3. The parameters of α_X chosen in this example are not random ones: 50 is used for α_θ and 5,000 for α_δ (both with a top limit of π). The procedure to decide what pair of parameter values to assign to each variable relies on analysing the collected data and placing more emphasis on the range of the signal most commonly observed. For example, the value chosen for δ gives more variation close to 0 than that taken for θ (since $\alpha_\delta = 5,000 > 50 = \alpha_\theta$). This is adopted as δ is more likely to be small, being the angle of the hardest daughter to mother. Indeed, the hard daughter carries a higher fraction of p_T and is thus more aligned in the direction of the mother than the soft daughter. Appendix C.3 presents the chosen parameters and associated scaling for each variable.

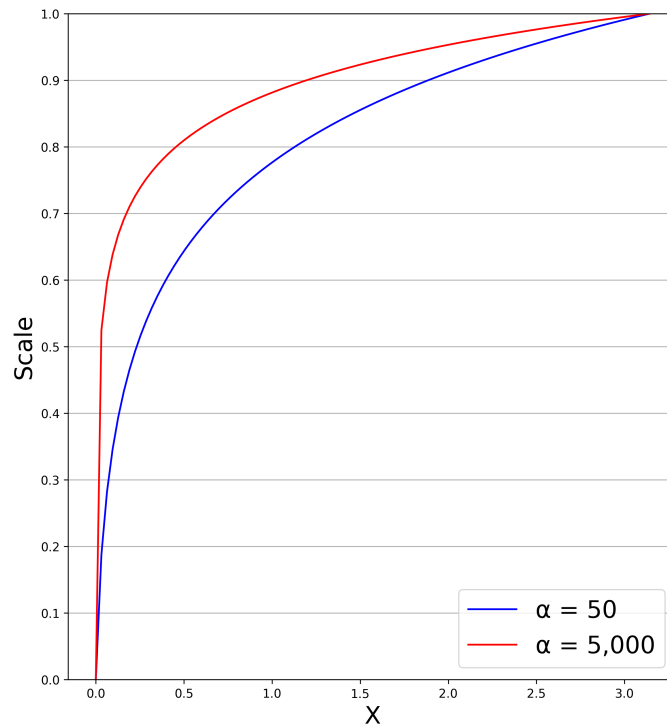


Figure 3.2.3 – Function 3.2.10 for $X \in [0, \pi]$, $X_{\text{limit}} = \pi$, and two different values of α_X : in blue $\alpha_X = 50$ and in red $\alpha_X = 5,000$. The grey horizontal lines indicate the limits of the bins in the case of a granularity = 10.

As explored in Appendix C.3, the mother and daughters four-momenta are also scaled, with their own sets of parameters. This is purely in the interest of decreasing the parametric shift required from the network, as this information does not get discretised.

Training Tricks

To help train the Junipr models, several different elements were added to the implementations, the most important of which are:

1. Favours a concurrent approach by opting for multiprocessing whenever possible.

Two elements materialise this point:

- Doing the entire forward and backward passes in a batch-way. The first implementation ran sequentially, as it is the easiest way of dealing with sequences (jets) of different sizes with padding. The padding size is one of the fixed parameters of the model, at 120 so that it is longer than any realistic (and observed) jet sequence. By defining *mask* tensors in the Junipr dataset class (inheriting from the PyTorch eponymous class), the whole processing can accept batches, with the final loss being masked when corresponding to padded values. Fortunately, PyTorch is excellent at managing padded tensors in recurrent ML components thanks to the duo of methods *pack_padded_sequence* and *pad_packed_sequence*. A final trick was used for the *mother branch*: to limit the softmax layer to the number of present particles at the given timestep (t possible mothers at timestep t), all entries of the *mother branch* output tensor of size $(batch, padded_length, padded_length)$ that need to be rejected are eliminated by summing an upper triangular tensor (for each batch sample, so it is triangular with respect to the last two dimensions) made of $-\infty$ values. These entries are then neutralised by the softmax step and no gradient propagates through them. The speed-up obtained by processing everything in a batch-way is significant for large batch sizes (for a batch size of 400, one step of the batched-version of Junipr is faster by a factor 8 compared to the sequential implementation).
- The Junipr data is read from a *json* file by the Junipr dataset class (each sample is a dictionary containing several tensors), turned into dictionaries of padded tensors, and gathered by a PyTorch *dataloader* into batches. This process has a non-negligible cost, at about 3 % of the time to process a whole batch for

large batches (~ 400 samples). This number rises to 16 % for small batches (~ 10 samples). To accelerate this step, some optimisation was carried out on the *num_worker* argument of the PyTorch *dataloader* function. A value of 1 was observed to be optimal: it lets the dataloader process two batches in parallel. When a new batch has to be passed to the model, it is taken from the cache of a worker that finished preparing a batch. This worker can then set out to process a next batch, skipping the data loaded by the other worker. The time to process new data is dramatically reduced in this way. Implementing this requires some care when running on the HTCondor system, as a conflict in accessing the data leads to a memory leak: the program copies the same data several times, quickly running out of storage and crashing the job. The issue can be solved by using the *manager.list()* wrapper from the multiprocessing library of Python 3.6, authorising shared access to the same data without copying it.

2. To manage the full *json* datasets, of about 8 GB for each label, a chunked version of the Junipr dataset class was developed, called *JuniprLargeDataset*. The quark and gluon samples are previously separated into *json* chunks of determined size (of about 400,000 jets each). When accessing the data during operation, a list of indices to be loaded in a batch is passed to the dataset class. If these indices correspond to the current chunk in memory, the data is just transferred normally. However, if some of the indices point to an unloaded chunk, it is loaded into memory when required and replaces the previous chunk. The only randomisation allowed in this approach is inside the chunks, not across them, to reduce to a minimum the number of chunk loading. This lack of inter-chunk randomness is not an issue as the chunks are made quite large to also reduce the number of I/O's required, so that the intra-chunk randomness suffices. Nevertheless, this method still requires more I/O's than in the regular case, an unavoidable consequence of using *json* files (no lazy-reading) of such large sizes. Two types of chunks are considered:

- random chunks of 400,000 jets of any energy,
- energy-ordered chunks, focusing on a specific range of jet energy (large energy bins are chosen to have large chunks).

Note that the chunked version is not compatible with a `num_worker` $\neq 0$. Serious concurrency issues can indeed arise in this case. For example, if each parallel worker needs a different chunk, this conflict can lead to a disastrous number of I/O's (at each new index, a worker has to reload the chunk it is using since the other worker modified the chunk in memory). A more involved solution would restrict the workers to focus on the same chunk and wait for each other before loading a new one (when a single batch-worth of data is left in a chunk to analyse). Another option would be to further divide the chunks in two and authorise each parallel worker to have its own chunk in memory. In this scenario, the chunks would be separated in two sets, one for each parallel worker. These two approaches were not implemented due to both a lack of time and necessity, the current setup proving sufficient for the needs of the project.

3. Most of the jobs ran on HTCondor and were launched from the pplxint machine of the Particle Physics subdepartment. Unfortunately, the system proved quite unstable at times, interrupting jobs for no reason, hiding error logs, or being disconnected altogether. To increase the resilience of the jobs, saving points and checkpointing were added to the training loop. Logging also played a large role in controlling the advancement of processes. TensorBoard logger, updated on the pplxint machines and transferring their outputs locally via ssh, provided a helpful tool, as did text loggers. A decisive advantage of the HTCondor system came from the possibility to run a large number of jobs in parallel, providing an invaluable speed-up and a flexible setup to adapt and launch new jobs based on the progress of prior ones.

Summary of the Chapter

In this chapter, the methodology adopted in the project is presented. From data collection, to processing, and analysis, to the descriptions of the three machine learning models selected: boosted decision trees (used as a benchmark given their popularity in the field), deep neural networks (for their efficiency with high-level variables), and the Junipr framework (for its promising results at the generated level and interpretability). A global overview of the the codebase implementing this methodology is available in [Appendix A.1](#).

The objectives of the project and the methodology to explore the selected questions introduced, this chapter uncovers and assesses the results obtained in the course of the work. Each experiment carried out and presented in this section arose from interrogations stemming from previous endeavours. The adopted presentation aims to follow this natural order, thereby justifying the steps taken and providing a global view of the results obtained.

4.1 Limitations of Boosted Decision Trees and Deep Neural Networks

Why are BDT's and DNN's not enough? Why is there a need for some deeper model? The simplest and obvious answer to that question is the need to maximise the accuracy of the discriminator. Particle physicists rely on a large toolbox of methods to decorticate their data and simulations and unveil the underlying theoretical basis governing the laws of nature. The more accurate the techniques they use, the finer the resolution of their analyses. The world of Particle Physics plays at vastly different scales, leading to extremely fine details easily being missed and blurred away by dominating backgrounds. For example, in the MultiJet dataset (set 3), some processes have a relative cross-section to the likeliest

process, the production of dijet events, in the order of 10^{-16} . It means that, in order to satisfy the SM “rate of productions”, for each event of these unlikely processes, there should be 10^{16} events of dijet. If the physics analysis was targeting one of these unlikely processes, it would need to distinguish interesting events among a 10^{16} times larger set of background events before even accounting for other processes.

Particle Physics has traditionally focused on separating the searched signals from backgrounds by incorporating every possible information on their differences into a set of cuts. Such an approach requires a deep knowledge of the Physics involved in both cases as well as collecting high-level information. The task then reduces to choosing the best set of cuts on the available variables. This effort benefits greatly from BDT and DNN as these methods are specialised in finding the best set of cuts from available information. The problem is that the only meaningful information to be extracted resides in the input variables, bringing two major disadvantages to BDT’s and DNN’s:

1. Information-rich variables have to be collected. This, in Particle Physics, requires a fair amount of processing and cannot be considered for fast online triggering operations, where processes have to run at a comparable frequency to the activity in the detector.
2. The information is limited to the variables given. They are incapable of exploiting the structures and patterns theoretically and phenomenologically associated to jets.

Other issues related to BDT’s and DNN’s are the difficulty in interpreting their results and their limited application, being only able to work on the task for which they were trained (no dual discriminative and generative use for example). They, however, have the attractive advantage of being easy to train, both in terms of implementation and duration.

The first part of the project, due to unexpected difficulties encountered in collecting data compatible with Junipr, focused on training several BDT’s and DNN’s models on the data of sets 2 and 3 (set 1 was used as prior version to prepare the models). Only 1 % of the available data was analysed, representing an already sizeable ~ 15 M jets, as using extra amounts of data was not observed to bring any improvement. The metric by which

the performance of a model is measured is the Area Under the Curve (AUC) drawn by the ROC, for its ability to provide an aggregated measure of performance over different thresholds in the binary discrimination case. The variables from sets 2 and 3 that are used in these models are presented in table 4.1.1. In all cases, the fraction of quark and gluon jets used are forced to be of equal size (~ 7.5 M jets each).

Variable	Explanation
jetPt	Jet p_T
jetEta	Jet η
jetPhi	Jet ϕ
jetMass	Jet mass
jetEnergy	Jet energy
jetEMFrac	Fraction of the jet energy collected in the ECAL
jetHECFrac	Fraction of the jet energy collected in the HCAL
jetChFrac	Fraction of the jet p_T associated with charged particles
jetNumTrkPt500 jetNumTrkPt1000	Number of tracks with a p_T above 500 or 1000 GeV associated with the jet
jetSumTrkPt500 jetSumTrkPt1000	Sum of the p_T of tracks with a p_T of at least 500 or 1000 GeV
jetTrackWidthPt500 jetTrackWidthPt1000	Width of the structure formed by the tracks with a p_T above 500 or 1000 GeV associated with the jet

Table 4.1.1 – The engineered variables used from sets 2 and 3 to train the BDT’s and DNN’s.

The BDT model parameters were optimised with the help of grid search on random folds of the test data [66]. Three successive dichotomous steps were taken to explore the search-space $\{\text{depth} \in [2, 5]\} \times \{\text{n_estimator} \in [100, 700]\} \times \{\text{learning rate} \in [10^{-3}, 1]\}$. The optimal configuration was found to deliver stable performance, even under small changes to the parameters. It corresponds to:

- number of trees = 300,
- depth of each tree = 3,
- learning rate = 0.15.

The DNN architecture was optimised by running 7 models on dataset 2 and 8 models on dataset 3. The models have ReLu activations for the hidden layers, a single artificial neuron with sigmoid activation for the last layer, and the objective function to minimise is the binary cross entropy with the truth label. The following possible set of parameters values was considered for the rest of the architecture:

- number of hidden layers: 2 or 3,
- number of units per hidden layer: 20, 32, 64, or 128,
- weight decay applied to each layer: 0.01, 0.001, 0.0001, or 0.000001,
- dropout rate applied to the first layer: 0, 0.1, or 0.2,
- learning rate: 0.001 or 0.0001.

The optimisation was done sequentially, avoiding the grid-search approach to reduce the processing time. The structure adopted is as follows:

- number of hidden layers: no difference was observed between 2 and 3 layers of 20 units so 2 was kept (smaller model),
- number of units to each hidden layer: 32 was slightly better,
- weight decay applied to each layer: 0.0001 was found to be optimal,
- dropout rate applied to the first layer: 0 was found to be optimal,
- learning rate: both values offer comparable result. The following learning rate schedule was adopted: learning rate of 0.001, 0.0005, and 0.0001 each for each successive third of training.

Naturally, selecting parameters sequentially on different runs does not guarantee that the resulting configuration offers the optimal performance. This approach was adopted only because it is much faster and because the AUC obtained from different models is very similar. Indeed, figure 4.1.1 presents the ROC curves obtained by training a subset of the models on set 2. Clearly, the performance is stable across models. Figure 4.1.2 shows some ROC curves obtained by training on set 3. In both plots, the orange curve labelled "Given

BDT" represents the AUC obtained by a BDT model used in the ATLAS collaboration. Its performance, as suggested by the two plots, is slightly lower than all models derived here. This is mostly due to the probably significant differences between the dataset used for training it and the ones employed in this project. In all likelihood, the ATLAS BDT weighted the processes depending on their cross-section. In the present test, it is exposed to some very rare processes with a probability far greater than their corresponding relative cross-sections. Therefore, the different result in AUC is not surprising and the ATLAS BDT is simply included to display a “benchmark of the benchmark”.

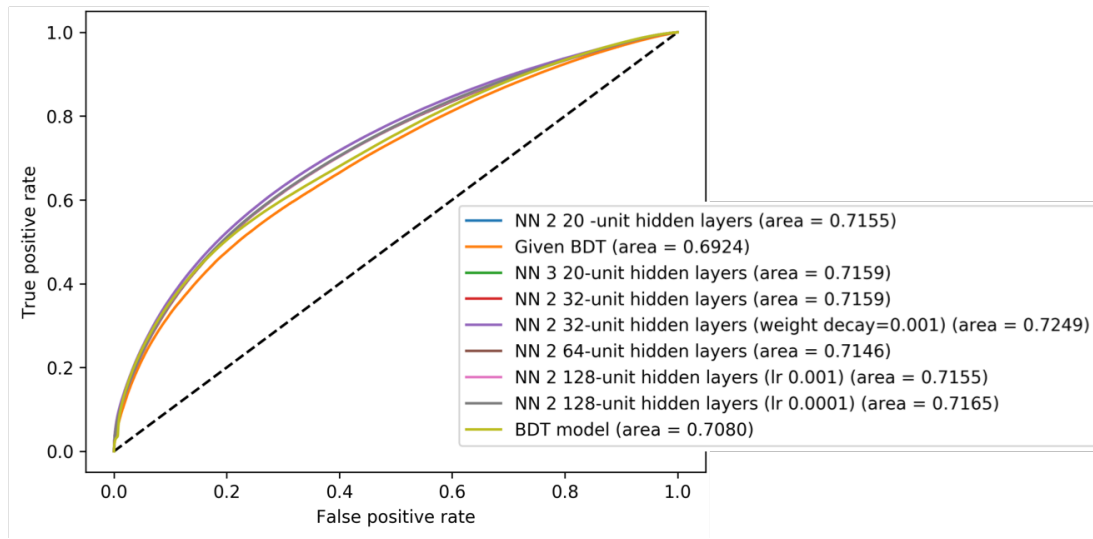


Figure 4.1.1 – ROC curves obtained by different models trained on set 2. “Given BDT” is an ATLAS pre-trained tool.

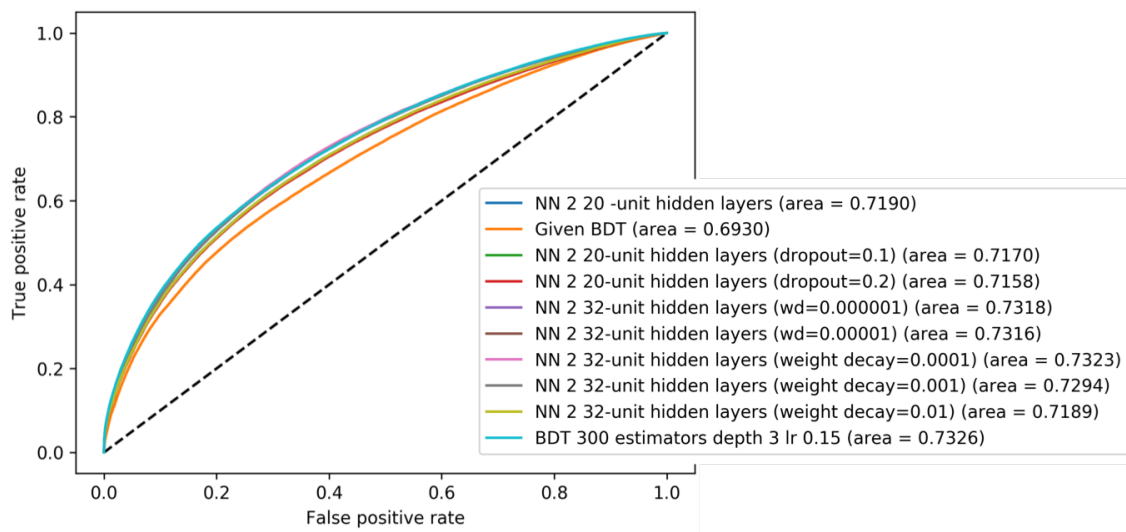


Figure 4.1.2 – ROC curves obtained by different models trained on set 3. “Given BDT” is an ATLAS pre-trained tool.

Note how close the ROC of these different models are. It seems that all models are close to exploiting the discrimination power of their input variables to their full extent and, therefore, the accuracy saturates. The AUC's obtained are comparable to the typical performance reached by cut-based, BDT, and DNN approaches [27].

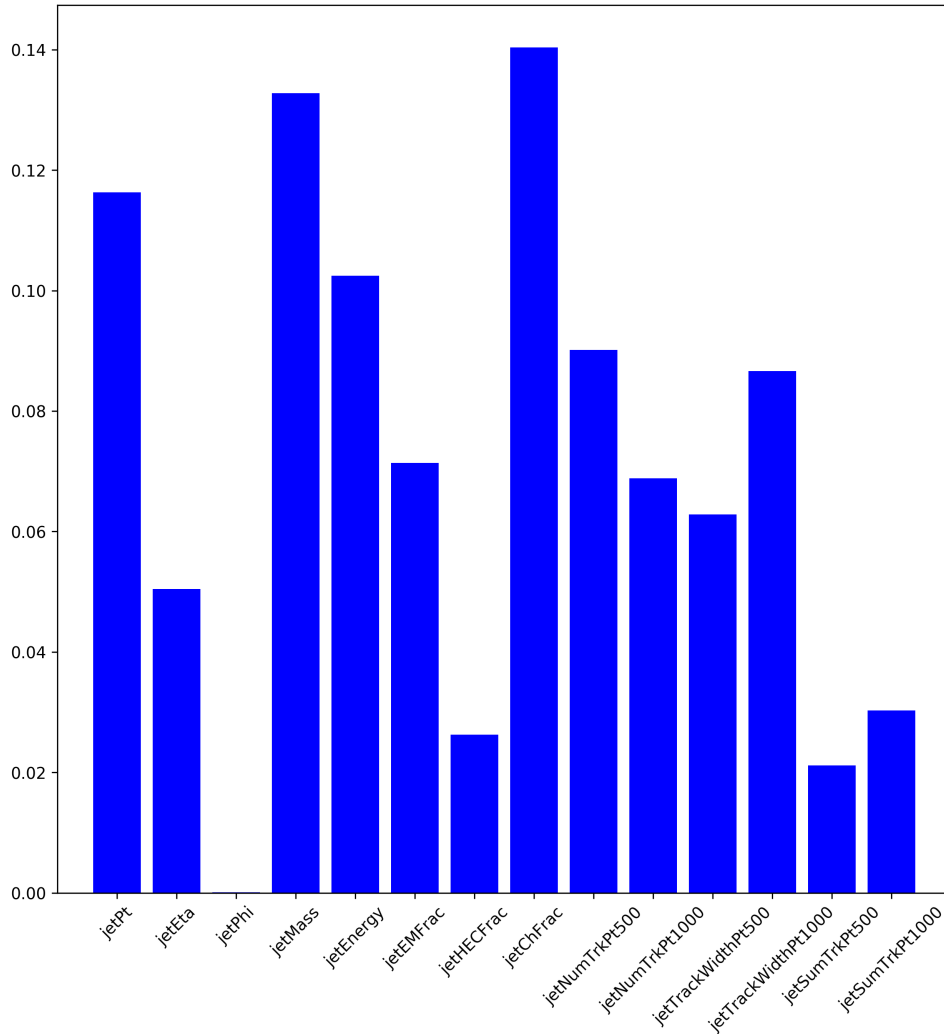


Figure 4.1.3 – The importance of each feature used in the optimal BDT for set 3.

To have a closer look inside the models and better understand how they operate, the importance of each feature of the BDT trained on set 3 is displayed in figure 4.1.3. It indicates that the fraction of energy associated with charged particles (*jetChFrac*), the jet mass, and its p_T are the three most important variables to discriminate between quark and gluon jets. Given the *jetChFrac* distribution is rather similar for quark and gluon

jets, it can only play a significant role by orienting later cuts (figure B.1.7 of Appendix B presents the *jetChFrac* for sets 4 and 6, which is similar to that of set 3 used here).

The obtained hierarchy of variables for discrimination differs significantly from that proposed in several generated level studies. These suggest the number of particles required to account for 95% of the jet p_T , called n_{95} , and the jet width are the most discriminative variables. These two variables are not directly available in this case, though equivalent notions are offered by the *jetNumTrkPt* 500 and 1000 for n_{95} and by the *jetTrackWidthPt* 500 and 1000 for the jet width. However, as uncovered in [23], the blurring effect of the reconstruction phase for detector simulations reduces the power of variables based on the number of constituents (the cited paper uses Delphes to simulate an average detector, not the specific ATLAS detector). This might explain why dynamical variables, such as *jetChFrac* and the jet mass, are favoured in this case by the BDT, given the data is fully reconstructed to match the ATLAS detector signal. Such information, at the level considered here, is indeed among the best reconstructed. A reassuring element from the feature importance distribution is the lack of any discriminative use for the jet ϕ distribution. This variable is indeed absolutely incapable of offering any discriminative power since it is uniformly distributed: physical processes have a symmetry with respect to ϕ .

The BDT and DNN models trained on set 6 are presented in section 4.4, dedicated to a comparison between these benchmarks and the Junipr approach. The variables are listed in table 4.1.2, each referring to figures displaying the distributions for both the quark and gluon samples (all these variables have their distribution shown in this dissertation). Note that the list does not contain exactly the same variables as those listed in table 4.1.1. These differences come from the raw data used for set 6, as some of the previous variables were not contained in the DAOD_JetM6 distributions. The smaller set of variables is not expected to prove too detrimental given the important information on the jet is still available, particularly the width and the dynamical variables.

Variable	Explanation
jetPt	Jet p_T (figure 3.1.8)
jetEta	Jet η (figure B.1.3)
jetPhi	Jet ϕ (figure B.1.4)
jetMass	Jet mass (figure B.1.5)
jetEnergy	Jet energy (figure 3.1.7, same for quark and gluon jets)
jetWidth	Jet width (figure 3.1.10)
jetEMFrac	Fraction of the jet energy collected in the ECAL (figure B.1.6)
jetChFrac	Fraction of the jet p_T associated with charged particles (figure B.1.7)
jetSumTrkPt500	Sum of the p_T of tracks with a p_T of at least 500 (right of figure B.1.2)
jetNumTrkPt500 jetNumTrkPt1000	Number of tracks with a p_T above 500 or 1000 GeV associated with the jet (figures B.1.8 and B.1.9)

Table 4.1.2 – The engineered variables from set 6 used for the BDT and DNN benchmarking models.

4.2 Unary Junipr

This section presents results concerning unary Junipr models. Each configuration considered is run both for a quark and a gluon model. These pairs of models can then be combined in the binary case to perform discrimination. Taken alone, they only offer information on the likelihood of a jet as arising from a given model.

Training a unary model is equivalent to performing maximum likelihood estimation on the Junipr network parameters. The weights are indeed optimised to maximise the probability of observing the input data under the model (corresponding to the enriched label in the dataset). In fact, the PyTorch implementation of Junipr minimises the inverse log-likelihood, though, of course, both objectives are exactly identical. The probabilities of jets outputted by Junipr are, therefore, expressed as $-\ln(\text{probability})$. Most of the procedure adopted to train and implement the Junipr models attempts to follow the steps of the two Junipr papers to make the results as comparable as possible [8, 9]. However, substantial differences emerge from the significantly different type of data employed: low-

level (reconstructed) topoclusters momenta versus their high-level (generated) final state particles momenta. A comparison between the results obtained is, therefore, expected to uncover differences arising from the data and not the procedure.

In total, 10 pairs of different unary Junipr models were trained for the purpose of this project, with 5 of them on intermediary versions of the dataset (incomplete and/or unscaled in energy - these are not included in the present dissertation). The names and general information regarding the different Junipr (unary and binary) models trained and discussed in this dissertation are listed in Appendix C.2, table C.2.1. For each model, the test set amounts to 20% of the quark or gluon sample ($\sim 360,000$ jets if the whole sample is taken). For unary models, of this 20%, a set of 10 % of jets (thus 2% of the global quark or gluon samples, $\sim 36,000$ jets) is set aside for validation (such a small fraction is taken to avoid slowing training). The results presented here focus on a single model: a run on the data of set 4 ($E_{sub} = 1$ GeV), with a hidden state of size 30 updated by an LSTM cell, a single 10-unit hidden layer in every DNN mapping, a granularity of 10, the multi-branch implementation of the *daughter branch*, and the training schedule of table 4.2.1 for 6 epochs in total. The other models are introduced later in the text and should be assumed to have the same architecture unless specifically stated. This architecture was chosen as it corresponds to the one deployed in the binary Junipr paper [9]. All training curves are available in Appendix C.2.

Epoch	1	1	1	1	1	1
Learning Rate	10^{-2}	10^{-3}	10^{-4}	10^{-3}	10^{-4}	10^{-5}
Batch size	10			100		

Table 4.2.1 – Training schedule for unary Junipr.

This particular training schedule is adopted to help the training of the unary model. It is a shortened version of the one introduced in the original Junipr paper [8] as the dataset used here is larger (1.8M jets per label versus their 500k jets). This reduction was also applied in the paper introducing binary Junipr [9] for the first leg of training (this paper uses 800k jets). The main idea behind this schedule is to slowly decrease the learning rate to converge to the best optimum. When this optimum is sufficiently

approached, it increases the batch size to reduce variance in training and at the same time adopts a larger learning rate. The motivation behind this modification is to escape any local minimum that may have been reached and that could get the training stuck. The learning rate is then continuously decreased to the lowest value of 10^{-5} so that the model can optimise its parameters to get as close as possible to the desired extremum. Note that recent studies have shown that increasing the batch size is preferable to decreasing the learning rate [67, 68]. While the effect of a larger batch size is obviously beneficial in reducing the variance of a gradient computed with Stochastic Gradient Descent (SGD), it was also shown to improve the performance of Adam (letting it adapt its parameter more efficiently). Indeed, with the same number of epochs, similar results were obtained with batch schedules instead of learning rate schedules. The advantage of the former is that fewer parameter updates are required. The training schedule adopted here leverages these two approaches.

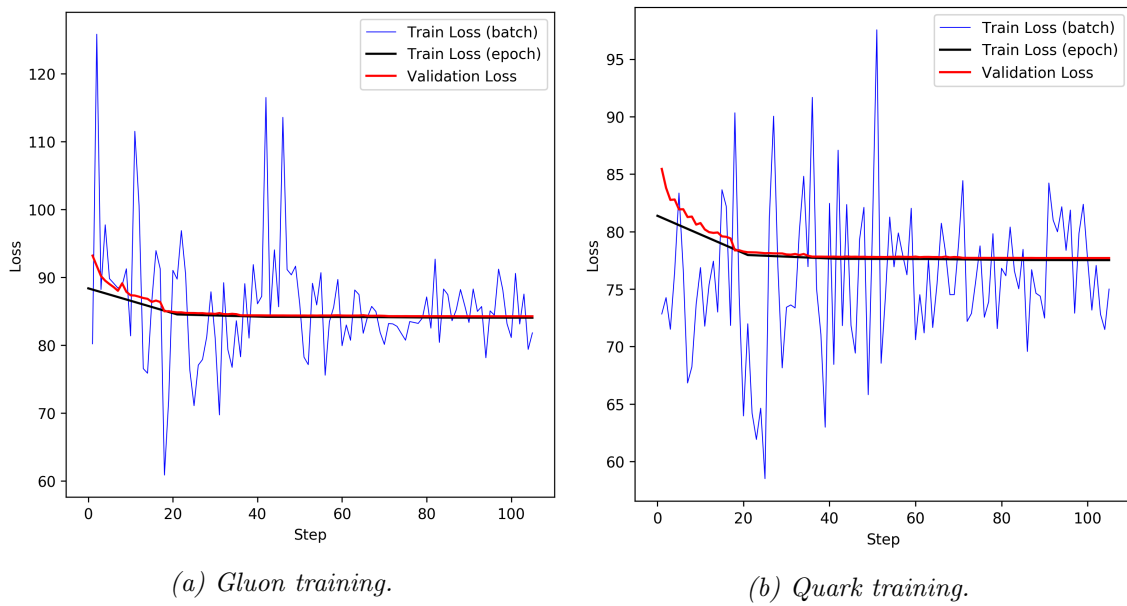


Figure 4.2.1 – The learning curves of the gluon and quark models trained for 6 epochs on set 4. The blue curve is the loss of a random batch at the step indicated, the black curve is the interpolated loss of the epoch, and the red curve is the loss on the validation set. One step corresponds to processing 80,000 jets.

The learning curves obtained by training two unary Junipr models with this configuration on set 4 are presented in figure 4.2.1. Two phases of training can easily be identified: the first part, with the small batch size of 10, has a lot of variance while the second part, starting

around step 55, smoothens the associated oscillations by using the larger batch size of 100. The validation and epoch-training curves both show the model is efficiently learning and stabilising at a specific loss value. To take further precaution against underfitting, the models were trained for 10 more epochs, with the extended training schedule of table 4.2.2 (thus training for 16 epochs). The objective of this extension is to let the model get as close as possible to the elected minimum, with a faster pace at first (larger learning rate) changing to a finer resolution after two epochs. The resulting learning curves are shown in figure 4.2.2 for the total 16 epochs of training (including the 6 epochs from the first leg of training). The quark model is referred to as *UJ Q1* and the gluon one as *UJ G1*.

Epoch	2	8
Learning Rate	10^{-4}	10^{-5}
Batch size	100	

Table 4.2.2 – Extension of the training schedule for unary Junipr. This schedule follows that presented in 4.2.1.

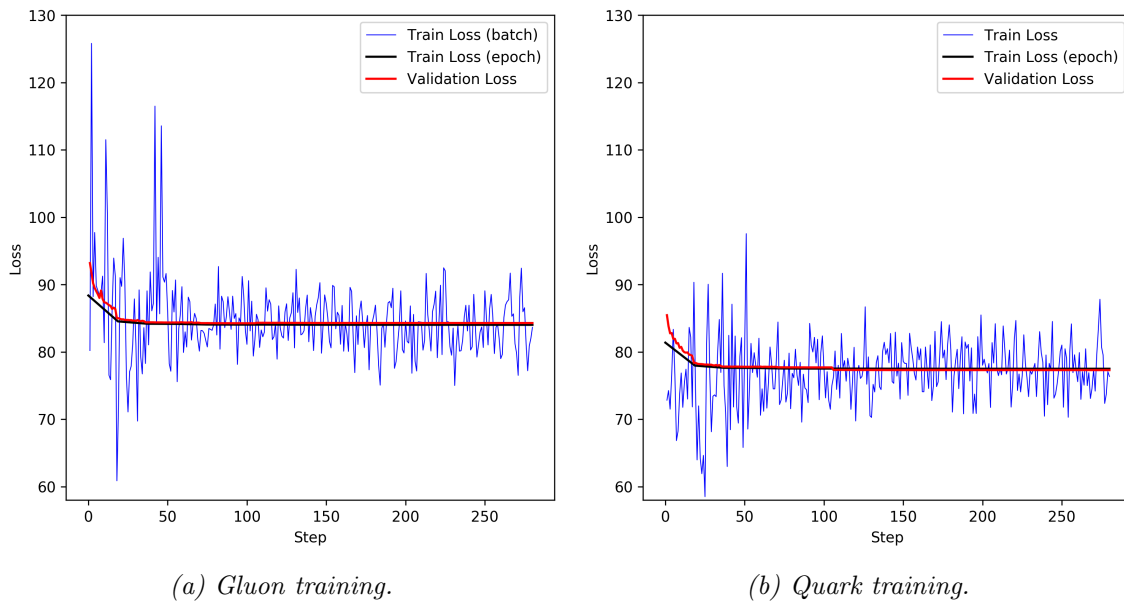


Figure 4.2.2 – The learning curves of the gluon and quark Junipr models trained for 16 epochs on set 4 (*UJ G1* and *UJ Q1*). The blue curve is the loss of a random batch at the step indicated, the black curve is the interpolated loss of the epoch, and the red curve is the loss on the validation set. One step corresponds to processing 80,000 jets.

An interesting element of comparison between the two models is the fact they do not converge to the same loss. Indeed, the quark model converges to a loss value of 77.5 while

the gluon model converges to 84.0. This difference is due to the underlying data: gluon jets tend to be made of more constituents, each carrying on average a lower fraction of the p_T of the jet compared to quark constituents, as revealed in figure 3.1.9 and previously discussed. A longer jet, made of more decays (nodes), will inherently have a lower probability given it has more parameters to fix: each node samples from three probability distributions, corresponding to the three branches. The gluon jet model is thus assigned a lower average probability, corresponding to a higher loss, simply because of the structure of its training data. This does not imply the gluon model is performing badly compared to the quark model. They are indeed equally sensitive to the effect of larger trees and, in the binary comparison, will both analyse the same data so that no bias is introduced.

A powerful attribute of the Junipr approach is presented in figure 4.2.3. It displays two jet trees as analysed by the unary Junipr quark model (the model trained on the quark-enriched sample). The nodes correspond to $1 \rightarrow 2$ decays and the edges are associated with states (for the last constituents, these are the topoclusters from which the jet history is reconstructed). Note that the horizontal distance between nodes is fixed to a given length, explaining why they align in columns. At each node of the tree, the value indicated corresponds to $-\ln(\text{probability of the node})$. The global probability of the tree,

$$-\ln(P(\text{tree})) = \sum_n -\ln(P(\text{node } n)) - \ln(P(\text{end after last node})),$$

is presented as *Probability* with the label of the jet being analysed (0 for gluon, 1 for quark). The probability that the tree does end after the last node is not represented directly (since it is not associated with a node), though it is included in the global value *Probability*. The smaller the value, the likelier the jet. These trees are displayed as two-dimensional representations of three-dimensional objects. The angle at each node corresponds to the daughter-to-mother angles in the associated decay, with the global orientation (whether the hard daughter goes up or down) taken by the relative orientation of the hard daughter to seed-momentum. The energy of the states is represented by a logarithmic colour scale. Note that this $3D \rightarrow 2D$ projection may sometimes lead to nodes violating the energy-momentum conservation rule *in appearance*. Some nodes may also give the impression to

have a single daughter. This occurs when both daughters are very forward (close to the direction of the mother): the plotting algorithm then draws one on top of the other.

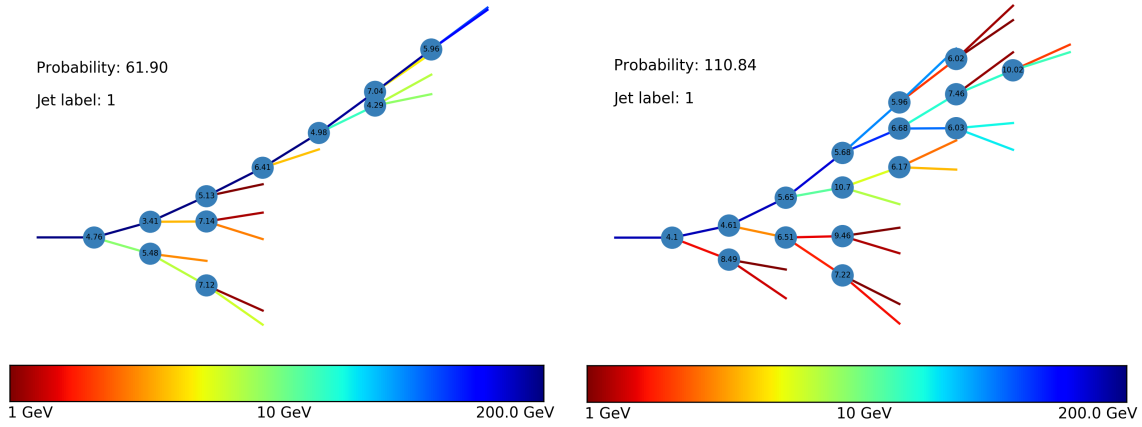


Figure 4.2.3 – Quark jets analysed by the unary Junipr quark model trained for 16 epochs on set 4.

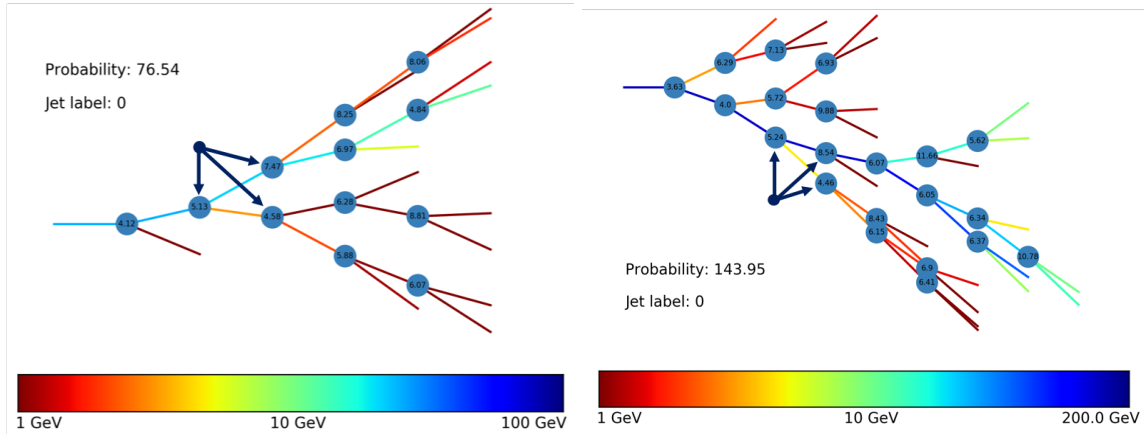


Figure 4.2.4 – Gluon jets analysed by the unary Junipr gluon model trained for 16 epochs on set 4. The dark blue arrows point to nodes discussed in the text.

This representation of jets data condenses a lot of low-level information into a physically interpretable picture. It corresponds to a phenomenological expansion of a jet probability conditioned on a parton label as structured by a specific factorisation algorithm. Some interesting patterns are visible in these trees:

- The nodes at the leaves tend to have higher values, meaning lower probabilities. This is very much expected: at lower-energies, there is more dynamical freedom for the configuration of the daughters and there are more candidate mothers to decay.

- The root nodes tend to have lower values, meaning higher probabilities. This is also expected as the data is centralised and a root node must be present in every jet.
- Considering the additional jets plotted in figure C.4.1 of Appendix C.4, an interesting difference of pattern can be uncovered. Quark jets tend to have a path of highly probable nodes along the most energetic branches. Indeed, the two plots of figure 4.2.3 have the most probable nodes all connected along the same blue path (indicating the highest energy). This behaviour breaks down only at the 6th column of nodes in the trees. On the contrary, gluon trees do not exhibit such a pattern. Take the two plots of figure 4.2.4. Note how, for the left one, after the second decay it is the lower orange branch that leads to the more probable node instead of the higher energy blue one above, as indicated by the dark blue arrows. The same observation can be made on the right figure, after the third bottom node since the yellow branch leads to a more probable value. The quark model therefore assigns higher probabilities to high energy decays than the gluon one. This echoes the discussion linked to splitting function at the end of section 3.1.3: quark jet constituents are more likely to carry a higher fraction of the momentum.
- Finally, as previously mentioned, the probability of a jet tree decreases with its size. This is an expected effect as a larger tree samples from a larger global probability distribution, having more variables to fix.

To assess the quality of the probability models learnt by unary Junipr models, the probability distributions of the variables sampled at each node are computed and compared to the underlying distributions of the input data. This is done in two steps:

1. Assess the probability model of Junipr. During the forward pass of the testing phase, Junipr outputs a probability corresponding to each observable value. The ending branch outputs the probability that the tree ends at the node, the daughter branch outputs a probability associated to each bin of z , θ , ϕ , and δ , and the mother branch outputs the probability for each possible mother index (at timestep t , as many as t possible mothers are considered) to be the one decaying at the node. For each node of each tree, these probabilities are accumulated into histograms and then normalised,

thereby revealing the probability distributions over possible predictions.

2. A similar approach is taken with the Monte Carlo simulations: a histogram of the recorded information is established and normalised to obtain the underlying probability distribution.

Figures 4.2.5, 4.2.6, and 4.2.7 display this information for the unary Junipr gluon and quark models, respectively in the left and right columns. Each row corresponds to one of the 6 variables considered. In each plot, two distributions are presented: the Junipr distribution is in orange and the simulations (Monte Carlo) in blue. When the two signals overlap, the result is displayed in a brownish colour. Most of the plots appear in this mixed brown colour, indicating an excellent agreement between the probability distributions of the simulations and the Junipr models. It is in fact difficult to find any part of the distributions that do not overlap (zooming at the top of some bin sometimes indicate a fine layer of excess from one of these signals). A worrisome fact is displayed in this set of plots when comparing the two columns: the quark and gluon distributions match extremely well. Indeed, the differences between the two models are very thin and can be summarised as:

1. State length distributions (top of figure 4.2.5): the gluon distribution has a longer right tail while the quark sample has a more protruding peak around 13. This implies that quark tends on average to have a lesser number of constituents than gluons, a conclusion supported by figure 3.1.9. Note how the Junipr models were able to adapt their predictions to the cut removing jets with less than 5 constituents: almost no probability weight is given to state length values below 5.
2. Mother index in energy sorted list (bottom of figure 4.2.5): the first bin of the quark distribution is higher than the corresponding one for the gluons. Conversely, the last bins are more populated for the gluons, indicating a slight tendency for gluon mothers to be less energetic than other possible candidates at their timestep compared to quark mother.
3. Distribution of θ (top of figure 4.2.6): both distributions are very close, though the gluon one is slightly more forward. This supports the point made in the previous

chapter that gluon jets tend to be wider.

4. Distribution of δ (bottom of figure 4.2.6): the variable takes larger values on average for gluons than for quarks (note how in particular the peak around 0.02 is more protruding for gluons). This also supports the point that gluon jets tend to be wider.
5. Distribution of z (top of figure 4.2.7): interestingly, the quark distribution has stronger first and second bins, indicating more events with a low fraction of energy going to the soft daughter. This implies that daughters in quark trees are more likely to have a larger energy difference than in gluon trees. Conversely, the gluon distribution has a more distinct peak at larger z values, indicating a fairer split of the available energy. This resonates with the conclusion that quark jets tend to have more high p_T constituents (associated with the hard daughters).
6. Distribution of ϕ (bottom of figure 4.2.7): both distributions are close to being uniform and are very similar. Note that the slight excess for values of ϕ close to π and $0/2\pi$ is probably due to the method employed to reconstruct the angles. There is indeed some numerical instabilities in computing angles that approach these values. For example, the $\cos^{-1}(0.9999) = 1.41 \cdot 10^{-2}$ while $\cos^{-1}(0.999999999) = 4.47 \cdot 10^{-5}$. The trouble is that the method employed here to reconstruct the jet history, based on PyJet, has a number of significant digits after the decimal point limited to 12. All angles are therefore reconstructed with a number of significant digits after the decimal point of 8. Some controls have been added to the routine computing angles such that situations where $\cos \theta - 1.0 < 10^{-12}$ or $\cos \theta + 1.0 < 10^{-12}$ output, respectively, 0 and π . This was added in particular to avoid the nasty situation where the angle between a constituent and itself is non-zero. Since both the Junipr models and simulation have this bias, it does not introduce a significant error in the model (note that this bias is also present in the GitHub results of [9]).

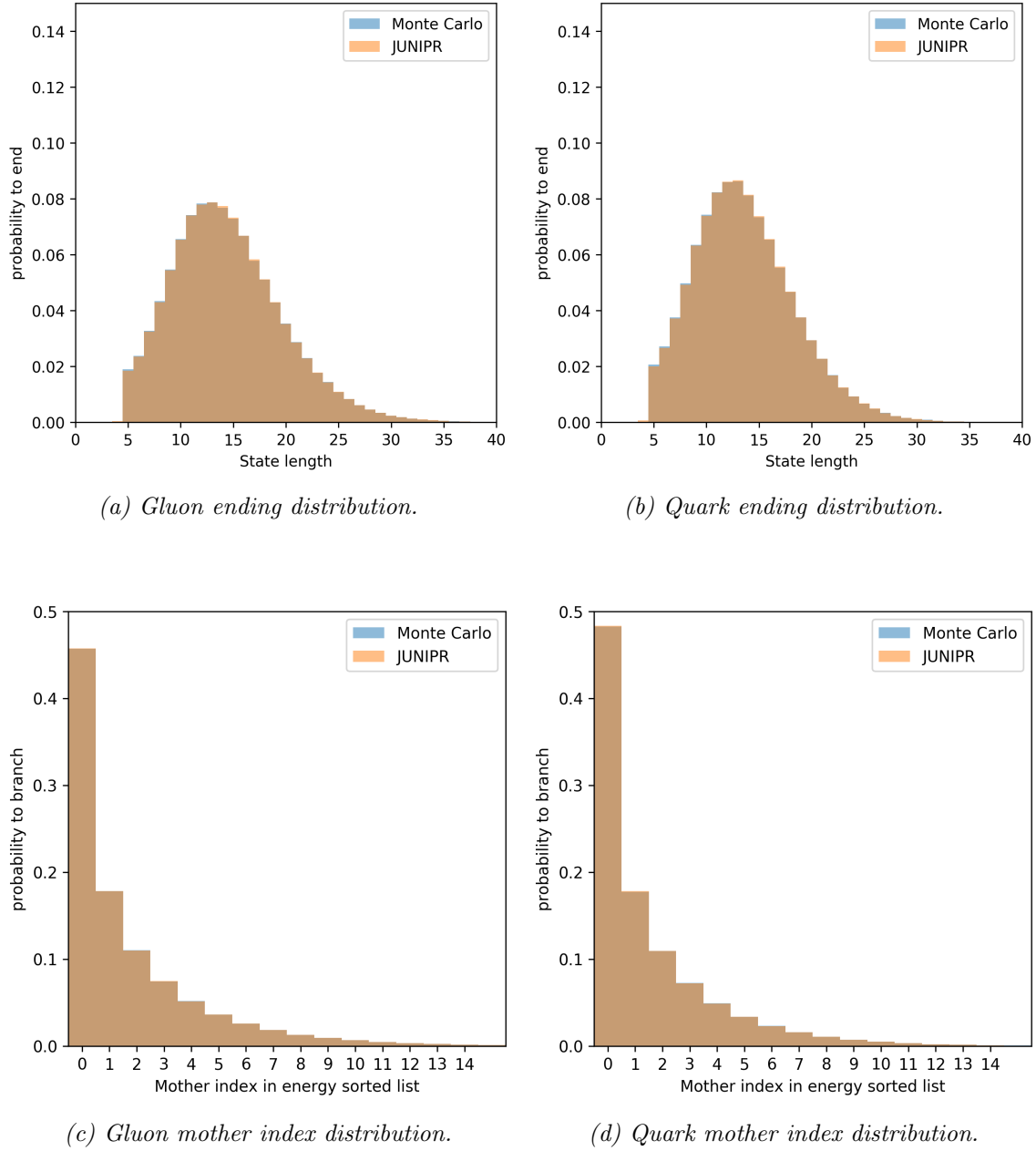


Figure 4.2.5 – Probability distributions of the state length (top) and mother index (bottom) for the gluon (left) and quark (right) models. Junipr probability in orange versus the probability distributions from simulations in blue. Overlapping distributions are shown in brown.

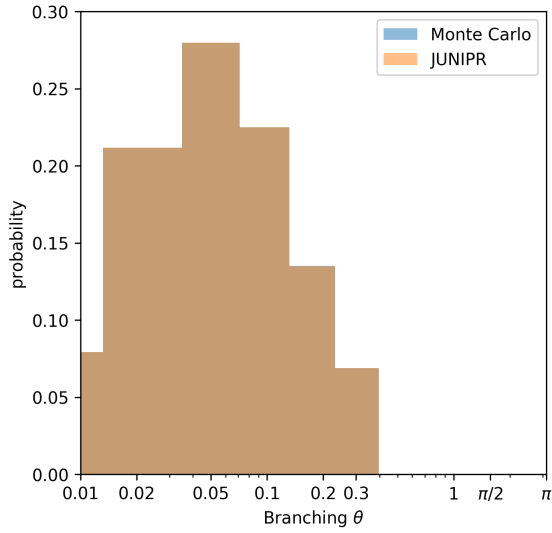
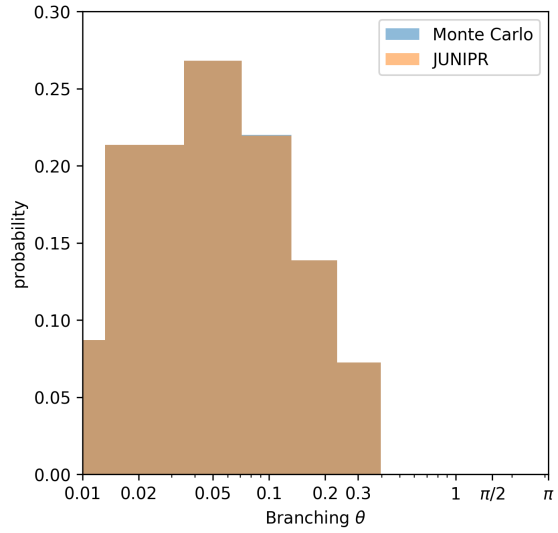
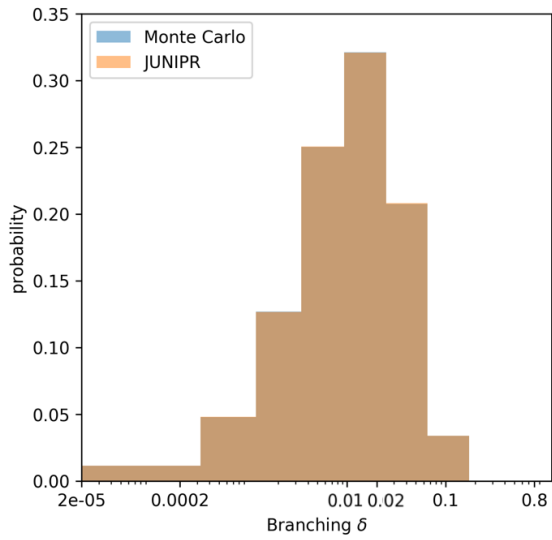
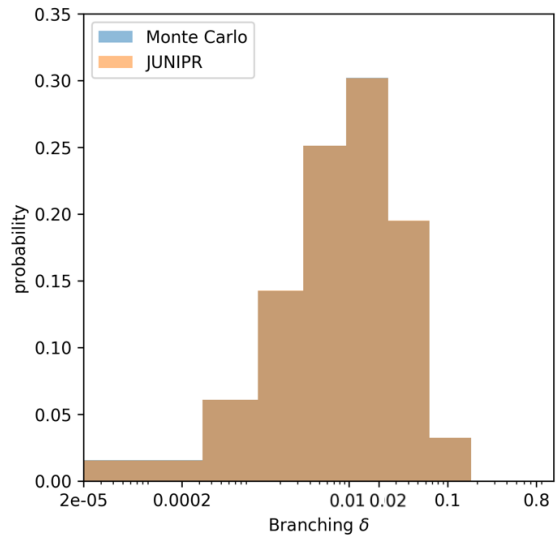
(a) Gluon θ distribution.(b) Quark θ distribution.(c) Gluon δ distribution.(d) Quark δ distribution.

Figure 4.2.6 – Probability distributions of θ (top) and δ (bottom) for the gluon (left) and quark (right) models. Junipr probability in orange versus the probability distributions from simulations in blue. Overlapping distributions are shown in brown.

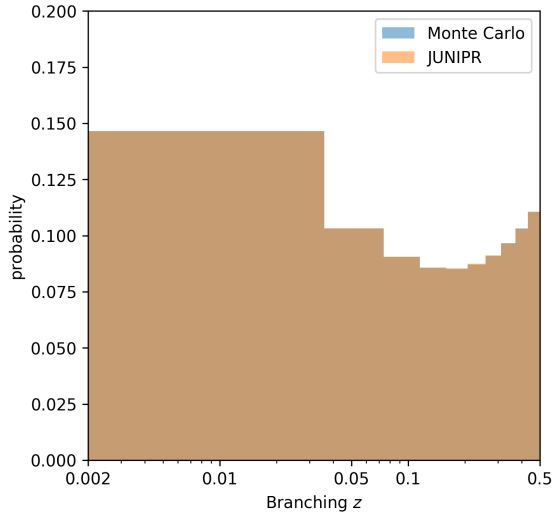
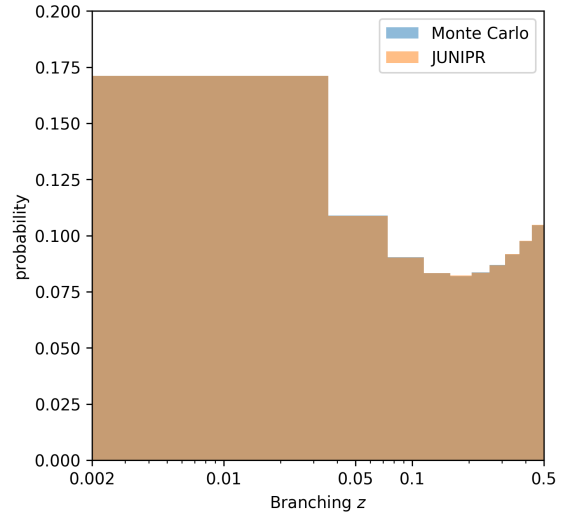
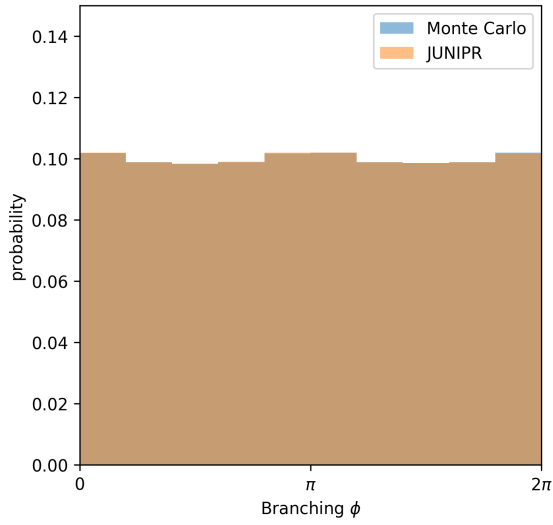
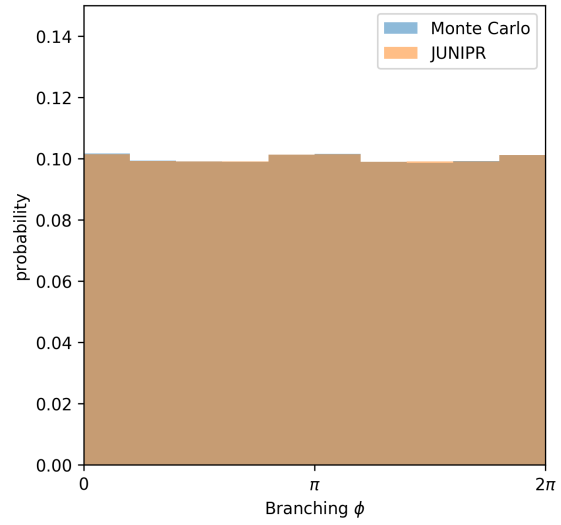
(a) Gluon z distribution.(b) Quark z distribution.(c) Gluon ϕ distribution.(d) Quark ϕ distribution.

Figure 4.2.7 – Probability distributions of z (top) and ϕ (bottom) for the gluon (left) and quark (right) models. Junipr probability in orange versus the probability distributions from simulations in blue. Overlapping distributions are shown in brown.

4.3 Binary Junipr

Having a pair of unary Junipr models trained on a quark and gluon datasets respectively, both models can be joined in a training loop optimising the binary objective of equation 3.2.9. The procedure consists in running two unary Junipr models in parallel, collect the likelihoods they output for a batch of jets, take the ratio of the probabilities of the quark model to that of the gluon one (in log-form, so the difference of the log-probabilities), apply a sigmoid, and finally compute the cross-entropy with the truth labels (0 or 1). This returns the loss to minimise with the Adam optimiser, leading the training to make a gradient step and adapt accordingly the parameters of both models for the specific task of discrimination. The training follows the schedule proposed in table 4.3.1, as adopted in the binary Junipr paper [9]. It fully leverages the idea of increasing the batch size rather than the learning rate. Attempts at introducing a learning rate schedule either did not improve training, slowing it down, or altogether led to non-optimal extrema.

Epoch	1	5	10	3
Learning Rate	10^{-3}			
Batch size	10	100	1000	2000

Table 4.3.1 – The 19 epochs training schedule for binary Junipr.

Taking the two 16 epochs unary models trained on set 4 (*UJ Q1* and *UJ G1*), as described in the previous section, figure 4.3.1 shows the learning curve when submitting them to binary training for the 19 epochs schedule. The resulting binary model is referred to as *BJ 1*. The dataset used here is the entire set 4, with 20% set aside for testing (~ 720 k jets) of which 5% is taken for the validation set (~ 36 k jets), the latter corresponding to a smaller fraction than in the unary case to reduce drag on training (though, in absolute numbers, it is equivalent).

In the right side of the figure, the accuracy measures the number of labels correctly predicted when thresholding the outputs of binary Junipr at the value 0.5. The steps at which the batch size changes are easily observable on the learning curves, as evidenced by the sudden discontinuities in the oscillating curves before they stabilise around a new

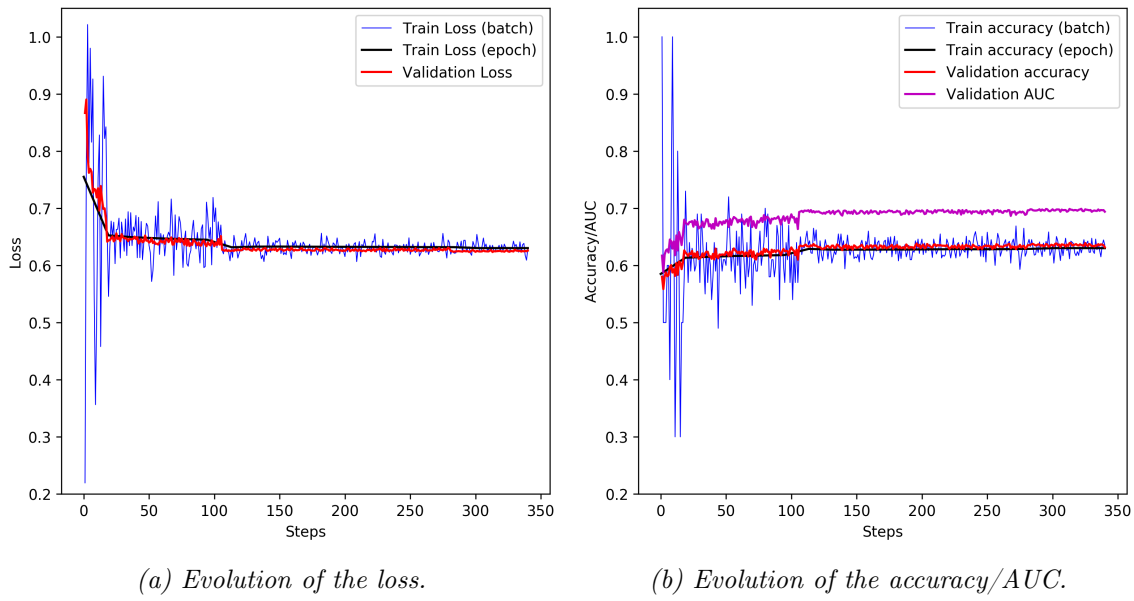


Figure 4.3.1 – The learning curves of the 19 epochs binary Junipr model (BJ 1) trained with the 16 epochs unary models on set 4 (UJ Q1 and UJ G1). The blue curve is the lost/accuracy of a random batch at the step indicated, the black curve is the lost/accuracy of the epoch, the red curve is the same information on the validation set, and the magenta curve is the AUC evaluated on the validation set. One step corresponds to processing 160,000 jets.

value. The binary model seems to have reached saturation at a training loss of about 0.63, corresponding to an accuracy of 0.63. Note that the initial increase of the accuracy curve in the right plot does indeed suggest that a pair of unary models is not directly as efficient as a binary Junipr model for discrimination. The weights of both unary models were modified to improve the accuracy, as evidenced by the increasing AUC of the ROC.

Figure 4.3.2 displays the scores obtained in classification. The score is the value of

$$\text{sigmoid} [\ln(P(\text{Jet} \mid \text{Quark Model})) - \ln(P(\text{Jet} \mid \text{Gluon Model}))].$$

This value is to be thresholded to return a label. Unfortunately, it seems the two distributions are not as separated as one might hope. Discriminating based on this score will therefore lead to a low accuracy. Although a large part of the distribution overlaps, two peaks do stand out. Interestingly, the gluon jets (in orange) show a secondary peak around 0.03, indicating a subset of gluon jets is correctly labelled with high confidence (very low score).

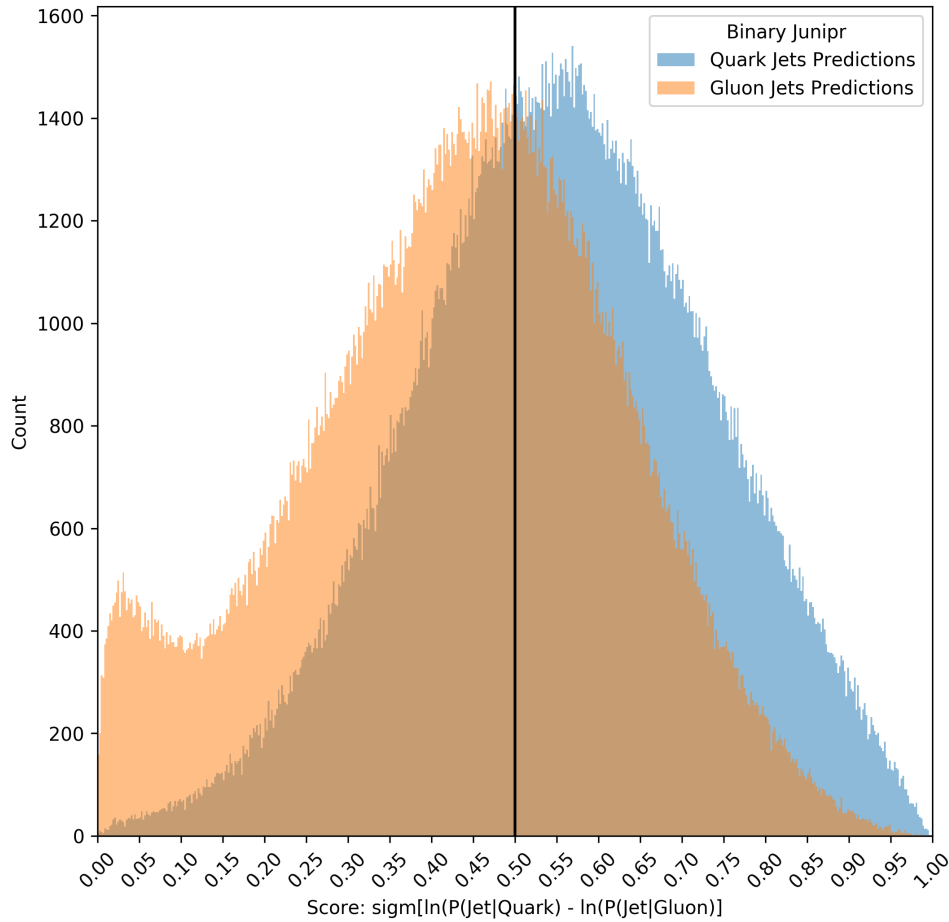


Figure 4.3.2 – The score distribution of the 19 epochs binary Junipr model on the test set of set 4. The score of jets with a gluon truth label is shown in orange and those with a quark truth label in blue, with overlapping in brown. The vertical black line indicates the 0.5 value of the score.

Binary Junipr also offers the powerful interpreting tools of jet trees, as exemplified in figure 4.3.3. These trees are built using the same method as the unary trees. In this context however, the values indicated in the nodes correspond to the log-likelihood ratio $\ln(P(\text{Jet} | \text{Quark Model})) - \ln(P(\text{Jet} | \text{Gluon Model}))$, with the global ratio indicated as *Probability*. Again, the last ratio, concerned with the probability for the tree to end after the last node, is not shown though it is contained in *Probability*. Any value above 0 suggests a quark-jet behaviour, while values below 0 give evidence of a gluon initial parton. A value of 0 acknowledges a lack of preference from the model between the two hypotheses (it equates to a likelihood-ratio test of 1).

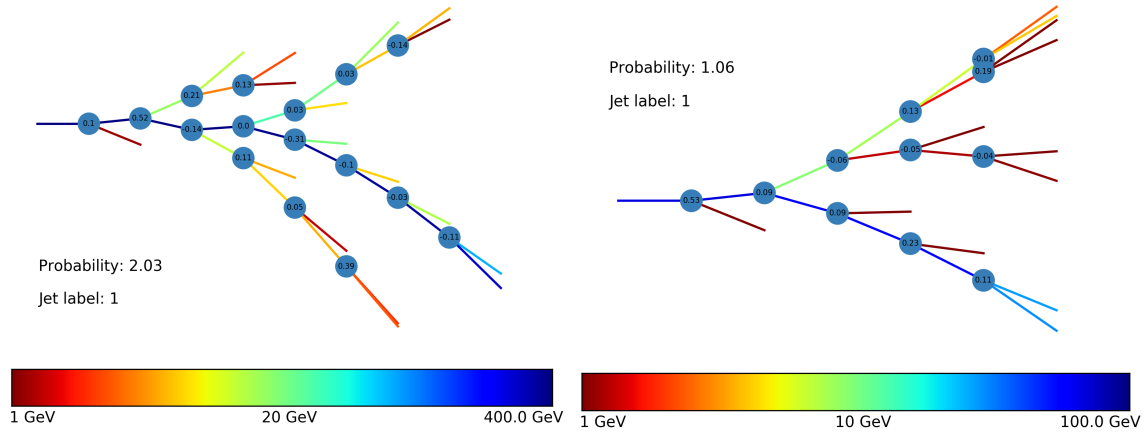


Figure 4.3.3 – Quark jets from set 4 analysed by the binary Junipr model BJ 1.

4.4 Comparison of Binary Junipr to the BDT and DNN Benchmarks

This section compares the results of binary Junipr to the benchmarking BDT and DNN. To achieve this objective, a BDT and a DNN are trained on set 6 (the variables exploited are listed in table 4.1.2 - the DNN is trained for 20 epochs) using the architecture found to be optimal on set 2 and 3, as described in section 4.1. Note that other architectures were explored but showed no improvement on this dataset. The result is summarised in figure 4.4.1, displaying the ROC curves of the three models as well as their AUC's. Clearly, binary Junipr delivers a lower performance than the two other models, the BDT and DNN both leading in terms of efficiency.

To further analyse the results of the BDT and DNN, figure 4.4.2 presents their score distributions. Beware of the different range of the support of the BDT score. Qualitatively, all score distributions indicate a secondary peak for the gluon jets (for the BDT, it is shown as a long left tail). An interesting difference, however, is that these two models have a stronger gluon peak than quark one, while the binary Junipr model (*BJ 1*) had the opposite (shown in figure 4.3.2). Additionally, there is a clearer separation of the gluon and quark scores, particularly visible for the DNN distribution, as confirmed in figure 4.4.3 displaying the ratio of the normalised score distribution DNN/binary Junipr. Indeed, a blue excess (above 1) is visible for quarks (high scores) and an orange one for gluons (low score). The combination of these attributes explains the better efficiencies of the BDT and DNN.

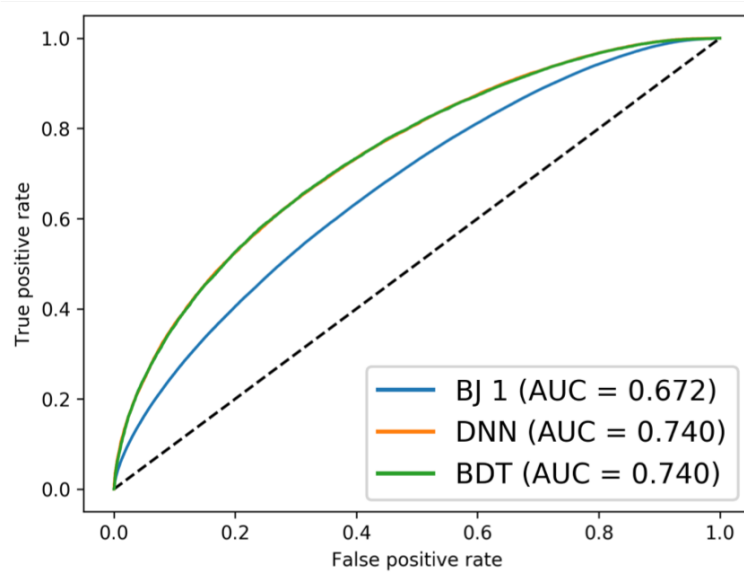


Figure 4.4.1 – The ROC curves of the 19 epochs binary Junipr model (BJ 1, in blue), the optimal BDT (green), and the optimal DNN (orange, trained for 20 epochs) on set 4 and set 6 for the last two models. The BDT and DNN curves are superimposed given their similar efficiency (this hides the orange curve).

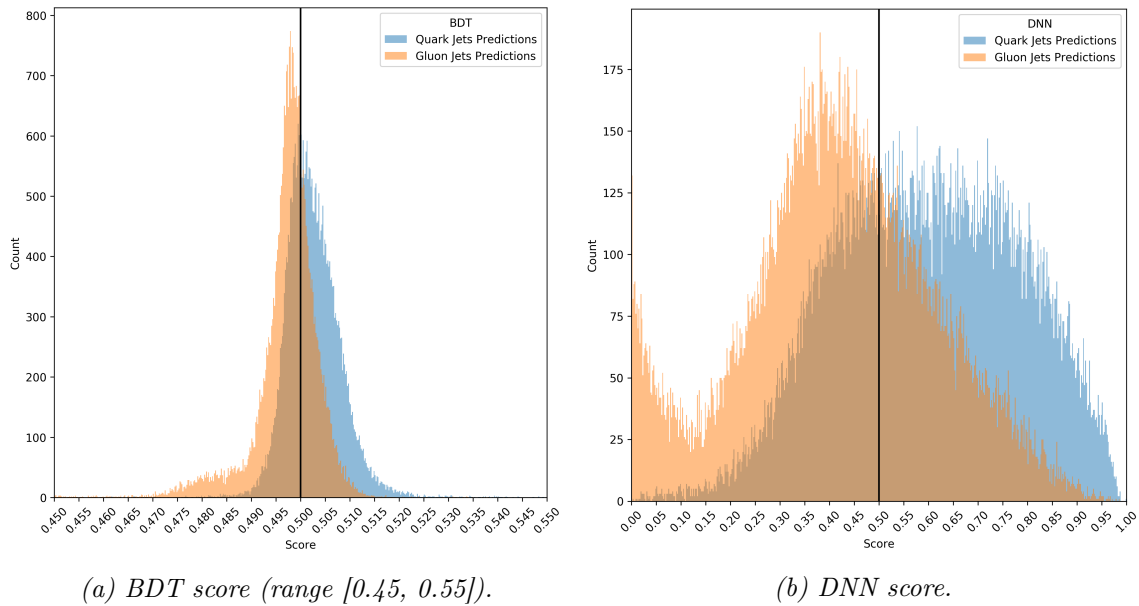


Figure 4.4.2 – The score distribution of the BDT (left) and DNN (right, trained for 20 epochs) models on set 6. In this case, the score is the output of the BDT or the DNN. The score of jets with a gluon truth label is shown in orange and those with a quark truth label in blue, with overlapping in brown. The vertical black line indicates the 0.5 value of the score.

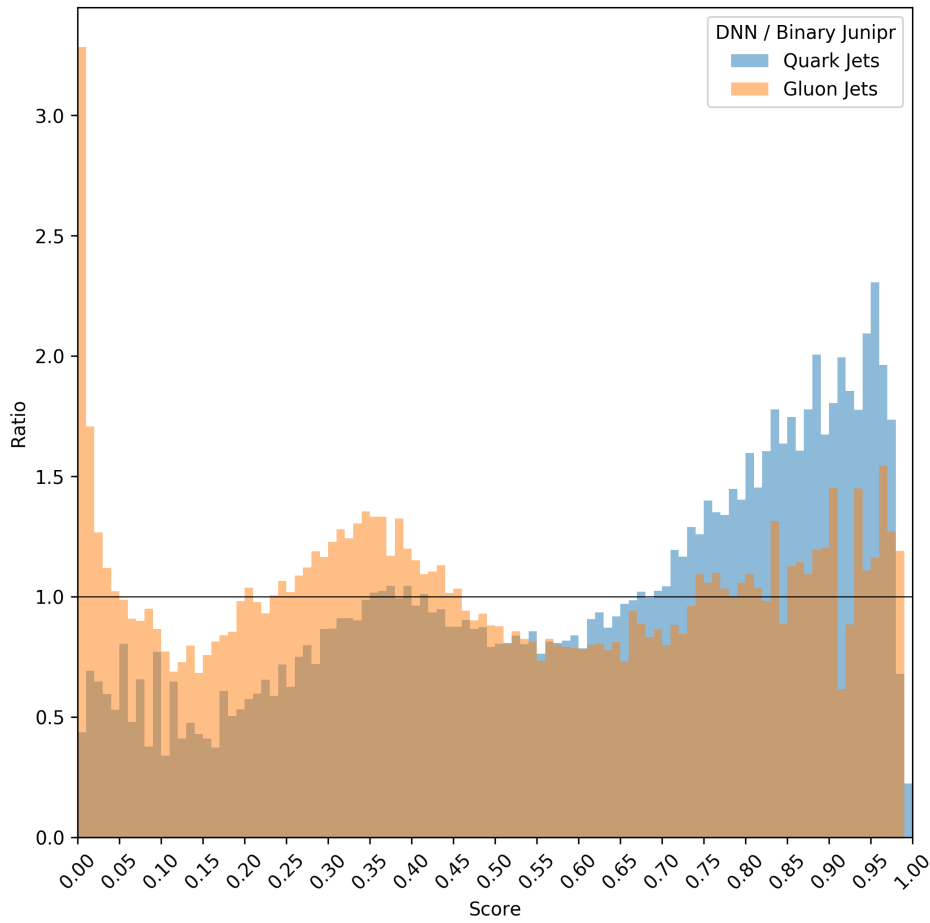


Figure 4.4.3 – The ratio DNN/binary Junipr of normalised score distribution for set 4, gluon jets in orange and quark jets in blue. It is the ratio of the distribution of the right plot of figure 4.4.2 on that of figure 4.3.2.

Could the efficiency of binary Junipr suffer from the cut removing constituents with an energy below 1 GeV, as is done for set 4? To explore this question, set 5 was produced with a cut removing only those constituents with an energy below 0.5 GeV. The motivation behind this lower value is to keep more of the low-energy final state constituents. These are indeed likely arising from soft radiations that may have a significant impact on the width of the measured jet, a relevant piece of information for discrimination. The change of cut is observed to add only but a small fraction of jets (below 0.001 %) so no new HDF5 dataset was produced and the BDT and DNN models presented so far are kept as benchmarks. Indeed, the only way this cut accepts an otherwise rejected jet is if this jet had less than five constituents under the 1 GeV cut and now has at least five of them after the 0.5 GeV cut. This is unlikely, given the other cuts and the properties of the samples. A binary

Junipr model was trained for 26 epochs on this dataset (set 5), with 7 epochs added to the last leg (batch size 2000) of the training schedule of table 4.3.1 to form this extended schedule. This binary model is referred to as *BJ 2* and its quark and gluon unary models as *UJ Q2* and *UJ G2*, respectively. Appendix C.2 shows both the training curves of the unary models (figure C.2.1) and the learning curves of this binary model (figure C.2.2). The motivation behind the extension of the training schedule is as a precaution to avoid under-training the model. The ROC curve of this model is displayed against the previously obtained ones in figure 4.4.4.

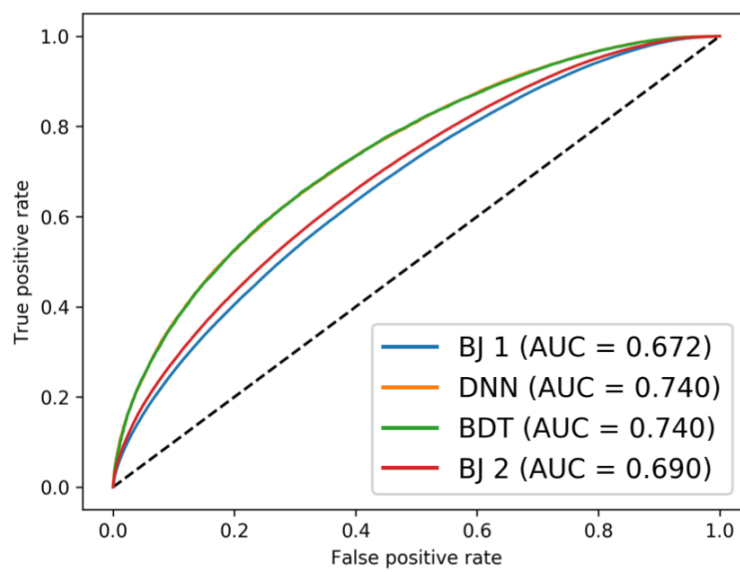


Figure 4.4.4 – The ROC curves of the three previous models (*BJ 1* in blue, *BDT* in green, and the *DNN* in orange) trained on sets 4 and 6 versus that of *BJ 2* (in red), trained on set 5.

Evidently, keeping more of the soft radiations helps the binary Junipr approach in the task of discriminating quark and gluon jets, as suggested by the larger AUC of *BJ 2* compared to the *BJ 1* binary model. It is however not sufficient to overpass the efficiencies of both the BDT and the DNN. This might be expected, as the Junipr models work on much lower-level information. Particle physicists have indeed worked hard to construct jet variables that carry most of the meaningful information in an event. Comparing the results of an approach combining calorimeter topoclusters momenta to another one using the full power of the detector and the underlying Physics is perhaps too ambitious. However, there is a subtlety in the Junipr approach that can help explain this difference in efficiency.

Indeed, the results shown here for binary Junipr are far lower than the claimed efficiency in the paper introducing the method [9]. Working at the generated level, the authors obtain an AUC of 0.8986, a state-of-the-art result. Their study uses a dataset produced in [32], adopting the following setup:

- The processes simulated with Pythia 8.226 are $Z(\rightarrow \nu\bar{\nu}) + g$ and $Z(\rightarrow \nu\bar{\nu}) + (u, d, s)$: a Z boson decaying to a pair of neutrinos (this signature is invisible: neutrinos pass undetected in the sort of detector considered here) and either a gluon or an up, down, or strange quark.
- Hadronisation and underlying event processes are added, though no PU nor detector effect are considered. Hence their dataset is at the generated level. Jets were reclustered with anti- k_T (radius = 0.4) and analysed with C-A (as is done here).
- The phase space configuration is limited to jets with $p_T \in [500, 550]$ GeV and a rapidity $|y| < 2.0$.

While the different levels of the variables, of simulations, and the different processes studied compared to this project might play a role, the crucial difference stems in fact from the last point. Though the rapidity cut is of no significant concern here (the data is just restricted to what is observable in the ATLAS detector), the p_T cut implies a restriction to much higher energies than adopted in this project. As suggested in figure 3.1.7, most of the jets tend to have an energy below 100 GeV (the median is ~ 75 GeV). With a higher transverse momentum, one can expect a higher energy (transverse momentum, contributing to the kinetic energy, is a component of the total energy). Higher energies inevitably lead to larger jet trees, as more decays are required to bring the initial and intermediate elements to stable forms. A larger tree leaves more room for Junipr to output the correct global likelihood ratio test. It decreases the importance of each individual node and thereby reduces the bias introduced by poorly estimated node probabilities.

Therefore, it is logical to expect the Junipr approach to strongly correlate with the energy of the jet samples. To explore this avenue, figure 4.4.5 displays the ROC curve obtained by model *BJ 2* on different bins of energy.

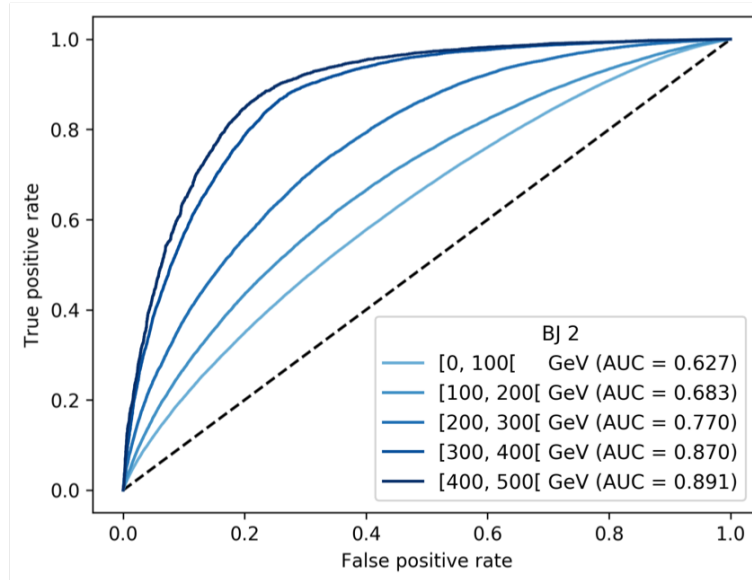


Figure 4.4.5 – The ROC curves obtained by assessing BJ 2 on different energy bins of set 5.

The impact of changing the energy bin is remarkable. The AUC continuously increases as the average energy of the analysed jets grows. The result obtained in the original paper is even closely approached by the best result of BJ 2, reaching an AUC of 0.891 in the [400, 500[GeV region (note that the open bracket on the right side is equivalent to [400, 500) and means the right border is not included). Not shown on the plot, the AUC of the [500, 550[GeV bin is of 0.887 (this energy bin corresponds to that selected in the binary Junipr paper). This result is lower than that obtained on the previous bin, an important observation that is addressed later in this chapter. Note that it is not only the accuracy of the Junipr approach that improves with energy: the label predictions themselves also become more reliable. Indeed, as shown in figure 4.4.6, the output score of quark and gluon jets separate to their respective confidently-predicted regions when going to higher energy: the global output of the binary Junipr model becomes more reliable. None of these effects are discussed in the Junipr papers [8, 9].

Note that the previous point is an inherent property of the Junipr model. It is not an artefact of the scarcity of data degrading the result at low energy (jets below 100 GeV are the most common) nor is this behaviour stemming from specificities of the learning procedure. To account for the latter effect, a pair of unary Junipr models (*UJ Q3* and *UJ G3*) were trained specifically on the [100, 200[GeV energy bin of set 5 for 50 epochs,

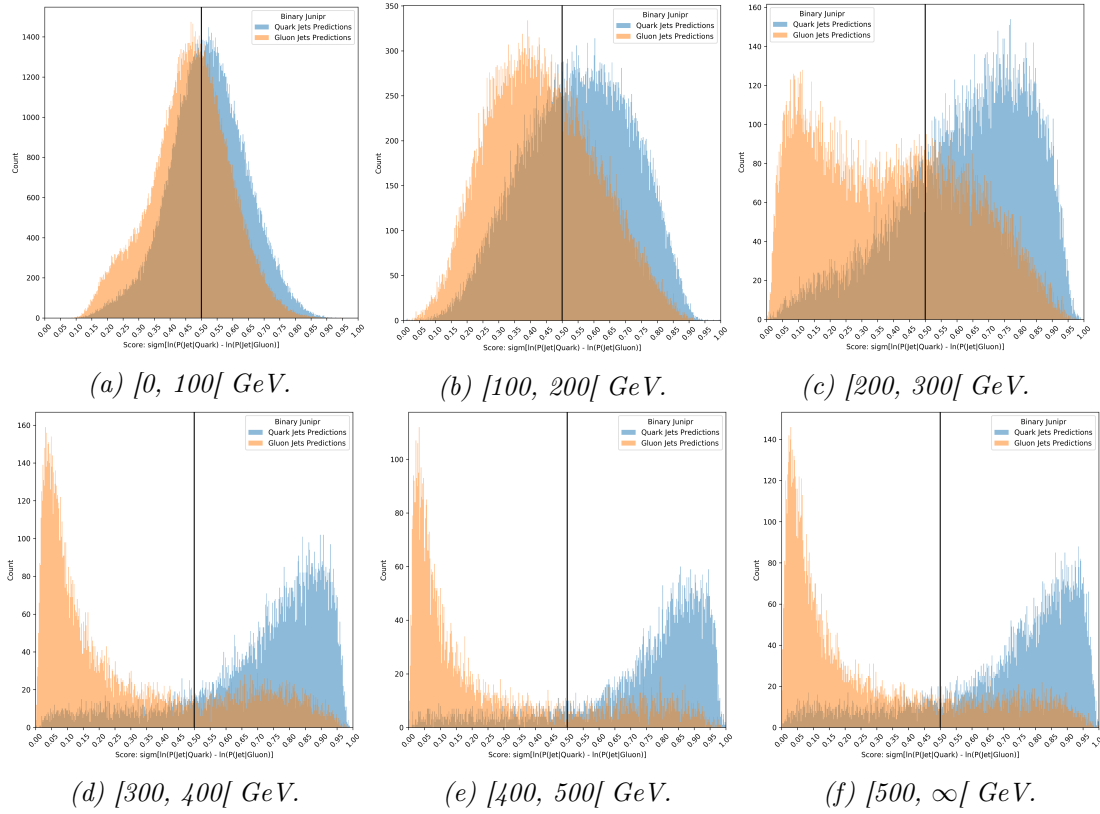


Figure 4.4.6 – The score distributions of *BJ 2* for quark (blue) and gluon (orange) jets for different energy bins of set 5 (as indicated under each plot).

following the long extended schedule of table 4.4.1. This schedule is adopted as this energy bin is far less populated than the global jet samples of set 5, containing $\sim 347k$ jets of each type. A binary Junipr model, *BJ 3*, was trained with these two unaries for 14 epochs, skipping the last 5 epochs of the training schedule of table 4.3.1 (the model was observed to have its efficiency saturate early enough, as suggested by the learning curves displayed in Appendix C.2).

Epoch	5	10	10	10	10	5
Learning Rate	10^{-2}	10^{-3}	10^{-4}	10^{-3}	10^{-4}	10^{-5}
Batch size	10			100		

Table 4.4.1 – The long extended training schedule for unary Junipr models *UJ Q3* and *UJ G3*.

The ROC curve obtained from the model thus trained is displayed in figure 4.4.7. The performance delivered by *BJ 3*, specifically trained on this energy bin, is inferior to that obtained by *BJ 2*. It can only be caused by the lack of sufficient data for the architecture

to fit the specific requirements of such jets. Indeed, the training curves (figures C.2.3 and C.2.4) show the models involved in *BJ 3* reached saturation and the architectures of both Junipr models are strictly the same. The only differences come from the different amount of data and the restricted energy window for training. This result suggests binary Junipr is capable of transferring learning done in other energy regions: this is expected given the elementary items used in discrimination are the nodes and those display a similar structure regardless of the energy.

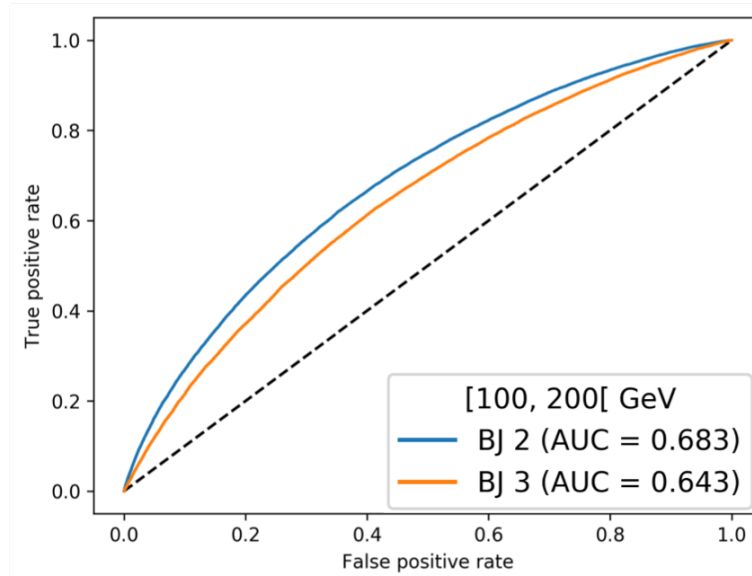


Figure 4.4.7 – The ROC curves obtained by assessing *BJ 2* (trained on whole energy range) and *BJ 3* (trained on [100, 200[GeV range) on the [100, 200[GeV energy bin of set 5.

All the main conclusions of this project are reflected in figure 4.4.8. It shows that the efficiencies of all models considered (binary Junipr, BDT, and DNN) are strongly correlated with the energy of the jet samples they analyse. For each model, the AUC values are computed in successive bins of increasing jet energy (from their respective datasets). These bins are adapted to contain enough statistics for stable AUC computation: their size slightly increases with the energy since the datasets are depleting for higher values of this variable. As indicated by the dashed curves, plotting the normalised distributions of number of jets per energy bin, the growth of the bin size does not fully compensate the increasing scarcity of jets with energy but stabilises it. This figure also unveils the performance of a fourth binary Junipr model, named *BJ 4*, that is trained on set 5 with a larger architecture (the hidden state size was grown from 30 to 40 and each DNN hidden

layer is given 20 units instead of 10). It uses 16 epochs unary models, $UJ\ Q4$ and $UJ\ G4$, with the same training schedule as $UJ\ Q2$ and $UJ\ G2$. The binary training also followed the 26 epochs schedule of $BJ\ 2$ (more details on the training of this model can be found in Appendix C.2).

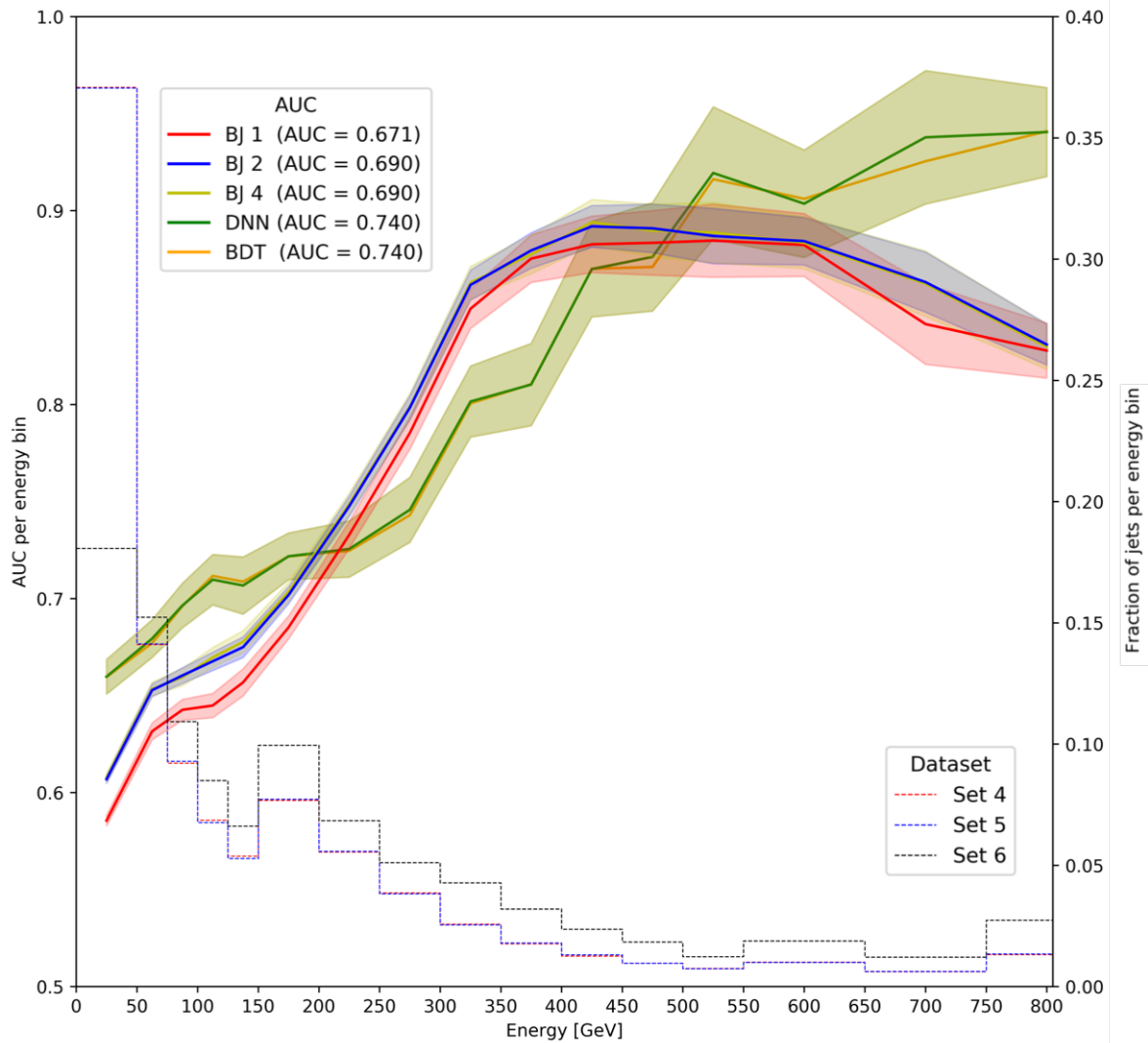


Figure 4.4.8 – The AUC of the binary Junipr models $BJ\ 1$, $BJ\ 2$, and $BJ\ 4$ versus the benchmarking BDT and DNN models as a function of energy. These models are respectively trained and tested on set 4, set 5, set 5 and set 6 for the two benchmarks. Continuous lines of different colours are used to describe the evolution of the AUC in increasing energy bins (left ordinate axis). The last bin contains all jets with an energy above 750 GeV. These energy bins are represented by the dashed stepping curves, presenting the fraction of jets in set 4 (red), set 5 (blue), and set 6 (black) in each bin so that each distribution of jets per energy bin is normalised to unity (right ordinate axis). The shading around the AUC curves is a rough estimation of the statistical incertitude. The limits of this area for a given model correspond to $AUC \pm 1/\sqrt{\text{Number of jets in the bin}}$. Note that table C.2.1 summarises the differences between the Junipr models considered.

Four important conclusions are supported by this figure:

1. The efficiencies of discriminative models increase with energy, up to a certain point. This, as defended in section 3.1.3, cannot arise from differences in the energy distributions of the quark and gluon samples as these distributions are matched. It can be interpreted as a consequence of the Physics involved in the processes and the act of detection. Differences emerge with energy that the methods exploit to improve their discriminative power. The information gathered from the energy cannot be extracted in the same fashion by Junipr and engineered variable-based models.
 - A Junipr model improves its efficiency by reducing the bias introduced by poorly reconstructed nodes. This is further discussed later in the chapter.
 - A model using engineered features exploits the gain in accuracy from some of its variables with higher energies. The precision of the detector is indeed strongly correlated with the energy of the particles passing through its components. Furthermore, the distributions of the variables of the two processes follow a distinct evolution with the energy, simplifying discrimination. The BDT and DNN adapt their use of these high-level features based on the energy. This is suggested by analysing the feature importances of the BDT model, as presented in figure 4.4.9. Since the energy distributions are matched for the two samples, the only importance the jet energy can bring is in adapting cuts on the other distributions. This effect is visibly essential, the energy being presented as the most important feature. The jet p_T and width are also amongst the most helpful for discrimination, as theoretically expected. Similarly to figure 4.1.3, the ϕ feature plays a minor role. This is also expected given its uniform distribution and the symmetry in ϕ .
2. The improvement in the performance of Junipr saturates in the [300, 600] GeV energy region, leading Junipr to surpass the benchmarking models in the [200, 500] GeV region. The AUC obtained by *BJ 2* for the bin [500, ∞] is 0.864, still an excellent result but inferior to that of previous bins. The discriminative power of Junipr reduces for energies over a certain threshold, after reaching saturation. Increasing

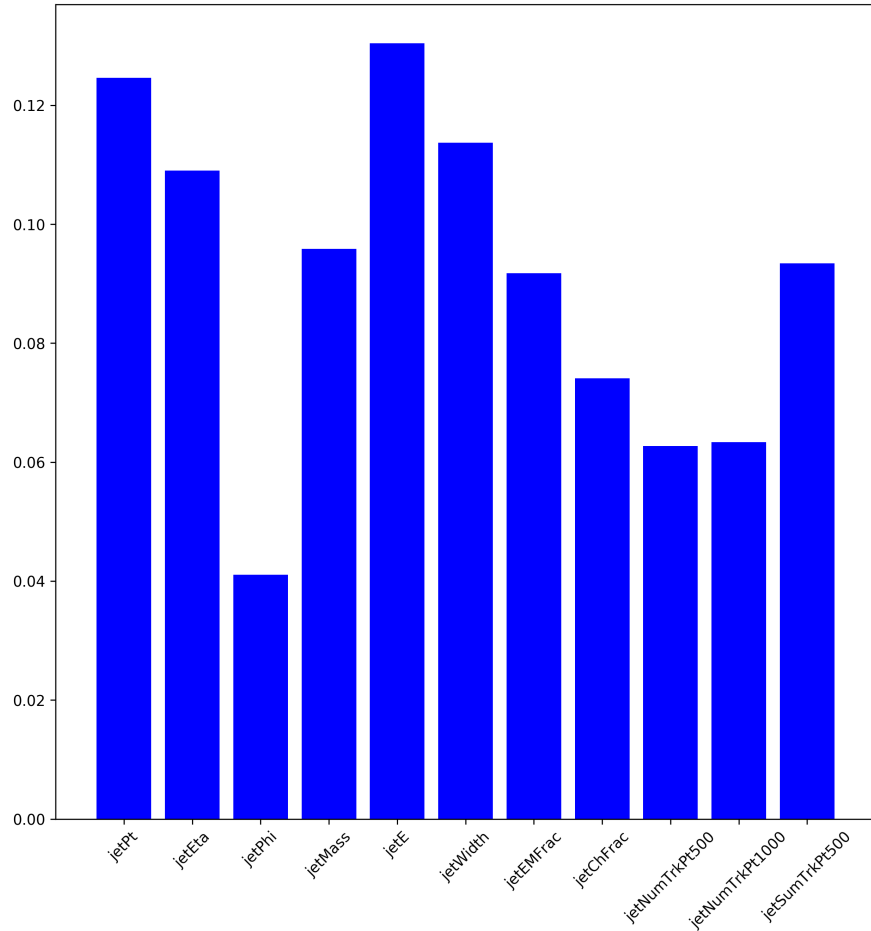


Figure 4.4.9 – The importance of each feature used in the optimal BDT trained on set 6.

the size of the model, as done for *BJ 4*, does not alter this behaviour. In fact, the larger model obtains very similar result to *BJ 2* indicating the saturation does not come from a lack of flexibility from the network. Such a mechanism might be the result of two competing processes: a positive one, increasing the performance with energy, and a negative one, degrading it for energetic jets. This is further discussed later in the chapter.

3. Observe how the normalised distribution of jets in set 4 and 5 agree (the dashed red and blue lines). Evidently, the E_{sub} cut does not drastically impact the global jet sample, as previously argued and justifying the use of set 6 for the global BDT and DNN benchmarking models. However, it has a clear and distinct effect on the performance of Junipr since *BJ 2* (trained on set 5) dominates *BJ 1* (trained on set 4) over the whole range of energy considered. Altogether, the analysis suggests that

a looser energy cut on constituent is advantageous to the task of quark versus gluon jet discrimination. This conclusion is supported by the previous discussions on the importance of jet width to separate these processes. Indeed, such soft radiations, better preserved with a lower E_{sub} cut, are an important driver of the final jet width.

4. The impact of using low-level variables versus high-level ones is suggested by the dashed curves, plotting the fraction of jets (in sets 4, 5, and 6) in each energy bin. Indeed, note that, from the same raw dataset, the reconstructed quark and gluon jet distributions differ when considering a jet as the set of its EMTopoJet constituents or as described by the engineered variables. There is a significant difference in energy distribution: jets reconstructed from EMTopoJet constituents have a lower energy than the EMTopoJet as fully reconstructed by the ATLAS collaboration. While the cut removing constituents with less than 0.5 GeV or 1 GeV play a role, several layers of processing and interactions with different parts of the detectors explain most of the origin of this gap. Indeed, although topocluster energy does account for a significant portion of the information used to construct jet energies, scalings, calibrations and corrections are also considered to reconstruct the energy with better precision. This typically involves joining calorimeter and tracker information, removing pile-up tracks, and calibrating calorimeter cell energy. The difference in jet energy induced by the distinct approaches, the low-level versus high-level, tend to make the former have smaller energies than the latter. This is shown in the plot by the much larger first bin of sets 4 and 5 versus set 6 of the normalised number of jets per energy bin distributions. When considering the global samples, jets seen by the binary Junipr models tend to have a lower energy which correlates with lower performance. This helps explain why the Junipr approach seemed to lag behind the benchmarking models when considering global energies. Note that this shift of energy distribution when going from high- to low-level variables explains why the best performance of the local Junipr implementation matches that of the binary Junipr paper [9] in a lower energy bin. Indeed, *BJ 2* reaches a maximum AUC of 0.891 in the [400, 500[GeV region while the Junipr paper obtains 0.898 in the [500, 550] GeV bin. In fact, figure 4.4.10 gives evidence to this behaviour by indicating the AUC curve as

a function of energy for a fifth binary Junipr model, *BJ 5*, trained similarly to *BJ 2* but on a dataset of Truth jets instead of EMTopoJets called set 5b. This dataset is made of final state truth particles instead of topoclusters and is, therefore, based on higher-level information. It is processed in the exact same way as set 5, hence its name (more information on the training of *BJ 5* is available in Appendix C.2). Clearly, the efficiency of *BJ 5* is only marginally better than *BJ 2*. Interestingly, it reaches its maximum in the $[500, 550]$ GeV bin at an AUC of 0.894, the same region and close to the AUC obtained in the binary Junipr paper, thereby supporting their results.

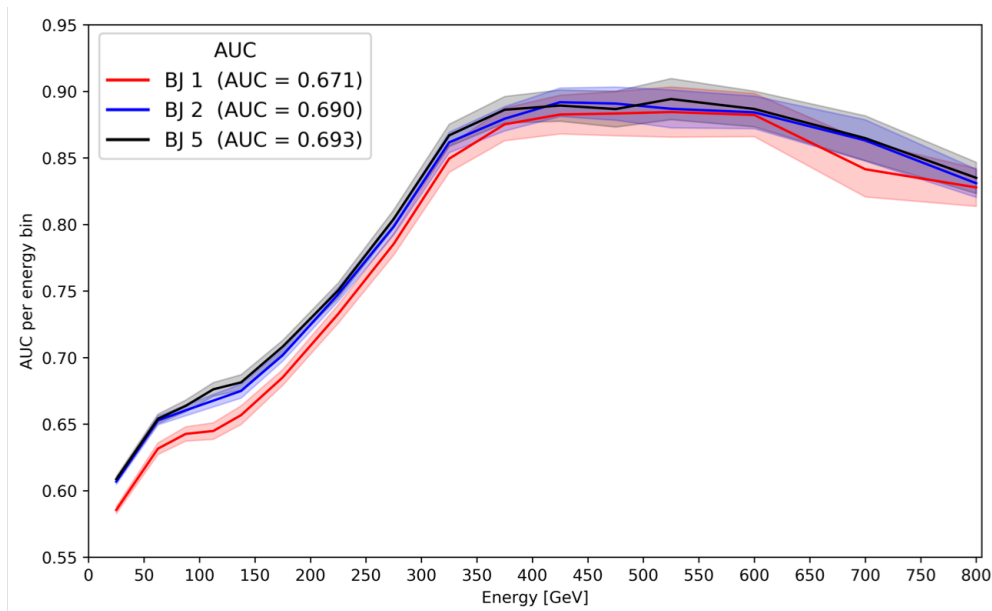


Figure 4.4.10 – The AUC of the binary Junipr models *BJ 1*, *BJ 2*, and *BJ 5* as a function of energy (same bins as in figure 4.4.8). The models are respectively trained and tested on set 4, set 5, and set 5b. The area around the curves correspond to $AUC \pm 1/\sqrt{\text{Number of jets in the bin}}$. Note that table C.2.1 summarises the difference between the Junipr models considered.

The interpretability of the Junipr framework offers some useful tools to analyse its efficiency. Figures 4.4.8 and 4.4.10 indicate an interesting behaviour at large energy: the performance of binary Junipr models saturates in the range $[300, 600]$ GeV and decreases afterwards. As previously mentioned, such a behaviour seems likely to be the result of at least two competing processes, respectively helping and degrading the accuracy when the energy increases. A positive effect is counterbalanced passed a certain energy level by a negative one, thereby reducing the efficiency and giving the observed saturated curve.

This process can be analysed by collecting information on the nodes of the binary trees (focusing on model *BJ 2*). There are $\sim 700\text{k}$ jets in the test set 5, for a total of $\sim 8.5\text{M}$ nodes of which as many as $\sim 3.9\text{M}$ are wrongly labelled (45.89%). A quark (gluon) node is wrongly labelled when its binary score is below (above) 0. In this context, the score is the value of the log-likelihood ratio test (without sigmoid). The distributions analysed in the following discussion focuses on these wrongly labelled nodes. The effects they unveil are discussed in depth after briefly introducing the figures. An analysis of correctly labelled nodes is left for future research.

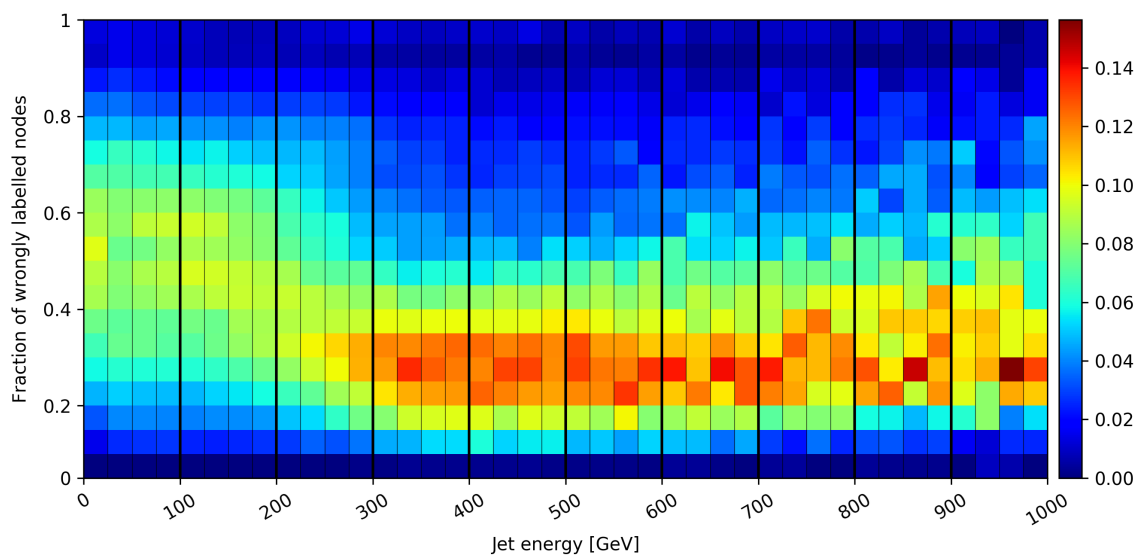


Figure 4.4.11 – Two-dimensional histogram of the miss-labelled nodes as analysed by *BJ 2* on set 5. Histogram of the jet energy (*x*-axis) versus the fraction of nodes in the tree that are miss-labelled (*y*-axis). The count is normalised to unity for each energy bin (each vertical column separated by thin black lines) and displayed with a colour-code.

Figure 4.4.11 displays a two-dimensional histogram of the jet energy (in GeV) versus the fraction of the nodes of the jet tree that are wrongly labelled (considering only jets with at least one miss-labelled node). Each energy bin (each column) is normalised to unity and the colour code indicates the fraction associated to each cell. The histograms should be read as a succession of vertical columns (separated by the thin black lines), evolving with energy and displaying the normalised distribution of the fraction of jet nodes that are wrongly labelled. Note that, from the jets that had at least one miss-labelled node, almost none had a very small fraction of miss-labelled nodes, as indicated by the deep blue colour of the bottom horizontal row of cells.

Figure 4.4.12 displays two two-dimensional histograms of the relative depth of the miss-labelled nodes versus their score (the first one has a logarithmic colour scale while the second one uses a linear colour scale). The relative depth is defined by the position of the node in the clustering history (the root node is at depth 0, the last node at depth 1, and values between are obtained by the ratio of their position to the total number of nodes). Interestingly, more gluon nodes seem to be wrongly predicted to match quark nodes than the opposite (more weight is placed in the score > 0 region). This is expected, given they are more gluon than quark nodes in total: as previously mentioned, gluon jets tend to be made of more constituents and the clustering history thus contains more nodes. Interestingly, the very last nodes are the not the most commonly miss-labelled ones.

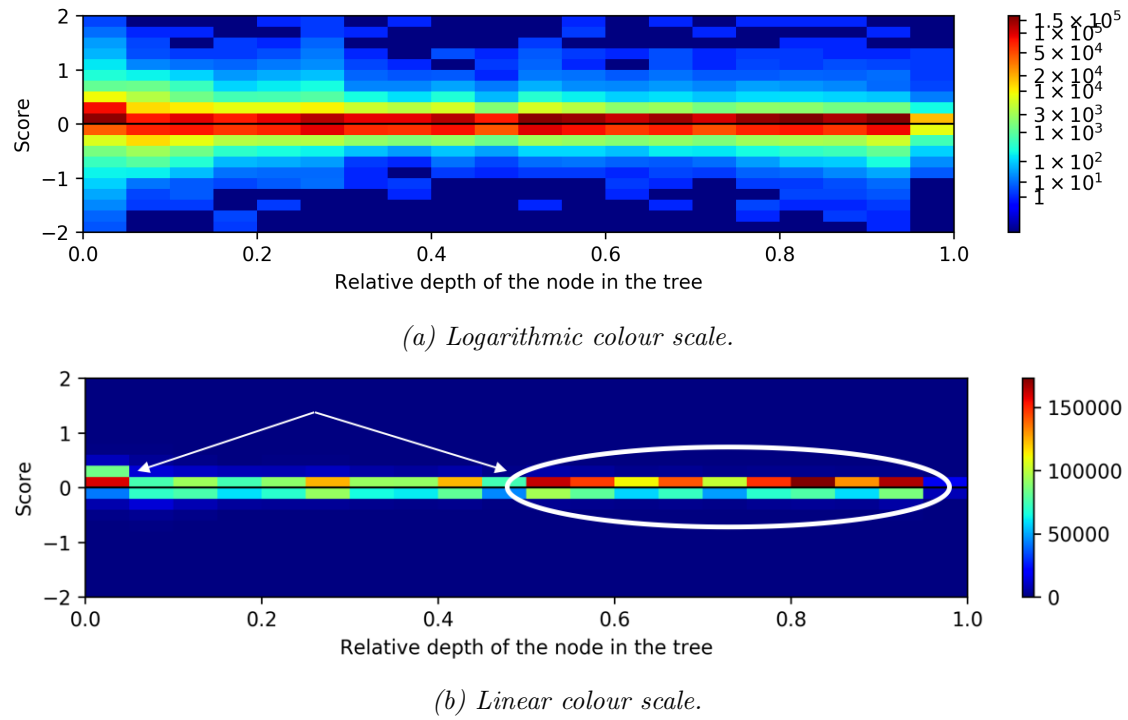


Figure 4.4.12 – Two-dimensional histograms of the miss-labelled nodes as analysed by BJ 2 on set 5. Histograms of the relative depth of the node (x -axis) versus the score (y -axis). Values above 0 (respectively, below 0) indicate nodes wrongly predicted to match a quark node (a gluon node). The top plot uses a logarithmic colour scale and the bottom one a linear colour scale. The white arrows and ellipse indicate parts of the plot mentioned in the text.

Figure 4.4.13 displays three plots corresponding to the jet energy bins $[100, 200[$ (top), $[300, 400[$ (centre), and $[500, \infty[$ (bottom). Each plot is a two-dimensional histogram of the mother energy (of the miss-labelled node) versus the score assigned. Naturally, it is impossible for a mother (of a node) to be more energetic than the jet considered, hence

the sharp interruption of the two top plots at respectively 200 and 400 GeV.

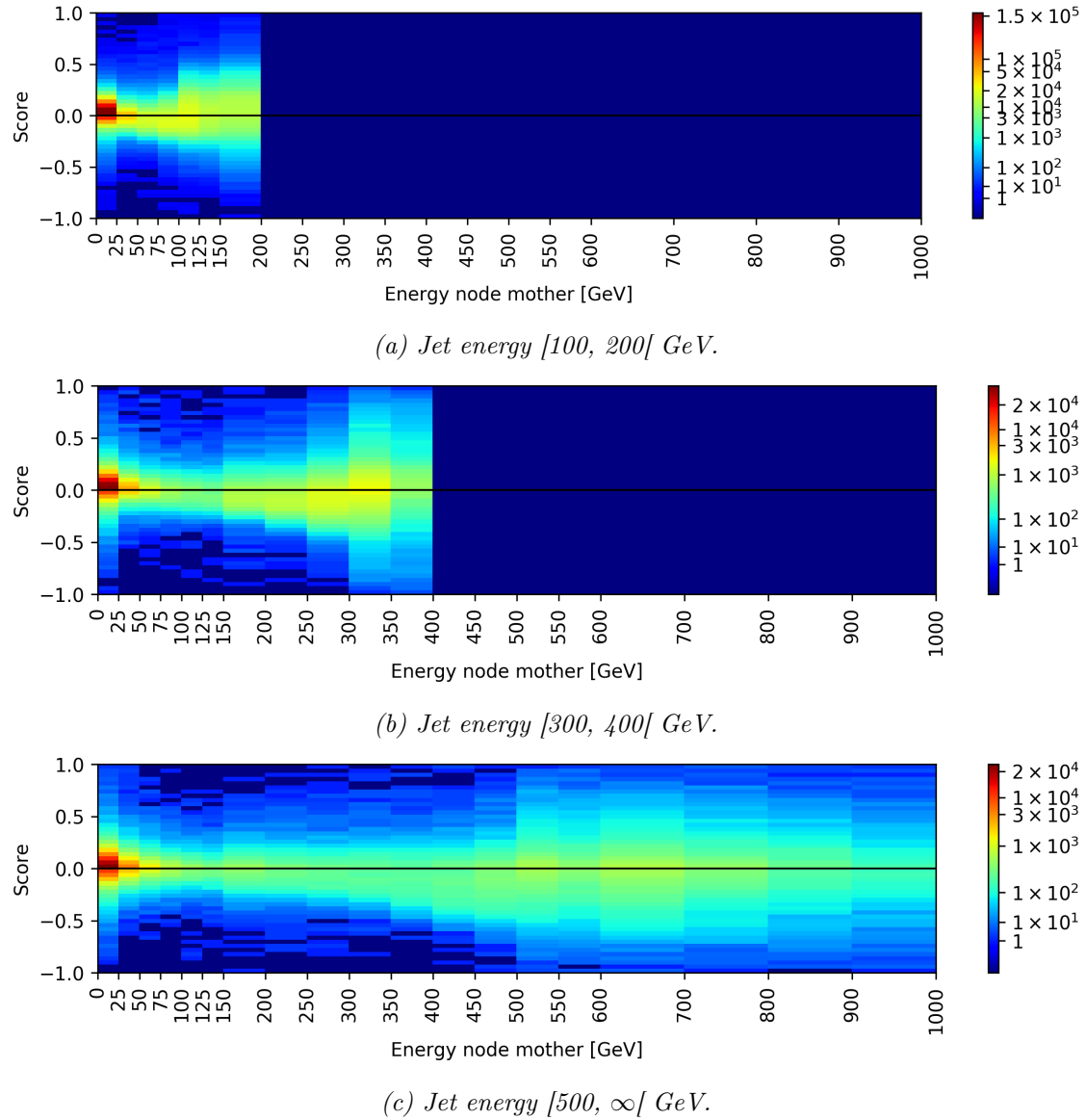


Figure 4.4.13 – Two-dimensional histograms of the miss-labelled nodes as analysed by BJ 2 on set 5. Histograms of the mother energy of miss-labelled nodes (x -axis) versus the score (y -axis). Each plot focuses on a specific jet energy window, as specified under them.

The information revealed in the three figures can now be analysed to explain the saturation of Junipr and its decreasing efficiency at very high energies. While a complete and rigorous analysis of potential effects arising is left for future work, three elements are observed to drive this mechanism. Note that the following points only address the number of wrongly assessed nodes, not the magnitude of their incorrect score.

1. The positive effect is visible in figure 4.4.11. It produces the following pattern:
 - At low-energies (< 200 GeV), the fraction of wrongly labelled nodes in the trees is rather high, with an average of about 0.5: half of the nodes of the jets are miss-labelled (for jets having at least one miss-labelled node).
 - This fraction reduces when the energy increases and focalises around 0.3 (in bright orange-red) for energies in $[300, 700]$ GeV.

This positive effect, increasing the performance with energy, can be understood to arise from a reduction in the bias introduced by wrongly predicted nodes (they get averaged out by more correctly assessed ones). In the first step, the proportion of correctly labelled nodes increases with energy.

2. In the last part of figure 4.4.11, the distribution (per column) seems to widen upwards, extending more of its support back up to the large fraction of miss-labelled nodes per jet (the variance increases and the distribution skews). This is particularly visible given the penultimate row of cells turns to a darker blue shade after 700 GeV. This observation suggests that, in the range of energy above 700 GeV, the fraction of miss-labelled nodes in specific jet trees increases. This negative mechanism can be understood as an increase in poorly reconstructed low-energy nodes. It dominates and shrinks the AUC by adding far more badly-assessed nodes than correctly-assessed nodes are introduced from the previous effect. This explains why, in figure 4.4.8, the gap between the AUC of *BJ 2* and *BJ 1* reduces when the energy increases: *BJ 1* is better protected against the increase in soft radiations since it removes constituents with an energy $E < 1$ GeV versus $E < 0.5$ GeV for *BJ 2*.

A final point supporting this hypothesis comes from figure 4.4.13. Whatever the jet energy bin considered, the most commonly miss-labelled nodes have mothers of very low-energy. Naturally, when increasing the energy, a larger number of decays will occur in the jet tree and more low-energy mother will be produced in the last stages of hadronisation. Therefore, passed a certain level of energy, this process increases the weight of low-energy nodes in the jet tree, generating the negative mechanism proposed here.

3. Finally, a last negative mechanism can be observed from the previous plots, figure 4.4.13. Indeed, while the effect of the low-energy nodes has just been discussed, there is also an apparent high-energy process. Observe how, for larger mother energies, the score distribution widens and takes on a yellow shade (after contracting in the [75, 150] GeV region, taking on a light blue-green shade). This signifies that very high energy nodes in the trees are also likely to be miss-labelled. These nodes are naturally located at the very beginning of the tree since the energy is shared between formed elements. This is visible in figure 4.4.12. Clearly, miss-labelled nodes mostly correspond to two specific areas: the low relative depth (left white arrow) and the second half of the tree (right arrow pointing to white ellipsis). These respectively mostly correspond to the high and low energy nodes.

The precision of Junipr on high energy nodes may be inherently lower than on average energy ones. Indeed, even when submitting a high energy jet sample to Junipr for training, most of the nodes of the trees have a much lower energy given it is quickly divided among the created elements. This makes the task of labelling high energy nodes inherently harder given the rarity of such items. Furthermore, these high energy nodes are usually located at the beginning of the tree. Therefore, it is possible that the predictions made by Junipr suffer from the low information contained in $h^{(t)}$ for small t . The hidden state can only contain information on the jet that has been uncovered previously. Consequently, unary Junipr models may not have a clear picture of the jet when producing a likelihood on these early nodes, thereby reducing the efficiency of the whole method. If a quark and a gluon unary Junipr models are not able to confidently tag such a node, the binary score outputted at this node will be close to 0. This leaves the other nodes of the tree more room to decide the label to predict. But, at high energies, many of these other nodes are related to low-energy transfers given these are common in the last stages of hadronisation. These nodes can then easily render the global prediction unreliable. The effect of this mechanism is suggested in figure 4.4.14, displaying two jet trees of $BJ\ 2$ in the $[500, \infty[$ GeV region (jet trees of other regions are presented in Appendix C.4). Note how the model is incapable of suggesting a preference from the first node of both trees. Most of the

miss-labelled nodes on these two trees are also localised towards the end.

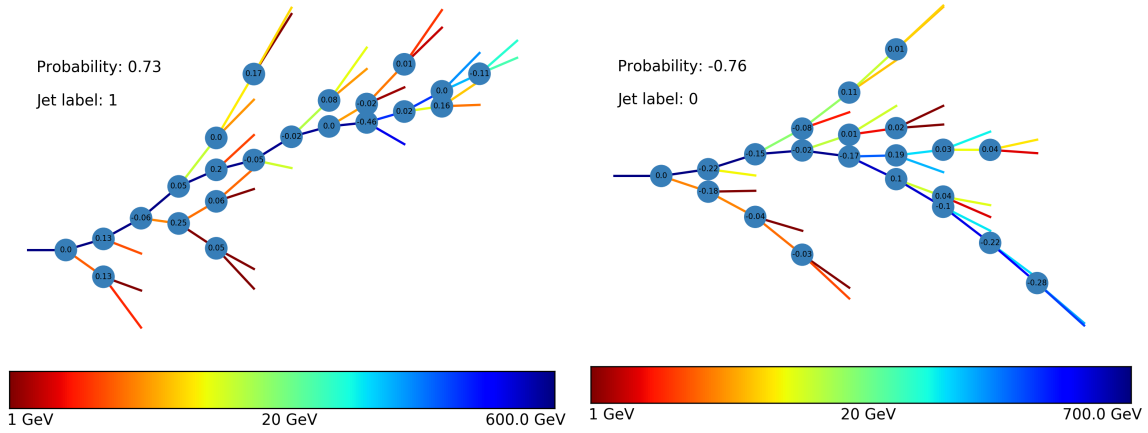


Figure 4.4.14 – Quark (left) and gluon (right) jets from set 5 in the $[500, \infty[$ GeV region analysed by the binary Junipr model trained (BJ 2).

Summary of the Chapter

This chapter explores two important questions related to the topic of quark versus gluon jet discrimination:

- Is a Junipr approach competitive at the reconstructed level versus BDT and DNN benchmarking models?
- Does low-level information from the calorimeter (topocluster momenta) offer sufficient information for the task?

The adapted Junipr framework provides a resounding *yes* to both interrogations. Compared to the benchmarking models using higher-level variables, not only is a Junipr model based on low-level information competitive across the whole energy range considered, it even dominates in the $[200, 500]$ GeV region. Some further experiments, taking a closer look at the properties of the wrongly assessed nodes, are presented to analyse the degrading performance of Junipr passed a certain energy threshold. Three mechanisms playing a role in the number of correctly labelled nodes are uncovered: the amount energy transferred in the associated decay, the position of a node in the factorisation tree, and the number of low-energy nodes are all shown to impact the quality of the predictions.

Conclusions and Future Work

Several important elements and conclusions have been uncovered and defended in this project. The approach is restricted to match as much as possible the one adopted in the Junipr papers [8, 9], so that a meaningful comparison to both the benchmarking models, the BDT and DNN, and the published Junipr studies can be established.

The pipeline adopted to implement the project relies on an extensive chain of programs to collect, produce, process, and analyse the data, driven by computational efficiency and physics-related requirements. Multiprocessing and interconnection of the codebase, structured into an appropriate set of classes, proved useful to provide an adaptable and efficient environment to carry out the necessary experiments. An implementation of the extensions discussed later in this chapter can in fact seamlessly and efficiently be integrated into the existing framework.

This work is the first study of an application of Junipr to the ATLAS experiment. The focus of the present dissertation is to test the algorithm in the presence of realistic conditions - including detector effects and pile-up, both of which have the potential to severely degrade its performance. It also investigates the possibility of using low-level information as sole input.

Facing these two constraints, the adapted Junipr framework has been shown to successfully and reliably achieve the task of quark versus gluon jet discrimination. The analyses indeed uncovered two major observations:

1. On a typical energy distribution of jets observed in the detector, the benchmarking models, the BDT and DNN, using high-level variables perform better than the Junipr model with low-level variables (calorimeter topoclusters). Indeed, the benchmarking models are able to efficiently leverage the extensive physics knowledge summarised into a set of well-designed variables. However, when restricting to the energy region $[200, 500]$ GeV, the Junipr approach dominates. The performance obtained is, remarkably, on par with that reported in the binary Junipr paper [9] at the generated-level, even in the presence of detector noise and high pile-up. Interestingly, the physics program at CERN is mostly focused on high-energy events as the more intricate aspects of the world only become visible at higher energies. Indeed, as explained in the physics introduction, section 2.1, higher energies reveal finer details of the fabric of our Universe.
2. It is possible to efficiently discriminate between quark and gluon jets using low-level information. This is not a light conclusion: a real application of this work is to implement it as a trigger. The Junipr model, while slow to train, is extremely fast at making a prediction from the detector point-of-view: the very first information gathered by the detector only needs to pass through a recurrent component and four parallel DNN's of small sizes (a single hidden layer of ten units was used in this work). It does not require high-level information nor complex computational structures. A trained Junipr model can be attached to a clustering algorithm. While combining final states information (such as topoclusters momenta) into a jet, a Junipr model could already produce a tag on the observed pattern. As a trigger, it would add the valuable tool of online quark and gluon jet tagging to the arsenal of the ATLAS experiment. This opens a path to several important physics analyses searching for evidence of extensions of the Standard Model.

Future work could include to rigorously understand the limitations of Junipr models. Several effects uncovered in this project indeed need to be studied further in-depth to assess the potential of this approach. The apparent saturation of the accuracy of Junipr preceding its decline at high energies raises important questions regarding the sensitivity of the model to wrongly predicted nodes. The competing mechanisms acting with energy still need to be properly exposed, demonstrated, and analysed. A rigorous study of the probability assigned to each node depending on the energy or splitting fraction (z) is a natural point to start such an extension. Another important element requiring a rigorous analysis is the probabilistic nature of the results. The experiments should be repeated several times with different random seeds to compute more statistically sound estimates, in particular concerning the efficiencies of each method. Finally, the datasets generated for the purpose of this project contained a lower number of events in very high energy bins. The analysis should be carried out with a larger sample of high energy jets to avoid any statistical fluctuation in the results.

The analysis presented in this dissertation only contains tools and architectures currently used in Particle Physics, plus that specifically introduced in the Junipr papers. The interest of this restriction is to offer meaningful points of comparison with the selected literature. However, the specific settings of the ATLAS experiment may require to divert away from these limitations. Examples of adaptations of the structure to explore include:

- Different recurrent cells, such as Gated Recurrent Unit (GRU) [69]. After all, the efficiency of the method depends on the accuracy of the internal jet representation Junipr builds within $h^{(t)}$.
- The setting of weak supervision also deserves to be studied in details. A competitive result could suggest training Junipr directly on samples of real collected data that is known to contain an enriched fraction of quark or gluon jets, thereby removing the computationally expensive process of simulating the signals.
- The global binary Junipr scheme deployed in this project corresponds to that proposed in its original paper [9]. It is strongly motivated theoretically since it corresponds to a likelihood ratio test, the most statistically powerful discriminant between

two hypotheses under the Neyman-Pearson lemma [65]. While the test is optimal, experimentally, the jets likelihoods employed are not. They are sensitive to miss-predicted nodes and unlikely events. The situation is akin to performing the best test with imperfect information. A more powerful approach could however easily be implemented. Instead of basing the binary objective on the global jet likelihood (the two values outputted by the unary models), it could be constructed from a subset of the available nodes that are deemed more reliable. At least three approaches can be adopted to implement this idea:

1. The simplest option is to add a third class corresponding to pure noise. The Junipr models could then avoid associating a quark or gluon label to noisy nodes and instead assign the third class label. Gathering a dataset enriched in this new process is fairly easy: pure noise is what is observed most of the time in the detector. There are sets of such events available, collected when the detector is analysing collisions without any trigger activated to select specific events. Furthermore, generalising Junipr to the task of n -discrimination poses no essential challenge of any sort, as discussed in section 3.2.3.3.
2. The trees of each unary model can be trimmed down to only contain the most reliable nodes. An analogous likelihood ratio test could then be deployed on the resulting structure. An obvious issue with this approach is that the likelihood of a tree is inversely proportional to its size (its number of nodes) since each node samples from a set of probability distributions. Therefore, if the quark and gluon trees have different sizes, the larger-size tree risks losing out to the smaller-size one. However, a larger tree means more nodes are considered reliable which, in turn, suggests this tree is more likely coming from the unary model trained on the same label. To compensate for that effect, a partial likelihood for each tree can be computed from a given number of nodes (this number could be the smallest tree size in the pair), taking for each resulting tree the nodes with the best values (the most positive ones for the quark tree and the most negative ones for the gluon tree). The two scores can then be compared in a ratio balancing towards the most reliable prediction.

3. A more subtle idea is to transfer the likelihood of each node from each unary tree, plus some information on the decays associated with these nodes, to a specialised binary classification network, referred to as *SBCN*. The mapping adapting the likelihood based on the size of the reliable part of the trees would then be learnt internally by adopting a convenient structure for *SBCN*. The first layer of this network could be formed by a set of $2 \times \text{number of nodes}$ blocks, with one block for each node of each unary tree. Each block of this first layer would receive some dynamical information on the node (energy of the mother, z , angles, ...), the model it comes from (quark or gluon), and the probability given to the node by this model. In subsequent layers, the blocks associated with each model would be connected to each other and to the block of the equivalent node from the other model. This network would therefore have the ability to exploit the information of each node of each tree of each model as it sees fit to optimise the binary objective. This approach sacrifices a part of the interpretability of Junipr but is expected to perform better, given the added flexibility. This is in the end what truly matters for a trigger or a method to classify data.

These generalisations are expected to perform well for a simple reason: in the limit case where the adaptive part is completely flexible (keeping every node or never assigning the third class), they correspond to the regular flavour of binary Junipr. The adapted cases would however have a margin to protect their predictions from poorly assessed nodes. They can, therefore, reasonably be expected to leverage this increased power to avoid the efficiency reduction at higher energies and could even improve the low-energy performance. The increase in computational cost during test time would be limited, each of them consisting in a simple pass through a network or a sorting algorithm selecting a given number of nodes. These methods can still be attached to a clustering algorithm and can therefore also be considered for online triggering.

- An adapted factorisation to the ATLAS experiment. The factorisation of the jet likelihood followed a generic physics approach based on the structure of QCD. While meaningful from a theoretical point-of-view, this is not the only way of factorising a

jet. An approach adopting a more experimental view could prove beneficial to the task. In this work, the factorisation structure of $1 \rightarrow 2$ binary mother-to-daughter *particles* decay was extended to work with calorimeter *topoclusters* momenta. There is nothing hindering an even more detector-focused factorisation, combining for example tracker hits constituents with topoclusters momenta. Such an extension would leverage more of the information collected by the detector.

- The Junipr models presented in this work adopted the root-to-leaves approach of the original papers. However, there is nothing restricting the direction of the analysis. In fact, the classical picture of $1 \rightarrow 2$ decay is a fully reversible process. Inverting the direction of time to get $1 \leftarrow 2$, a comparable jet structure is built. This inversion is indirectly used in the present work: the real simulated process is a sequence of *decay*, hence going from root-to-leaves ($\rightarrow$). However, Junipr jets are reclustered from leaves-to-root ($\leftarrow$) and analysed from root-to-leaves ($\rightarrow$). A logical extension of the work is to analyse a Junipr model working on the leaves-to-root ($\leftarrow$) sequence since this is the order of reconstruction performed when observing data in the detector. This step is vital for the online trigger application, so that a model can be included directly in a factorisation algorithm. If the global set of final state constituents is used in the DNN mapping producing the first hidden state, it could also prove beneficial to increase the initial information content of $h^{(1)}$.

The present dissertation focuses on the specific task of quark versus gluon jet discrimination. More generally, discriminating between any types of jets is a closely related subject. The two problems are in fact completely equivalent when approaching the tasks with Junipr, as evidenced by the very first example presented in its original paper [8]: comparing boosted Z boson jets to energetic quark jets. A prime candidate of such an extension for the ATLAS experiment is to discriminate between different flavours of quark jets (such as the charm c , strange s , bottom b , ...). The work carried out in this project could therefore prove useful in a large range of similar endeavours. This paves the way for an exciting set of new experiments and tests of the Junipr approach that could deliver novel and useful tools to the collaboration.

Furthermore, when considering the specific structure of the ATLAS detector, the Junipr framework could even generalise to tagging objects that are not jets. Its basis for discrimination is a factorised probabilistic model scaffolded around a tree structure (the tree structure, in the present case, arose from the factorisation of a jet as a follow-up of binary decays). To say that this is a common pattern in Particle Physics would be a euphemism. This structure appears everywhere in experimental Physics. It is not only due to theoretical aspects (such as in the factorisation theorem of QCD) but also to the often sequential nature of observations associated with a single object. Physics is governed by a naturally sequential chain of causal events, materialising the existence of a time arrow. Almost all observations made in the ATLAS detector, therefore, have a sequential structure and an associated tree pattern. For example, an electron going through the detector would leave a succession of hits in the tracker and calorimeter (ECAL) cells. This succession of hits and deposits forms a tree, albeit a very simple one given it mostly has one branch corresponding to the trajectory of the elementary particle. Note that it could still have more than a single branch because an electron can interact electromagnetically with several cells of the same layer (interaction at a distance). Nothing prevents the implementation of a Junipr model on this sort of information. The factorised probability distribution underpinned by such an approach models the likelihood of having the observed information in the given distribution of subdetectors. While it removes the interpretable aspect of Junipr, talking in purely experimental terms becomes possible which opens the door to possible improvements in tagging efficiency of a large variety of physics objects.

This last idea can even be pushed further: Junipr could be directly deployed to search for specific physical processes and not only tag objects. Indeed, consider a process as the set of information sequentially left in successive components of the ATLAS detector and a natural tree structure emerges. For example, a dijet event could have both its jets modelled within the same tree (of two clear branches) and a unique unary Junipr model can run on this structure. The hidden state of such a model would be tasked with extracting general information on the evolution of the process. Such a task is, naturally, more complex than simple objects analyses, but it might be possible to adapt networks designed in the context of natural language processing to this setting. In the end, the complexity of such patterns

is still inferior to that of language analysis. All these elements deserve their own enquiries to assess whether a Junipr approach can improve the current efficiencies obtained.

Note that all of the previous points raised in these outlook suggestions refer to the discriminative application of the Junipr model. However, as suggested in section 3.2.3.3, Junipr extends its reach further away from this specific task. Exploring its abilities to reweight distributions and generate events efficiently could prove valuable if a competitive setup can be implemented compared to the currently deployed methods.

In conclusion, the Junipr approach to model sequential information factorised into a tree structure provides exciting avenues of development in the experimental Physics context. The prize of any significant improvement in the discriminative, generative and reweighing tasks would be a finer toolbox of methods available to particle physicists, helping them bring to light the next major discoveries unveiling the fundamental rules and constituents of our Universe.

Bibliography

- [1] P. W. Higgs. Broken symmetries and the masses of gauge bosons. *Phys. Rev. Lett.*, 13:508–509, 1964. (Cited on page 7)
- [2] F. Englert and R. Brout. Broken symmetry and the mass of gauge vector mesons. *Phys. Rev. Lett.*, 13:321–323, 1964. (Cited on page 7)
- [3] The ATLAS Collaboration. Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC. *Phys. Lett. B*, 716:1–29, 2012. (Cited on page 7)
- [4] M.K. Gaillard, P.D. Grannis, and F.J. Sciulli. The standard model of particle physics. *Rev. Mod. Phys.*, 71, 1999. (Cited on pages 7 and 11)
- [5] CERN data centre passes the 200-petabyte milestone. <https://home.cern/news/news/computing/cern-data-centre-passes-200-petabyte-milestone>. Accessed: 2020-08-31. (Cited on page 8)
- [6] Processing: What to record? <https://home.cern/science/computing/processing-what-record>. Accessed: 2020-08-31. (Cited on page 8)
- [7] 53 incredible facebook statistics and facts. <https://www.brandwatch.com/blog/facebook-statistics/>. Accessed: 2020-08-31. (Cited on page 8)

- [8] A. Andreassen, I. Feige, C. Frye, and M.D. Schwartz. JUNIPR: a Framework for Unsupervised Machine Learning in Particle Physics. *Eur. Phys. J.*, C79, 2019. (Cited on pages [8](#), [21](#), [32](#), [34](#), [47](#), [52](#), [53](#), [56](#), [57](#), [58](#), [70](#), [71](#), [90](#), [104](#), [109](#), and [C. 18](#))
- [9] A. Andreassen, C. Frye I. Feige, and M.D. Schwartz. Binary JUNIPR: an interpretable probabilistic model for discrimination. *Phys. Rev. Lett.*, 123, 2019. (Cited on pages [8](#), [22](#), [53](#), [55](#), [56](#), [58](#), [70](#), [71](#), [78](#), [82](#), [89](#), [90](#), [96](#), [104](#), [105](#), and [106](#))
- [10] The Standard Model (of Physics) at 50. <https://blogs.scientificamerican.com/observations/the-standard-model-of-physics-at-50/>. Accessed: 2020-04-18. (Cited on page [11](#))
- [11] D. Hanneke, S. Fogwell Hoogerheide, and G. Gabrielse. Cavity control of a single-electron quantum cyclotron: Measuring the electron magnetic moment. *Phys. Rev. A*, 83:052122, 2011. (Cited on page [11](#))
- [12] Wikipedia: The Standard Model (of Physics). https://simple.wikipedia.org/wiki/Standard_Model. Accessed: 2020-04-18. (Cited on page [12](#))
- [13] S. Marzani, G. Soyez, and M. Spannowsky. Looking Inside Jets. *Springer Int. Pub.*, 2020. (Cited on pages [13](#), [32](#), [33](#), and [53](#))
- [14] The ATLAS Collaboration. Search for strong gravity in multijet final states produced in pp collisions at $\sqrt{s} = 13$ TeV using the ATLAS detector at the LHC. *JHEP*, 03:026, 2016. (Cited on page [14](#))
- [15] Esma Mobs. The CERN accelerator complex - 2019. Complexe des accélérateurs du CERN - 2019. 2019. General Photo. (Cited on page [15](#))
- [16] A.D. Martin, W.J. Stirling, and R.S. Thorne et al. Parton distributions for the LHC. *Eur.Phys.J.C*, 63:189–285, 2009. (Cited on page [16](#))
- [17] The ATLAS Collaboration website. <http://www.atlas.ch/>. Accessed: 2020-04-18. (Cited on page [16](#))
- [18] The ATLAS Collaboration. The ATLAS Experiment at the CERN Large Hadron Collider. *Journal of Instrumentation* 3, S08003, 2008. (Cited on page [17](#))

- [19] C. Ay et al. Monte Carlo generators in ATLAS software. *J. Phys. Conf. Ser.*, 219:032001, 2010. (Cited on page 19)
- [20] S. Sekmen. Beyond the Standard Model physics at the high luminosity LHC. *PoS, ICHEP2018*:283, 2018. (Cited on page 20)
- [21] D. Guest, K. Cranmer, and D. Whiteson. Deep Learning and its Application to LHC Physics. *Ann. Rev. Nucl. Part. Sci.*, 68:161–181, 2018. (Cited on page 20)
- [22] P. Baldi, P. Sadowski, and D. Whiteson. Searching for Exotic Particles in High-Energy Physics with Deep Learning. *Nature Commun.*, 5:4308, 2014. (Cited on page 21)
- [23] G. Kasieczka, Plehn N. Kiefer, Tilman, and J.M. Thompson. Quark-Gluon Tagging: Machine Learning vs Detector. *SciPost Phys.*, 2019. (Cited on pages 21, 43, and 69)
- [24] Pushpalatha C. Bhat. Multivariate Analysis Methods in Particle Physics. *Ann. Rev. Nucl. Part. Sci.*, 61:281–309, 2011. (Cited on page 21)
- [25] The ATLAS Collaboration. Quark versus Gluon Jet Tagging Using Charged-Particle Constituent Multiplicity with the ATLAS Detector. *ATL-PHYS-PUB-2017-009*, 2017. (Cited on page 21)
- [26] P. T. Komiske, E. M. Metodiev, and J. Thaler. An Operational Definition of Quark and Gluon Jets. *JHEP*, 11, 2018. (Cited on pages 21 and 22)
- [27] The ATLAS Collaboration. Quark versus gluon jet tagging using jet images with the ATLAS detector. *ATL-PHYS-PUB-2017-017*, 2017. (Cited on pages 21 and 68)
- [28] T. Cheng. Recursive Neural Networks in Quark/Gluon Tagging. *Computing and Software for Big Science 2.1*, 2018. (Cited on page 22)
- [29] Huilin Qu and Loukas Gouskos. ParticleNet: Jet Tagging via Particle Clouds. *Phys. Rev. D*, 101(5):056019, 2020. (Cited on page 22)
- [30] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P.S. Yu. A Comprehensive Survey on Graph Neural Networks. *IEEE transactions on neural networks and learning systems*, 2020. (Cited on page 22)

- [31] P.T. Komiske, E.M. Metodiev, and J. Thaler. Energy flow polynomials: A complete linear basis for jet substructure. *JHEP*, 2018:1–54, 2018. (Cited on page 22)
- [32] P. T. Komiske, E. Metodiev, and J. Thaler. Energy flow networks: deep sets for particle jets. *JHEP*, 2019:1–46, 2019. (Cited on pages 22 and 89)
- [33] The ATLAS Collaboration. Measurement of the charged-particle multiplicity inside jets from $\sqrt{s} = 8$ TeV pp collisions with the ATLAS detector. *Eur. Phys. J. C*, 76, 2016. (Cited on page 22)
- [34] J. Hernandez-Gonzalez, I. Inza, and J.A. Lozano. Weak Supervision and Other Non-Standard Classification Problems: a Taxonomy. *Patt. Recog. Lett.*, 69:49–55, 2016. (Cited on page 22)
- [35] L.M. Dery, B. Nachman, F. Rubbo, and A. Schwartzman. Weakly Supervised Classification in High Energy Physics. *JHEP 05*, 145, 2017. (Cited on page 22)
- [36] P. T. Komiske, E. M. Metodiev, B. Nachman, and M. D. Schwartz. Learning to Classify from Impure Samples with High-Dimensional Data. *Phys. Rev. D*, 98, 2018. (Cited on page 22)
- [37] E.M. Metodiev, B. Nachman, and J. Thaler. Classification without labels: Learning from mixed samples in high energy physics. *JHEP 10*, 174, 2017. (Cited on page 22)
- [38] S. Otten, S. Caron, W. de Swart, M. van Beekveld, L. Hendriks, C. van Leeuwen, D. Podareanu, R. Ruiz de Austri, and R. Verheyen. Event Generation and Statistical Sampling for Physics with Deep Generative Models and a Density Information Buffer. 2019. (Cited on page 26)
- [39] The ATLAS collaboration. The ATLAS Simulation Infrastructure. *Eur. Phys. J. C*, 70:823–874, 2010. (Cited on page 27)
- [40] T. Sjöstrand, S. Ask, J. R. Christiansen, R. Corke, N. Desai, P. Ilten, S. Mrenna, S. Prestel, C. O. Rasmussen, and P. Z. Skands. An Introduction to PYTHIA 8.2. *Comput. Phys. Commun.*, 191:159–177, 2015. (Cited on page 27)

- [41] J. Alwall, R. Frederix, S. Frixione, V. Hirschi, F. Maltoni, O. Mattelaer, H. S. Shao, T. Stelzer, P. Torrielli, and M. Zaro. The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations. *JHEP*, 07:079, 2014. (Cited on page 27)
- [42] C. Oleari. The POWHEG-BOX. *Nucl. Phys. B Proc. Suppl.*, 205-206:36–41, 2010. (Cited on page 27)
- [43] GEANT4 Collaboration. GEANT4 -a simulation toolkit. *Nucl. Instrum. Meth. A*, 506:250–303, 2003. (Cited on page 27)
- [44] The ATLAS softwares. <https://atlassoftwaredocs.web.cern.ch>. Accessed: 2020-08-17. (Cited on pages 28 and 29)
- [45] The RUCIO library. <https://rucio.readthedocs.io>. Accessed: 2020-08-17. (Cited on page 28)
- [46] R. Brun and F. Rademakers. ROOT: An object oriented data analysis framework. *Nucl. Instrum. Meth. A*, 389:81–86, 1997. (Cited on page 29)
- [47] The UpRoot package. <https://uproot.readthedocs.io/en/latest/>. Accessed: 2020-08-17. (Cited on page 30)
- [48] The ATLAS collaboration. Topological cell clustering in the ATLAS calorimeters and its performance in LHC Run 1. Topological cell clustering in the ATLAS calorimeters and its performance in LHC Run 1. *Eur. Phys. J. C*, 77(CERN-PH-EP-2015-304):490. 87 p, 2016. (Cited on page 31)
- [49] The PyJet library. <https://pypi.org/project/pyjet/>. Accessed: 2020-08-17. (Cited on page 31)
- [50] Matteo Cacciari, Gavin P. Salam, and Gregory Soyez. FastJet User Manual. *Eur. Phys. J. C*, 72:1896, 2012. (Cited on pages 31 and 32)
- [51] Htcondor version 8.9.8 manual. <https://htcondor.readthedocs.io/en/latest/>. Accessed: 2020-08-17. (Cited on page 40)

- [52] Particle physics linux condor batch farm. <https://www2.physics.ox.ac.uk/it-services/particle-physics-linux-condor-batch-farm>. Accessed: 2020-08-17. (Cited on page 40)
- [53] M. Somvanshi, P. Chavan, S. Tambade, and S. V. Shinde. A review of machine learning techniques using decision tree and support vector machine. In *ICCUBE*, pages 1–7, 2016. (Cited on page 45)
- [54] Y. Freund and R.E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning*, ICML’96. Morgan Kaufmann Publishers Inc., 1996. (Cited on page 45)
- [55] Yoav Freund and Robert E. Schapire. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *J. Comput. Syst. Sci.*, 55(1):119–139, 1997. (Cited on page 45)
- [56] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. (Cited on page 45)
- [57] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>. (Cited on page 46)
- [58] K. P. Murphy. *Machine learning : a probabilistic perspective*. MIT Press, 2013. (Cited on page 46)
- [59] A. Calinescu and C. Haase. Course: Machine learning. University of Oxford, Michaelmas term 2019. (Cited on page 46)
- [60] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in PyTorch. 2017. (Cited on page 46)
- [61] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning*

Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015. (Cited on page [47](#))

- [62] Ilya Feige and Matthew D. Schwartz. An on-shell approach to factorization. *Phys. Rev. D*, 88:065021, 2013. (Cited on page [47](#))
- [63] A. Sherstinsky. Fundamentals of Recurrent Neural Network and Long Short-Term Memory Network. *Elsevier BV*, 404, 2020. (Cited on page [49](#))
- [64] A.J. Larkoski, I. Moult, and B. Nachman. Jet Substructure at the Large Hadron Collider: A Review of Recent Advances in Theory and Machine Learning. *Phys. Rept.*, 841, 2020. (Cited on page [53](#))
- [65] Jerzy Neyman and Egon Sharpe Pearson. On the Problem of the Most Efficient Tests of Statistical Hypotheses. *Phil. Trans. Roy. Soc. Lond. A*, 231(694-706):289–337, 1933. (Cited on pages [54](#) and [107](#))
- [66] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13, 2012. (Cited on page [65](#))
- [67] Sam Smith, Pieter jan Kindermans, Chris Ying, and Quoc V. Le. Don’t decay the learning rate, increase the batch size. In *ICLR 2018*, 2018. (Cited on page [72](#))
- [68] Fengxiang He, T. Liu, and Dacheng Tao. Control batch size and learning rate to generalize well: Theoretical and empirical evidence. In *NIPS*, 2019. (Cited on page [72](#))
- [69] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS, Deep Learning and Representation Learning Workshop*, 2014. (Cited on page [106](#))

Appendices

A.1 Overview of the Project Codebase

This section aims to deliver a brief overview of the implemented codebase to carry out the project. A global overview of the main programs is presented in figure [A.1.1](#). This exposes the meta-structure of the interaction between the user (top black arrows, either directly running the *Main* or submitting the job to HTCondor), the *Main*, and the process ensuing. What process is to be run and how depends on the parameters the user sets in the *base_config.yaml* file, though some of these can be dynamically modified by parsing arguments when calling the main (this is however not possible when using HTCondor).

If the user is implementing an ML model, the thick dark blue arrow is followed, to initiate a *runner* for either: Junipr (unary or binary), a BDT, a DNN, or a set of pre-trained BDT and/or DNN models. The latter runner was used for the ROC curves from models trained on sets 2 and 3 (figures [4.1.1](#) and [4.1.2](#)). The user also has the choice of running other methods as indicated by the orange arrows: either the *UpRootProcessing*, to produce the HDF5 datasets of sets 2 or 3, or the *GranularUpRootTransformer*, to produce the Junipr *json* dataset, calling *JuniprProcessing*, of sets 4, 5, 5b, and the HDF5 of set 6.

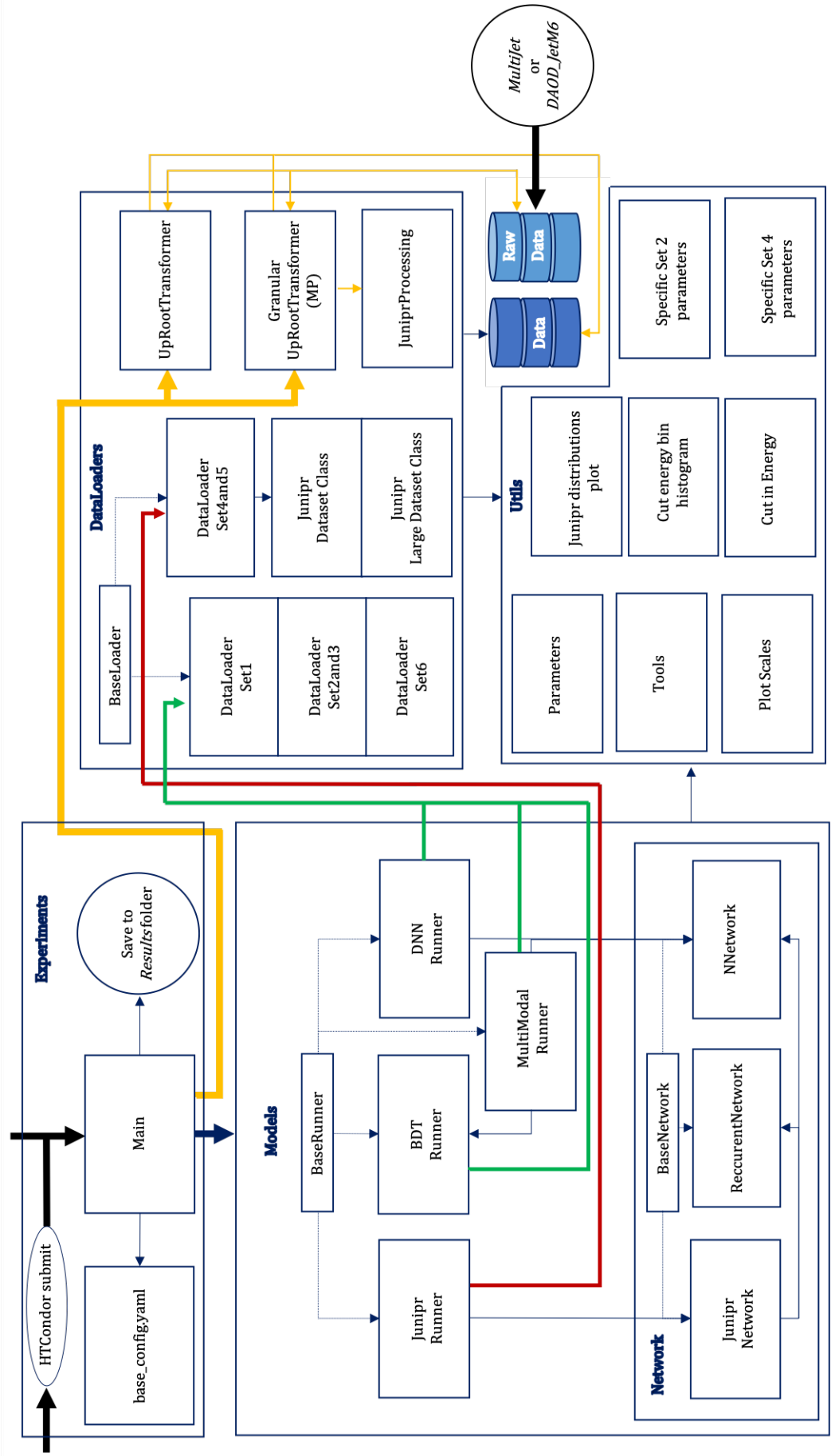


Figure A.1.1 – The architecture of the main codebase. Dashed arrows indicate class inheritance while continuous arrows point towards the component being called. Except the `base_config.yaml`, every components in a small blue frame is written in Python.

Going back to the case where a ML model is running, the first thing to be activated is the *runner*. This component is the one interacting with the main. In the case of Junipr or a DNN, it sets up the network by implementing the required components from the subfolder *Network* containing the different types of networks. The elementary networks are the *NNetwork* (implementing a DNN) and the *RecurrentNetwork* (implementing either a RNN or an LSTM). The *JuniprNetwork* aggregates these components as needed. The BDT runner is an exception, the network and runner being combined in the *BDTRunner* class. The runner having their network accessible, they can either train (for the BDT, it consists in the *fit* function of sklearn), test, save, or load a network. To access data, all models implement a dataloader from the eponymous folder. BDT and DNN have access to the dataloaders of sets 1, 2 and 3, or 6 (green arrow) while Junipr has access to the unique dataloader of set 4 and 5 (red arrow). A dataloader is a subclass of a generic *BaseLoader*. It opens the appropriate files, performs the required operations to combine the data, and returns access to this data either by a dataset (BDT and DNN) or as locally inherited PyTorch Junipr Dataset class (*JuniprDatasetClass* or *JuniprLargeDatasetClass*, the latter working on chunked data, as specified in section 3.2.3.4).

The *Utils* folder contains various methods used throughout this project, such as text loggers, ROC plotters, confusion matrix plotters, maps plotters, etc. It also contains lists of variables and pre-defined sets of parameters to ease the plotting of distributions of interest. The *Parameters* class is a set of tools to open, read, save, and format configuration files from the *base_config.yaml*. This python file was directly copied from a previous project in which I took part (mini-project in Advanced Topics in Machine Learning). Note that there are two implementations of *GranularUpRootTransformer*: one is made to run normally and one is a standalone designed for multiprocessing and launched from an alternative main called *Submitter_Condor_MultiProcessor* run by HTCondor to process the Junipr jets (not shown on the diagram).

This concludes the global overview of the main codebase. A secondary and independent codebase called *GranularDataGatherer* implements an Athena EventLoop algorithm to process the raw level ROOT datafiles into ROOT files that can be fed to *GranularUp-*

RootTransformer to form either sets 4, 5, 5b, or 6. It is based on three components:

1. A *build* folder, where the ATLAS softwares produces the makefiles of the code. The AnalysisBase version is also stored there and can be re-called. This folder was emptied at submission time.
2. A *run* folder, from which the EventLoop algorithm is launched and where the results are stored. This folder was emptied at submission time.
3. A *source* folder, where the C++ Algorithm and Runner are written. This is the heart of the codebase. CMakeList define the list of files to compile when building the program. In the *MyAnalysis* folder, the first folder, also named *MyAnalysis*, contains the header of the Algorithm and defines some of the ATLAS tools to be deployed as well as the variables to be populated when looping through the events. The *Root* folder defines the Algorithm itself and what to do at the initialisation, running, and finalisation phases. The middle phase is where the event analysis occurs and where the data is extracted, processed, reconstructed, and accumulated. This whole choreography is governed by the python file in the *share* folder. This wrapper points to the files to analyse (whether locally or on the GRID), initiate a driver (whether to run locally or on the GRID) and launch the algorithm. The paths were anonymised at submission time.

B.1 Diagnostic Plots of Sets 4 and 6

This appendix introduces additional plots of the engineered variables contained in sets 4 and 6. In particular, it reveals the distributions of all variables used for the BDT and DNN as well as some inside information on the energy distribution of the constituents of the jet. Figure [B.1.1](#) first explains why jets with less than five constituents are removed. This is adopted in order to get rid of the peak at 3-4 constituents that is typically due to taus (often decaying to three components) or muons. Taus and muons are leptons, as indicated in figure [2.1.1](#), and, therefore, cannot hadronise to form jets. They are mistaken for jets because they leave a large enough signal in the calorimeters.

The cut removing jets not associated with the primary vertex (“isNotPVJet = 1” in table [3.1.4](#)), is motivated by observing the distribution of the sum of tracks p_T above 500 GeV: $p = \sum_{i \in \text{tracks}} p_{T,\text{track}}^{500,i}$. As displayed in the left of figure [B.1.2](#) (the distribution of the variable with all cuts but the “isNotPVJet = 1” one), a strong peak is present at the null value, meaning many jets have no track above 500 GeV. Some of these are expected but an additional bias seems to have plagued the distribution: the pile-up. This indeed adds many events with a low number of tracks above 500 GeV (due to poor reconstruction of

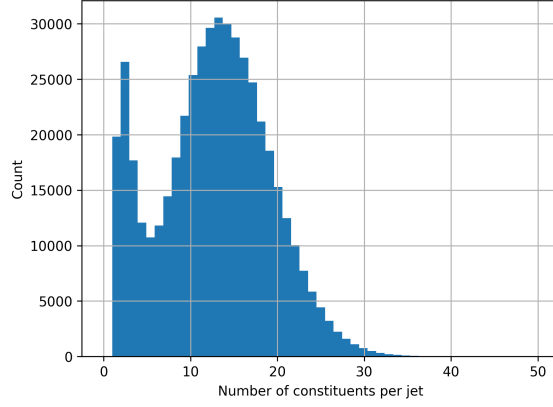


Figure B.1.1 – The number of constituents in a portion of the $t\bar{t}$ signal used for sets 4 and 6 before any cut.

the side event). Figure B.1.2 shows how this effect can be mitigated by adding the cut removing jets not matched to the PV (shown on the right). Its effect is clear: it mainly targets jets with a value of 0 GeV (first bin) and those with a value closed to 500 GeV (removing the small secondary peak around that value on the left plot). These are indeed likely to be PU jets.

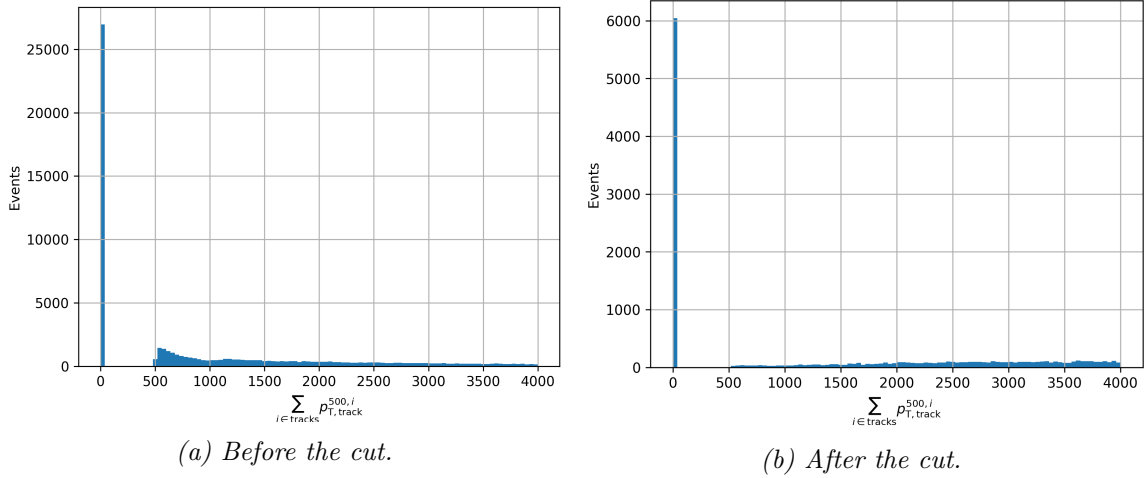


Figure B.1.2 – Distribution of the sum of track momentum (for tracks with a $p_T > 500$ GeV) for a portion of the $t\bar{t}$ signal used for sets 4 and 6, with all but the PV cut applied.

The 10 next pairs of plots present the rest (those not described in the main part of the text) of the jet variables used for the BDT and DNN trained on set 6, with the gluon distributions on the left and the quark ones on the right. They are:

- Figure B.1.4 shows the jet η distributions.

- Figure B.1.4 shows the jet ϕ distributions.
- Figure B.1.5 shows the jet mass distribution.
- Figure B.1.6 shows the fraction of jet energy collected in the electromagnetic calorimeter.
- Figure B.1.7 shows the fraction of jet p_T associated to charged particles.
- Figure B.1.8 shows the distribution of number of tracks with a p_T above 500 GeV.
- Figure B.1.9 shows the same distribution for p_T above 1000 GeV.

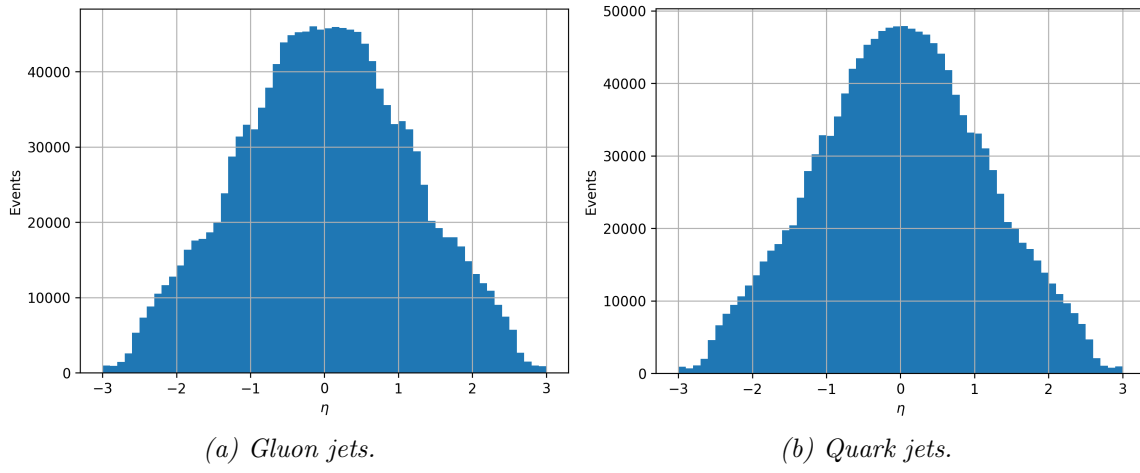


Figure B.1.3 – The distribution of jet η , sets 4 and 6.

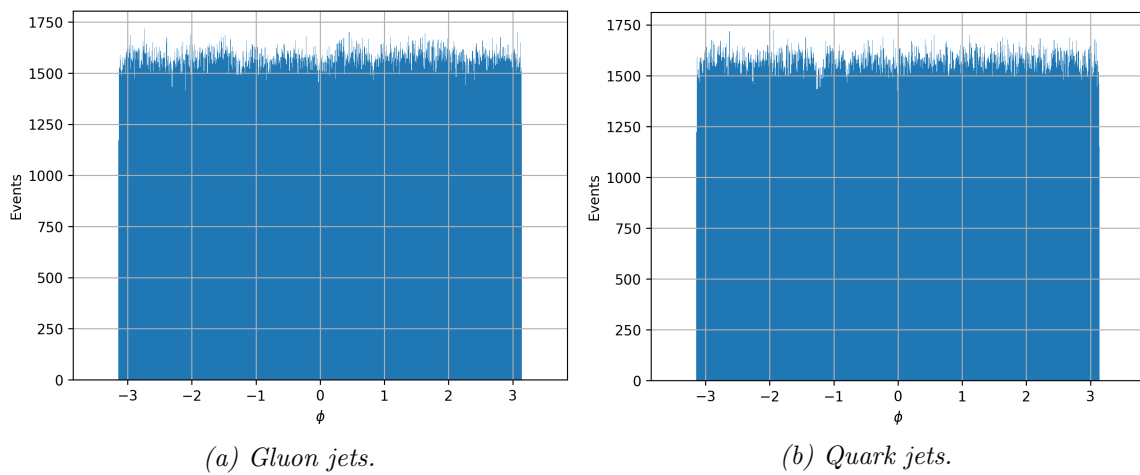


Figure B.1.4 – The distribution of jet ϕ , sets 4 and 6.

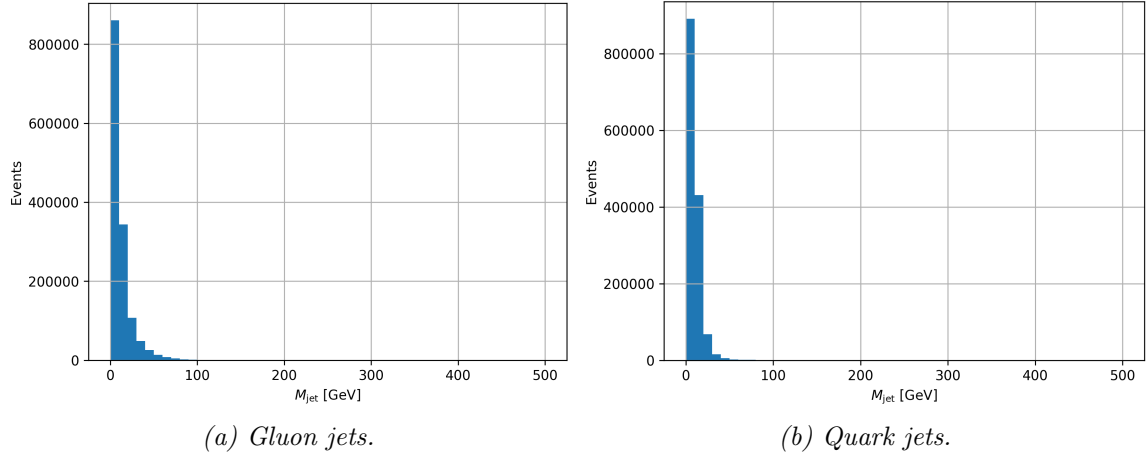


Figure B.1.5 – The distribution of jet mass, sets 4 and 6.

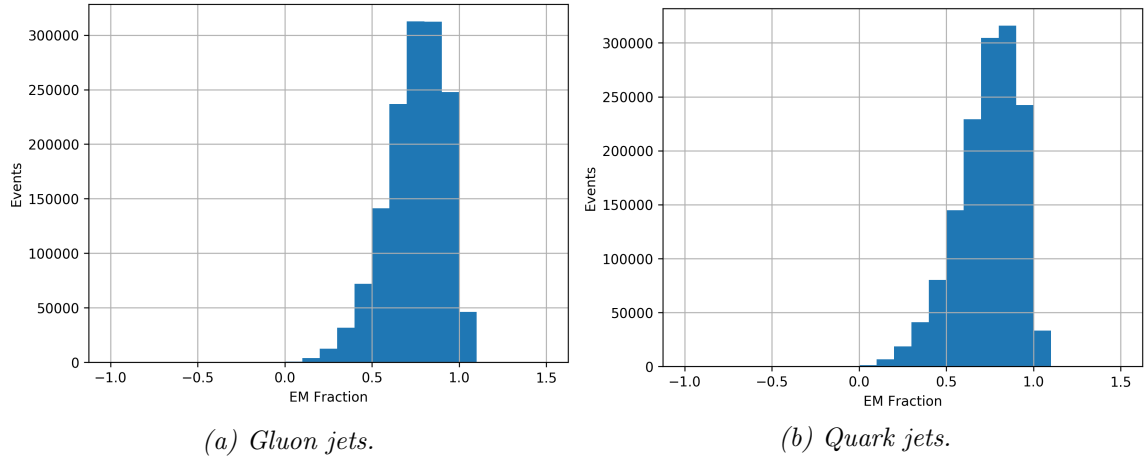
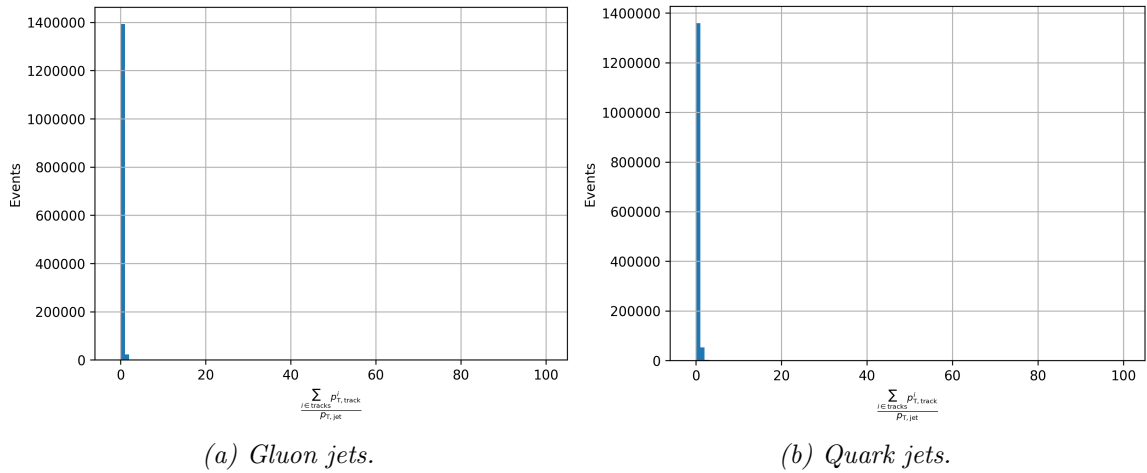


Figure B.1.6 – The distribution of the fraction of jet energy collected in the electromagnetic calorimeter, sets 4 and 6.

Figure B.1.7 – The distribution of the fraction of jet p_T carried by charged particles (ChFrac), sets 4 and 6.

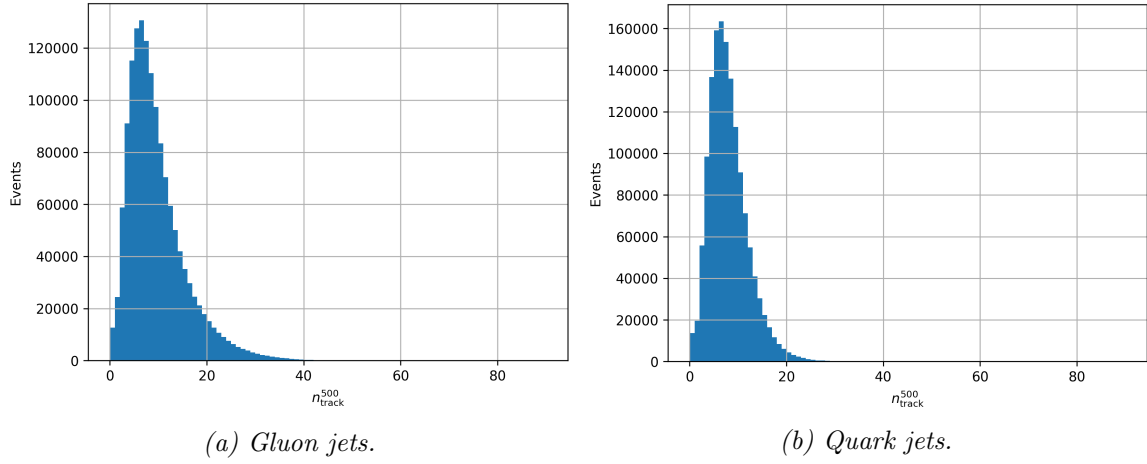


Figure B.1.8 – The distribution of number of tracks with a p_T above 500 GeV, sets 4 and 6.

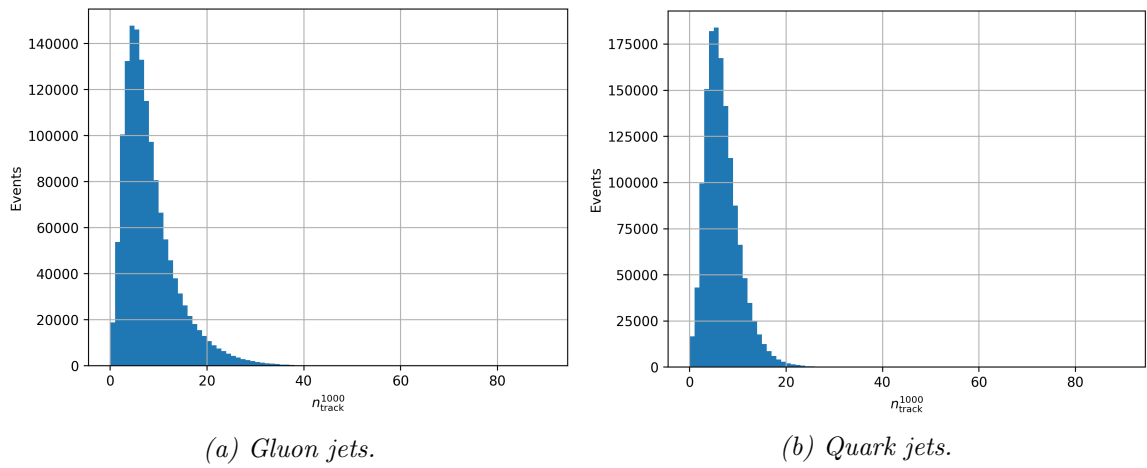


Figure B.1.9 – The distribution of number of tracks with a p_T above 1000 GeV, sets 4 and 6.

Additional information on the Junipr Model

C.1 Additional Diagrams of Junipr Analysis

Figure C.1.1 represents the analysis of the first timestep by the Junipr model, in a fashion similar to that of figure 3.2.2. Figure C.1.2 presents the last timestep analysis (for a jet of n final state constituents, this is timestep n).

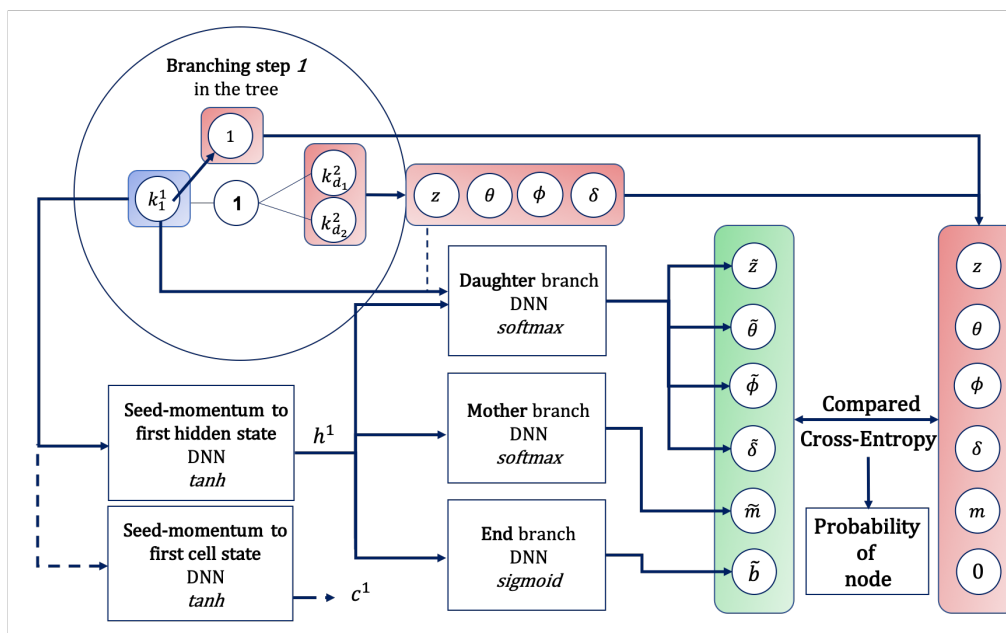


Figure C.1.1 – Diagrammatic representation of the transfer of information for the Junipr model processing at timestep 1. Information on this diagram is similar to that introduced in figure 3.2.2. The dashed arrow is only used for the LSTM recurrent core.

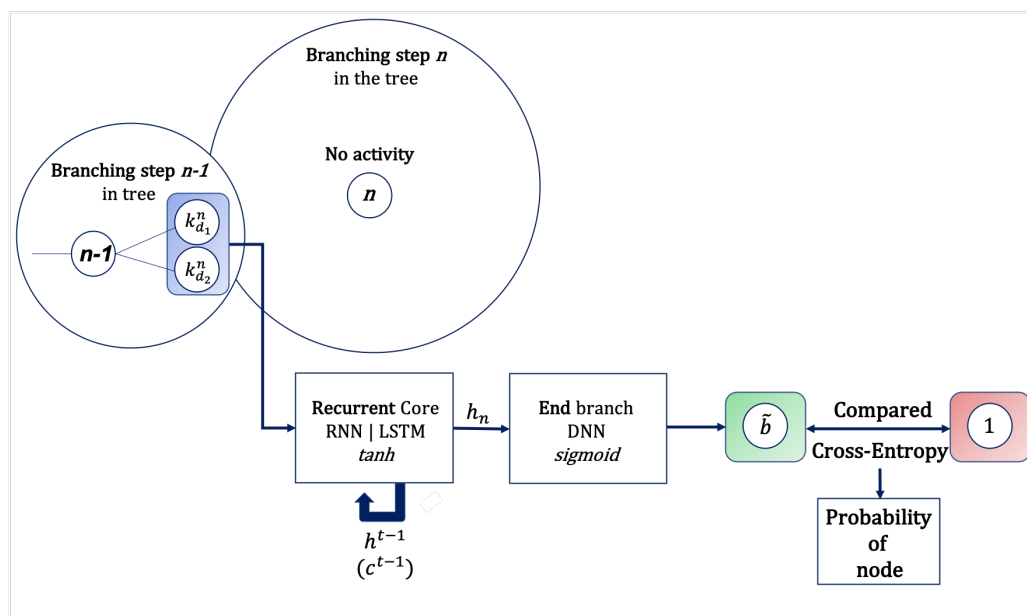


Figure C.1.2 – Diagrammatic representation of the transfer of information for the Junipr model processing at the final timestep (n). Information on this diagram is similar to that introduced in figure 3.2.2.

C.2 Information on the Trained Junipr Models

This section summarises the different Junipr models trained and lists some additional training/learning curves. Table C.2.1 lists all final Junipr models trained. All Junipr models trained have a hidden state of size 30 and 10 units per hidden layer in every DNN, except models $UJ\ Q4$, $UJ\ G4$, and $BJ\ 4$ as specified in the table. All models use an LSTM recurrent cell, granularity = 10, the multi-branch implementation of the *daughter branch*, and all binary models deploy the proposed objective from section 3.2.3.3, equation 3.2.9 (using a binary cross entropy loss between the score and the true label). The training schedule for unary Junipr are presented in tables 4.2.1, 4.2.2, and 4.4.1 (corresponding respectively to the normal, extended and the long extension) while the training schedule for binary Junipr is available in table 4.3.1. Any longer or shorter version increases or reduces the schedule from the right (so the 26 epochs version adds 7 epochs to the last leg of training while the 14 epochs versions removes the last 5 epochs).

Figure C.2.1 displays the training curves of the quark and gluon unary Junipr models on set 5, trained for 16 epochs with the extended schedule of tables 4.2.1 and 4.2.2. Figure C.2.2 than displays the learning curves of the binary Junipr model trained for 26 epochs using these 16-epoch unary models with an extension of the training schedule from table 4.3.1 (adding 7 epochs to the last leg). Figures C.2.3 and C.2.4 display the same quadruple of plots for the pair of unary models ($UJ\ Q3$ and $UJ\ G3$) and the binary model ($BJ\ 3$) trained on the [100, 200] GeV energy bin of set 5, with training schedules of tables 4.4.1 and 4.3.1 (the latter is shorten to 14 epochs by removing the adequate number of final steps) respectively. Figure C.2.5 and C.2.6 display the same information for the larger models $UJ\ Q4$, $UJ\ G4$, and $BJ\ 4$.

The last set of Junipr models indicated here, $UJ\ Q5$, $UJ\ G5$, and $BJ\ 5$, are trained on set 5b. This set is in all way similar to set 5 except it is made of Truth jets instead of EMTopoJets. Therefore, the final state constituents are particle and not calorimeter topoclusters. They are also recombined with anti- k_T (radius 0.4). The training curves are displayed in figures C.2.7 and C.2.8.

Name	Unary / Binary	Dataset	Quark / Gluon	Training Time	AUC	General Remarks
UJ Q1	U	Set 4	Q	~ 45 h	/	16 epochs with extended training schedule (figure 4.2.2).
UJ G1	U	Set 4	G	~ 45 h	/	16 epochs with extended training schedule (figure 4.2.2).
UJ Q2	U	Set 5	Q	~ 45 h	/	16 epochs with extended training schedule (figure C.2.1).
UJ G2	U	Set 5	G	~ 45 h	/	16 epochs with extended training schedule (figure C.2.1).
UJ Q3	U	Set 5 energy bin [100, 200] GeV	Q	~ 22 h	/	50 epochs with the training schedule of table 4.4.1 (figure C.2.2).
UJ G3	U	Set 5 energy bin [100, 200] GeV	G	~ 22 h	/	50 epochs with the training schedule of table 4.4.1 (figure C.2.4).
UJ Q4	U	Set 5	Q	~ 60 h	/	16 epochs with extended training schedule. Larger model: size of hidden state h is 40 and all DNN hidden layer have 20 units (figure C.2.5).
UJ G4	U	Set 5	G	~ 60 h	/	16 epochs with extended training schedule. Larger model: size of hidden state h is 40 and all DNN hidden layer have 20 units (figure C.2.5).
UJ Q5	U	Set 5b (Truth)	Q	~ 45 h	/	16 epochs with extended training schedule (figure C.2.7).
UJ G5	U	Set 5b (Truth)	G	~ 45 h	/	16 epochs with extended training schedule (figure C.2.7).
BJ 1	B	Set 4	Q / G	~ 65 h	0.672	19 epochs training schedule (figure 4.3.1).
BJ 2	B	Set 5	Q / G	~ 90 h	0.690	26 epochs training schedule (figure C.2.2).
BJ 3	B	Set 5 energy bin [100, 200] GeV	Q / G	~ 26 h	0.643	14 epochs training schedule (figure C.2.4).
BJ 4	B	Set 5	Q / G	~ 129 h	0.690	26 epochs training schedule (figure C.2.6).
BJ 5	B	Set 5b (Truth)	Q / G	~ 100 h	0.693	26 epochs training schedule (figure C.2.8).

Table C.2.1 – All Junipr models trained and used in the analysis. Binary Junipr model BJ X relies on the unary models UJ Q X and UJ G X .

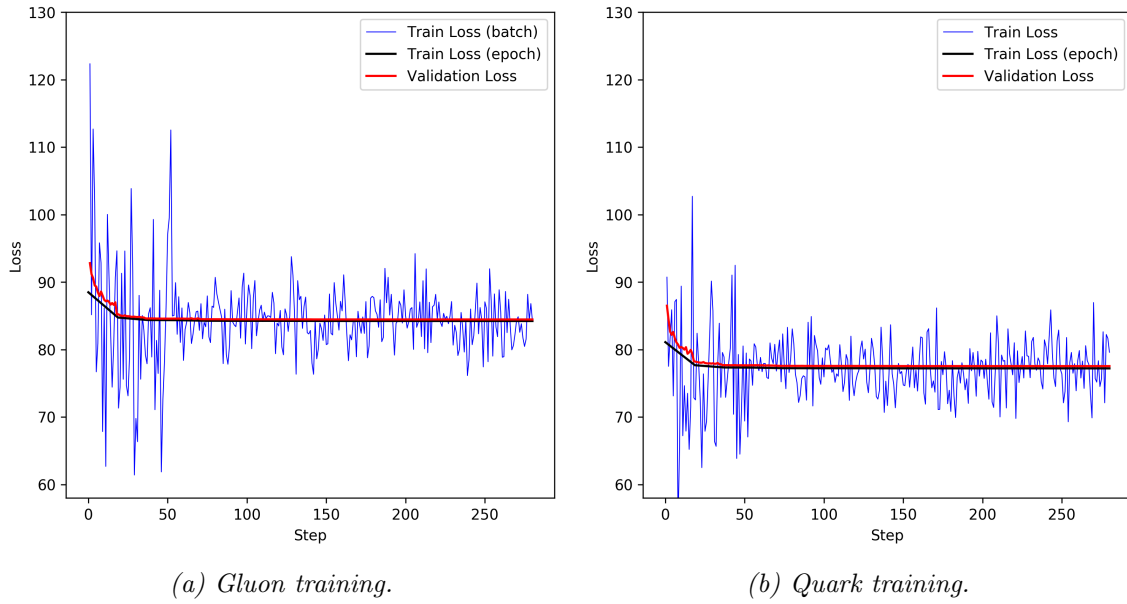


Figure C.2.1 – The learning curves of the gluon and quark Junipr models trained for 16 epochs each on set 5 (UJ Q2 and UJ G2). The blue curve is the loss of a random batch at the step indicated, the black curve is the interpolated loss of the epoch, and the red curve is the loss on the validation set. One step corresponds to processing 80,000 jets.

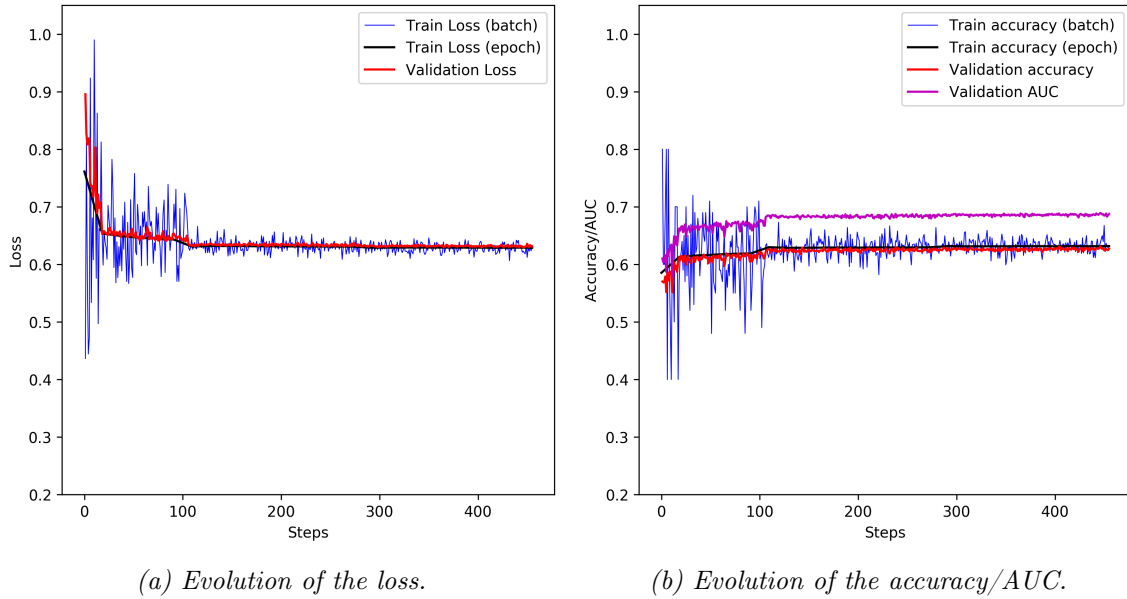


Figure C.2.2 – The learning curves of the 26 epochs binary Junipr model BJ 2 trained with the 16 epochs unary models trained on set 5 (UJ Q2 and UJ G2). The blue curve is the lost/accuracy of a random batch at the step indicated, the black curve is the lost/accuracy of the epoch, the red curve is the same information on the validation set, and the magenta curve is the AUC evaluated on the validation set. One step corresponds to processing 160,000 jets.

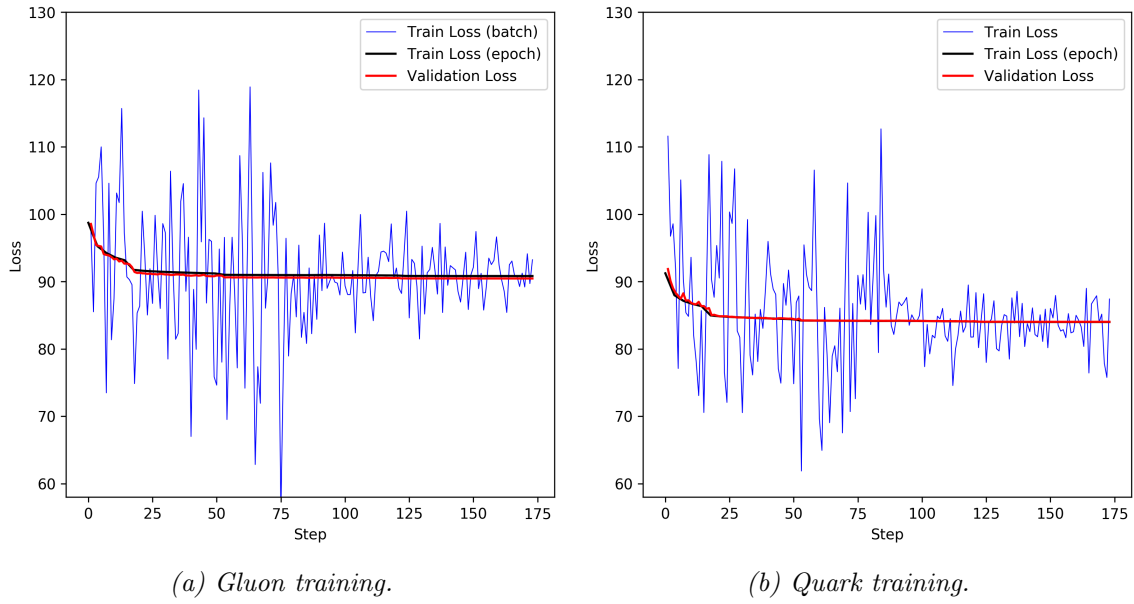


Figure C.2.3 – The learning curves of the gluon and quark Junipr models trained for 50 epochs each on the $[100, 200]$ GeV energy bin of set 5 (UJ Q3 and UJ G3). The blue curve is the loss of a random batch at the step indicated, the black curve is the interpolated loss of the epoch, and the red curve is the loss on the validation set. One step corresponds to processing 80,000 jets.

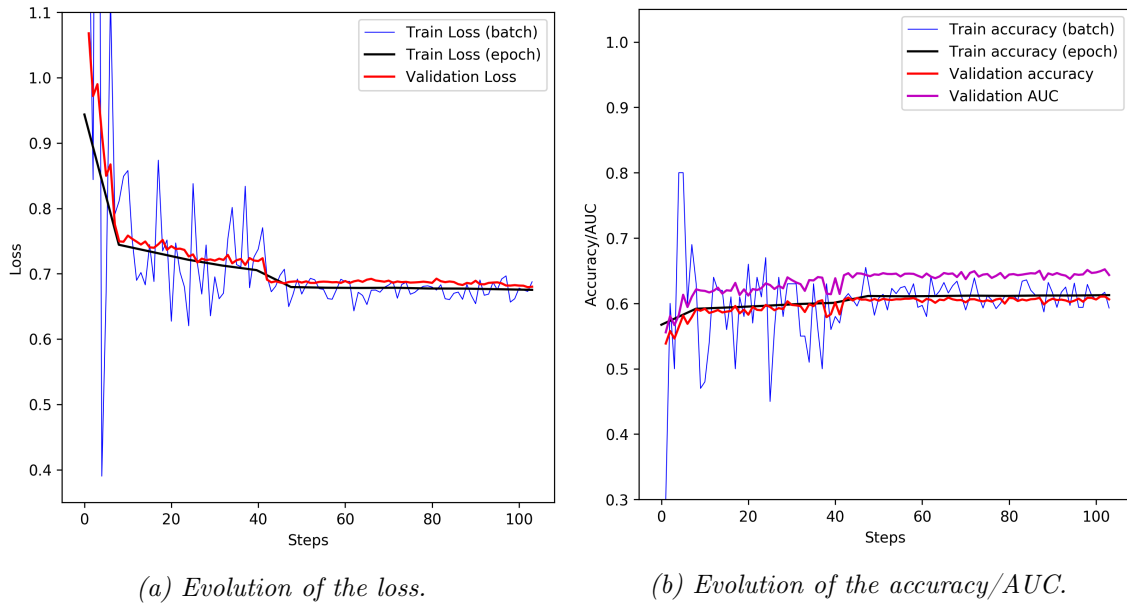


Figure C.2.4 – The learning curves of the 14 epochs binary Junipr model BJ 3 trained using the 50 epochs unary models trained on the $[100, 200]$ GeV energy bin of set 5 (UJ Q3 and UJ G3). The blue curve is the lost/accuracy of a random batch at the step indicated, the black curve is the lost/accuracy of the epoch, the red curve is the same information on the validation set, and the magenta curve is the AUC evaluated on the validation set. One step corresponds to processing 497 jets.

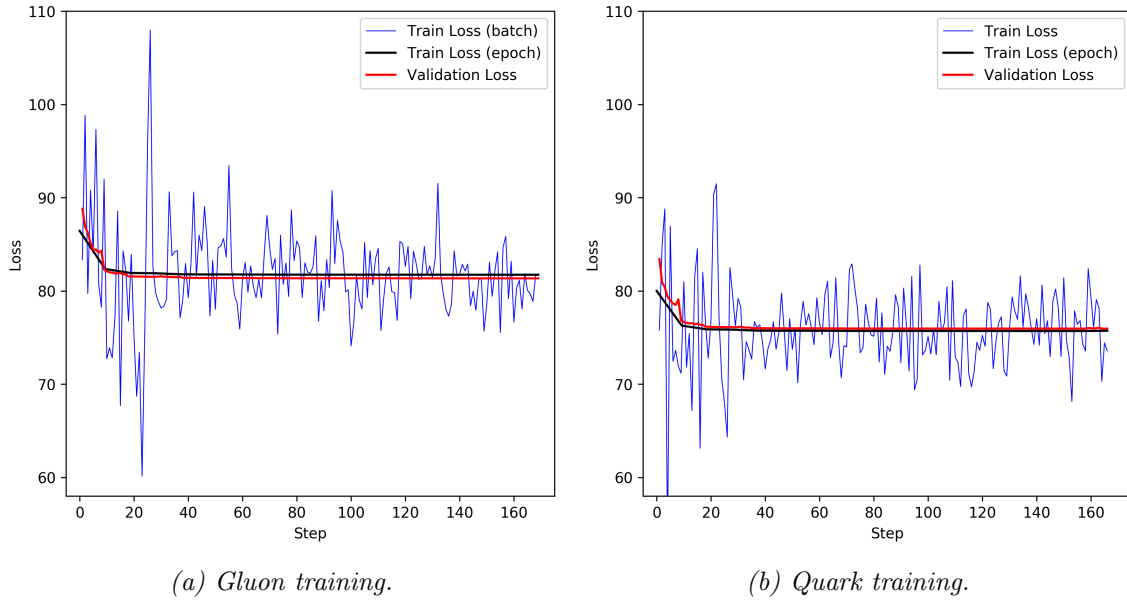


Figure C.2.5 – The learning curves of the gluon and quark large Junipr models trained for 16 epochs on set 5 (UJ Q4 and UJ G4). The blue curve is the loss of a random batch at the step indicated, the black curve is the interpolated loss of the epoch, and the red curve is the loss on the validation set. One step corresponds to processing 80,000 jets.

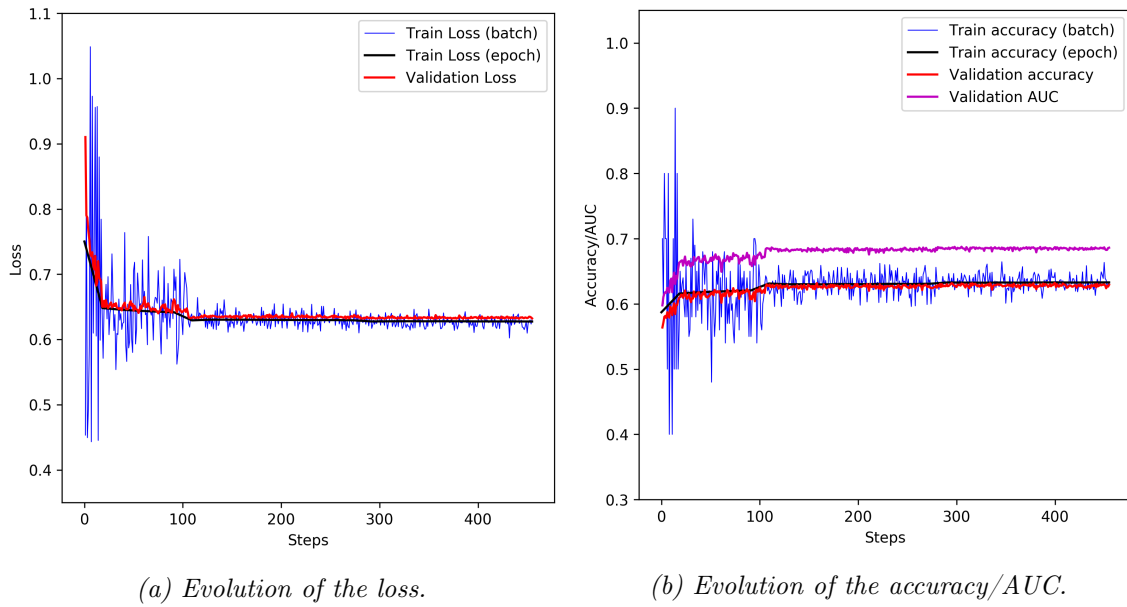


Figure C.2.6 – The learning curves of the 26 epochs large binary Junipr model BJ 4 trained using the 16 epochs unary models trained on set 5 (UJ Q4 and UJ G4). The blue curve is the loss/accuracy of a random batch at the step indicated, the black curve is the loss/accuracy of the epoch, the red curve is the same information on the validation set, and the magenta curve is the AUC evaluated on the validation set. One step corresponds to processing 160,000 jets.

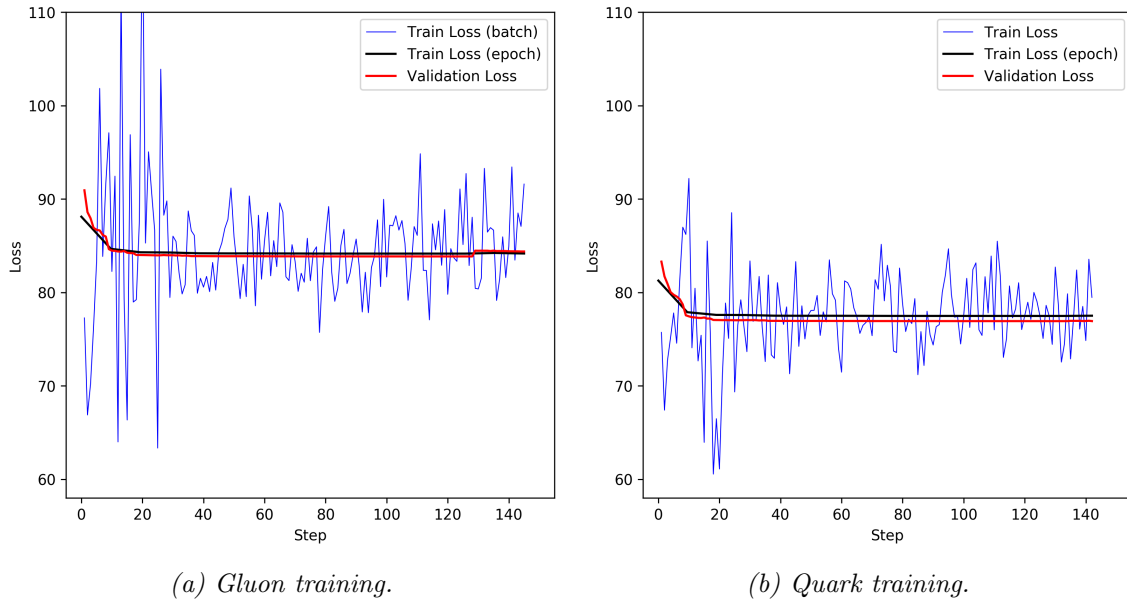


Figure C.2.7 – The learning curves of the gluon and quark large Junipr models trained for 16 epochs on set 5b (UJ Q5 and UJ G5). The blue curve is the loss of a random batch at the step indicated, the black curve is the interpolated loss of the epoch, and the red curve is the loss on the validation set. One step corresponds to processing 80,000 jets.

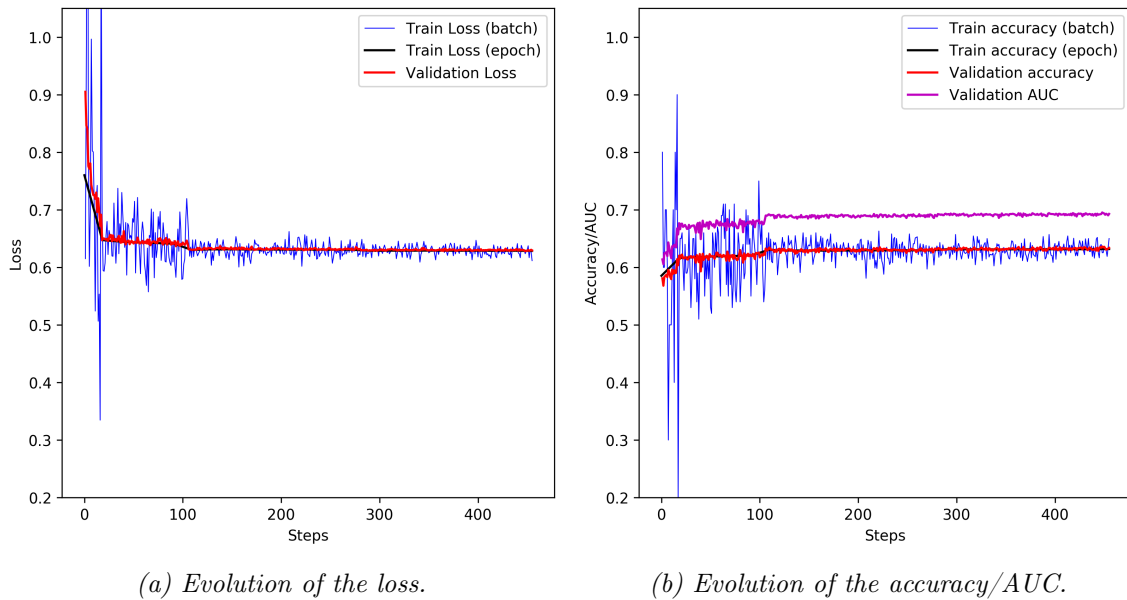


Figure C.2.8 – The learning curves of the 26 epochs large binary Junipr model BJ 5 trained using the 16 epochs unary models trained on set 5b (UJ Q5 and UJ G5). The blue curve is the loss/accuracy of a random batch at the step indicated, the black curve is the loss/accuracy of the epoch, the red curve is the same information on the validation set, and the magenta curve is the AUC evaluated on the validation set. One step corresponds to processing 160,000 jets.

C.3 Scaling Function for the Junipr model

This section presents the scaling function used for the Junipr input data. The one proposed in the Junipr paper [8] to map the real variables to the range $[0, 1]$ is the first to be presented. Their scaling relies on the following parameters:

- $E_{jet} \sim 500$ GeV, the centre of the energy window, and $R_{jet} \sim \pi/2$, the average jet radius.
- $E_{sub} = 1$ GeV, the lower limit of constituent energy, and $R_{sub} = 0.1$, supposedly the lower limit on the angle between constituents (though this is not explained and is not always used).

The branching information is then scaled using the following four equations:

$$\begin{aligned}
 \tilde{z} &= \frac{\log z - \log \frac{E_{sub}}{E_{jet}}}{\log \frac{1}{2} - \log \frac{E_{sub}}{E_{jet}}} \\
 \tilde{\theta} &= \frac{\log \theta - \log \frac{R_{sub}}{2}}{\log R_{jet} - \log \frac{R_{sub}}{2}} \\
 \tilde{\phi} &= \frac{\phi}{2\pi} \\
 \tilde{\delta} &= \frac{\log \delta - \log \frac{E_{sub} R_{sub}}{E_{jet}}}{\log \frac{R_{jet}}{2} - \log \frac{E_{sub} R_{sub}}{E_{jet}}}
 \end{aligned} \tag{C.3.1}$$

The scaling functions used in the local implementation of Junipr are different, given the lack of a centralised phase-space configuration. It relies on the function presented in equation 3.2.10. Table C.3.1 list the pair of parameters for each variables, except ϕ which is simply linearly scaled $\tilde{\phi} = \frac{\phi}{2\pi}$ since the distribution is uniform (the physics involved in the project is insensitive to ϕ , benefiting from a symmetry). The values were selected to have bin limits matching the typically observed data. Figures C.3.1, C.3.2, C.3.3, C.3.4, and C.3.5 present the scaling functions in the order adopted by the table. When light grey horizontal lines are present, they mark the limit of the bins if the granularity is set to 10.

Variable	α_{Variable}	Variable _{limit}	Target
Energy	10^{-1} GeV^{-1}	700 GeV	Mother and daughter momenta
Mass	0.2 GeV^{-1}	50 GeV	Mother and daughter momenta
z	2	0.5	Branching
θ	50	π	Branching + mother and daughter momenta
δ	5000	π	Branching

Table C.3.1 – Chosen parameter values for each variable rescaled (except ϕ).

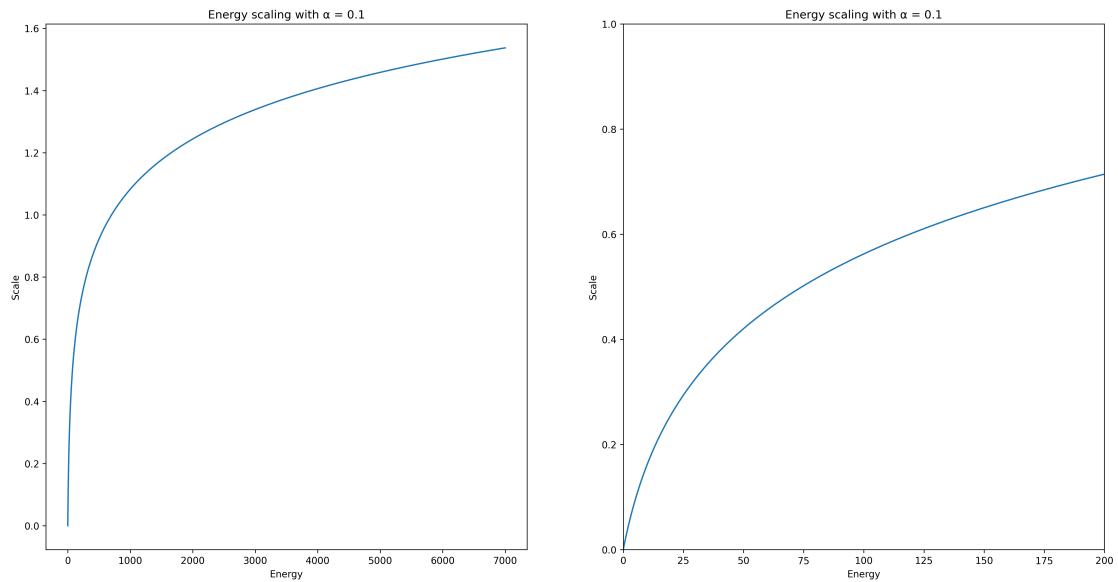


Figure C.3.1 – Scaling function for the energy.

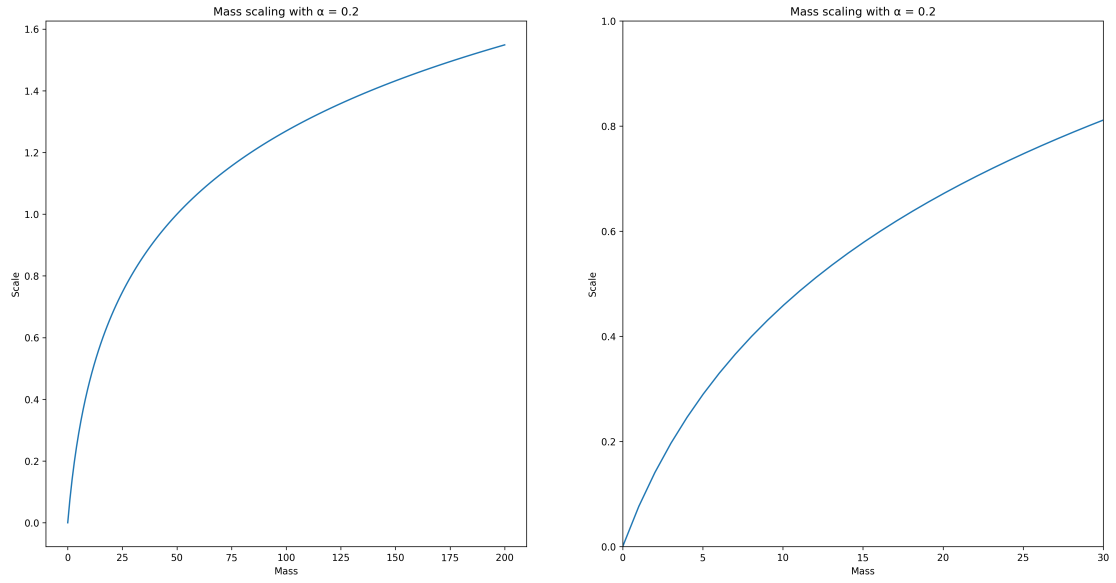


Figure C.3.2 – Scaling function for the mass.

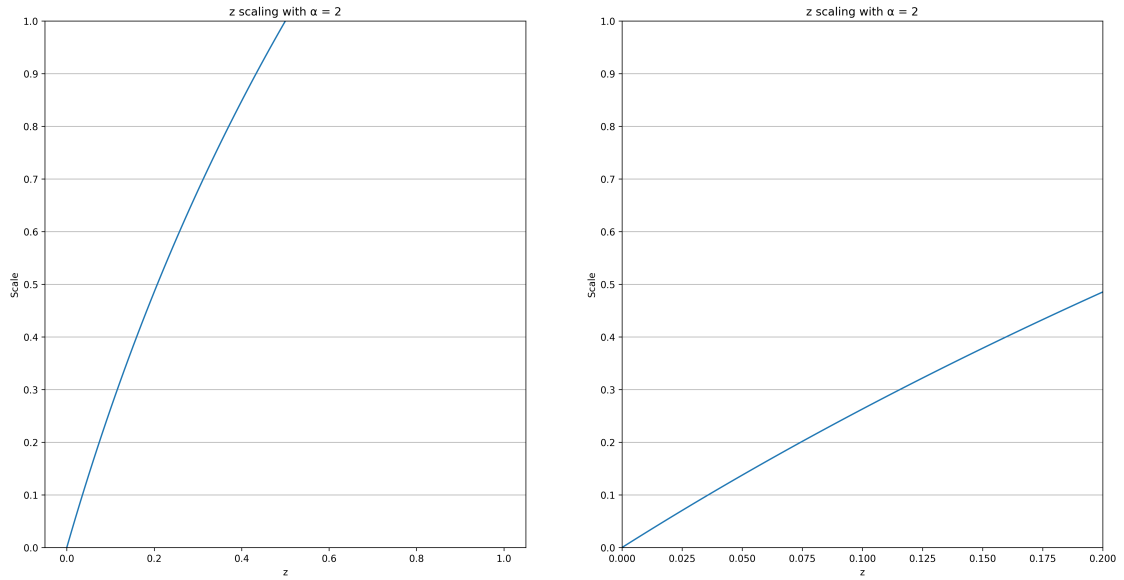


Figure C.3.3 – Scaling function for the soft daughter energy fraction z .

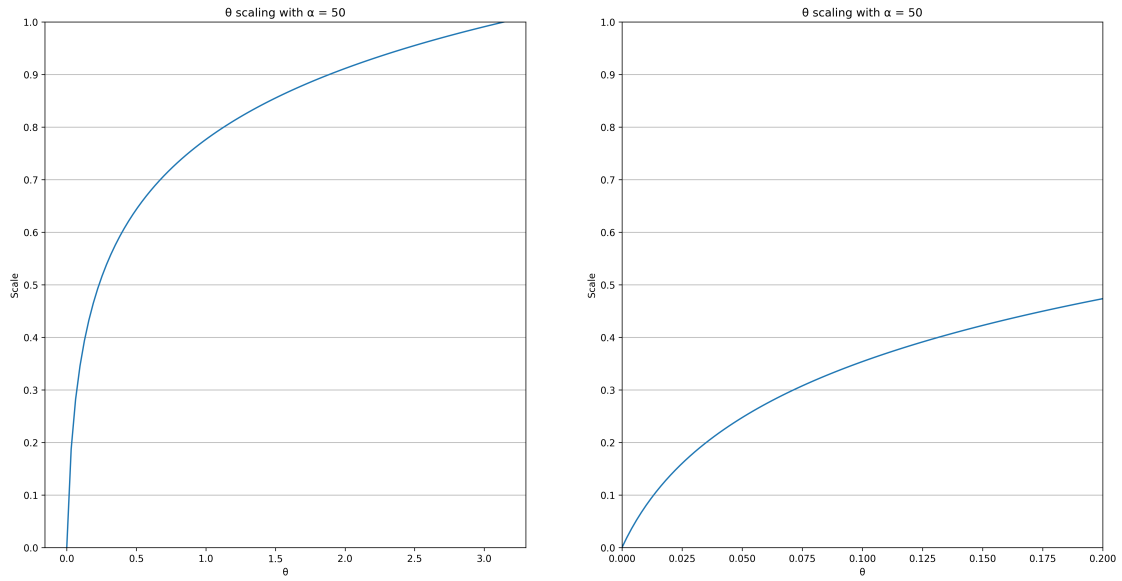


Figure C.3.4 – Scaling function for the angle between the soft daughter and the mother and the θ angle in a momentum (between the particle and the seed-momentum).

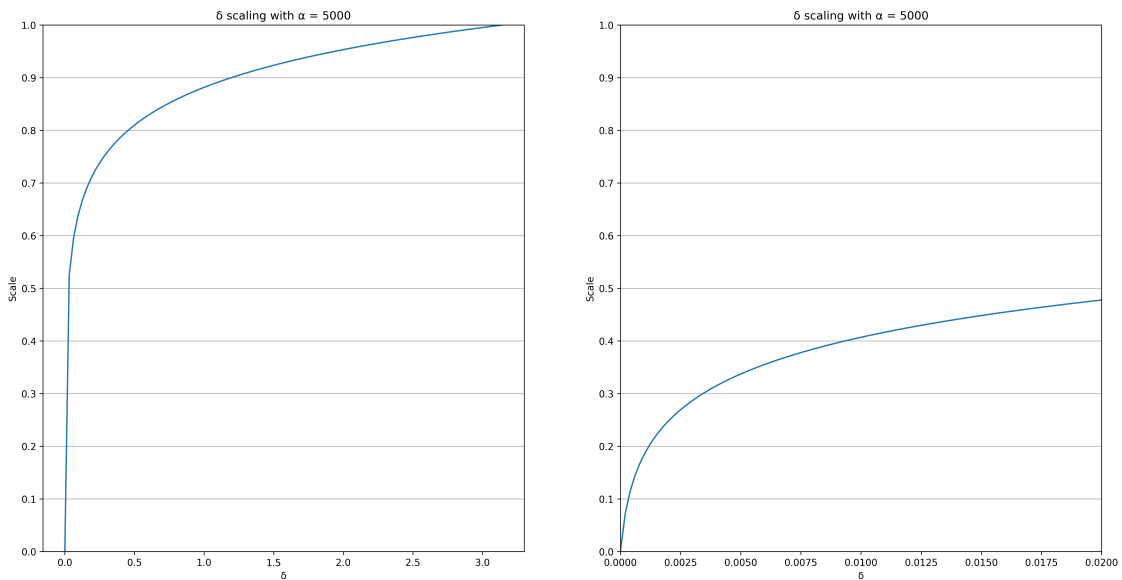


Figure C.3.5 – Scaling function for the angle between the hard daughter and the mother δ .

C.4 Additional Junipr Trees

Figure C.4.1 presents several unary Junipr jet trees for the quark (top) and gluon (bottom) unary models trained for 16 epochs on set 4 (*UJ Q1* and *UJ G1*, as described in section 4.2). Figure C.4.2 presents some binary Junipr jet trees as analysed by *BJ 2* in different energy regions of set 5.

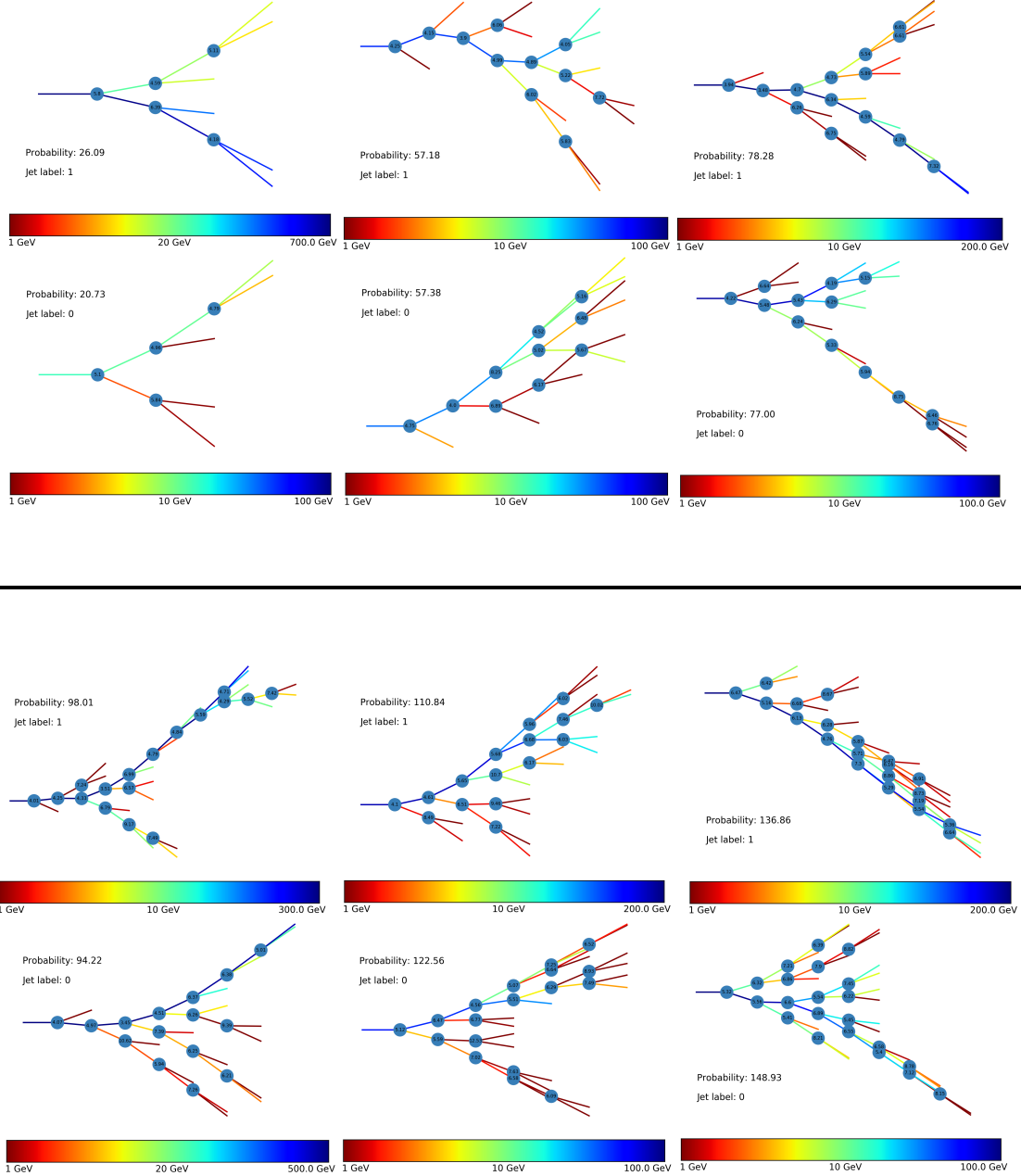


Figure C.4.1 – Some unary Junipr jets obtained from the 16 epochs trained models on set 4. Even rows are from the quark model (*UJ Q1*), odd rows from from the gluon model (*UJ G1*). Each column presents jets of close energy for the two top or the two bottom rows (separated by the black line).

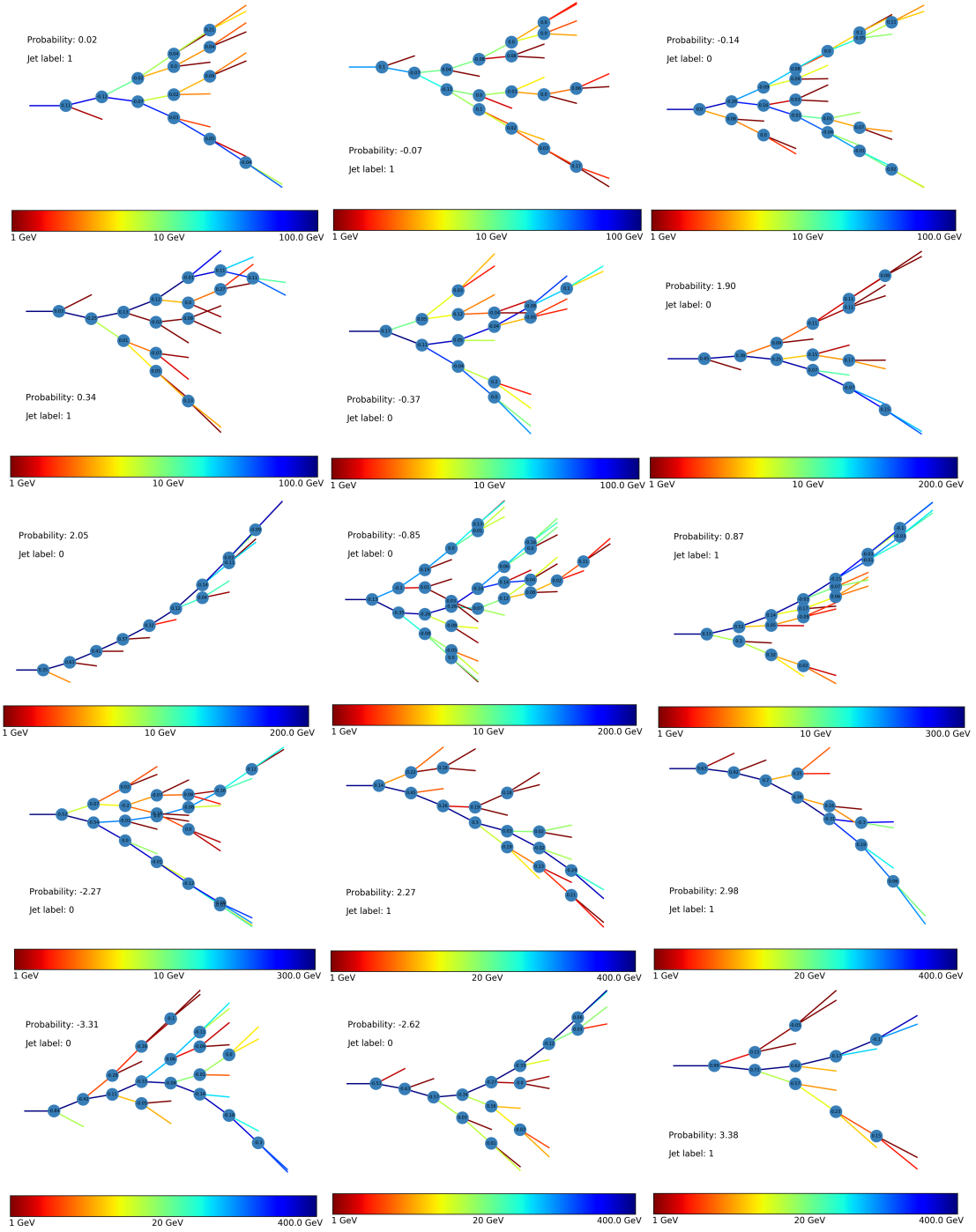


Figure C.4.2 – Some binary Junipr jets obtained from model BJ 2 on set 5. The top row corresponds to $[0, 100]$ GeV, the second row to $[100, 200]$ GeV, the third to $[200, 300]$ GeV, the fourth to $[300, 400]$ GeV, and the final fifth to $[400, 500]$ GeV.