Identification of the initial jet parton using machine learning techniques at the ALICE detector

Jerčić, Marko

University of Zagreb

FACULTY OF SCIENCE
DEPARTMENT OF PHYSICS

Marko Jerčić

IDENTIFICATION OF THE INITIAL JET PARTON USING MACHINE LEARNING TECHNIQUES AT THE ALICE DETECTOR

DOCTORAL THESIS

Zagreb, 2023

University of Zagreb

FACULTY OF SCIENCE
DEPARTMENT OF PHYSICS

Marko Jerčić

IDENTIFICATION OF THE INITIAL JET PARTON USING MACHINE LEARNING TECHNIQUES AT THE ALICE DETECTOR

DOCTORAL THESIS

Supervisor:
Assoc. Prof. Nikola Poljak

Zagreb, 2023

Sveučilište u Zagrebu

PRIRODOSLOVNO - MATEMATIČKI FAKULTET
FIZIČKI ODSJEK

Marko Jerčić

IDENTIFIKACIJA POČETNOGA PARTONA MLAZA POMOĆU TEHNIKA STROJNOGA UČENJA NA DETEKTORU ALICE

DOKTORSKI RAD

Mentor:
izv. prof. dr. sc. Nikola Poljak

Zagreb, 2023.

UNIVERSITY OF ZAGREB
Faculty of Science
Physics Department

Doctor of Philosophy

Abstract

Identification of the initial jet parton using machine learning techniques at the ALICE detector

by Marko JERČIĆ

Keywords: *Quantum Chromodynamics, Quark, Gluon, Jets, ALICE, Machine learning, Neural network*

Quarks and gluons produced in high-energy collisions at the Large Hadron Collider (LHC) are manifested as collimated jets of particles via fragmentation and hadronisation processes. Distinguishing quark-initiated jets and gluon-initiated jets can be very useful for theoretical understanding of the fragmentation process, for better measurements of the Standard model parameters, as well as in the search for physics beyond the Standard model. Using machine learning techniques on data from the ALICE (A Large Ion Collider Experiment) experiment at the LHC, this thesis would find a discriminant which separates a quark-initiated sample of jets and a gluon-initiated sample of jets. Based on this classification, we could perform an analysis of the dependence of the fraction of quark jets in the total number of jets on the jet transverse momentum, as well as look at the barion and meson production for different jet classes.

Acknowledgements

First of all, I would like to express my sincere gratitude to my mentor, assoc. prof. Nikola Poljak for his support and guidance during my PhD study. It was not an easy task at all, and he succeeded to the end. Any problem that arose was easily solved with his assistance. Also, I would like to thank the head of my research group, prof. Mirko Planinić who gave me many valuable insights and tips regarding the direction of my research. He introduced me to various experts who helped me continue in moments of despair. He made many of my trips around the world possible in which I attended scientific conferences and workshops.

I thank my work colleagues Filip Erhardt, David Karatović and Antonija Utrobičić for their support and encouragement. The conversations with them have helped me significantly on this journey. Also, I would like to say thank you to the entire Department of Physics on an excellent working environment.

Finally, a special thanks goes to my family and friends for their motivation and support. Their help over the past few years has really meant a lot to me. Everything is easier with them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Throughout history mankind sought to learn more and more about the building blocks of our universe. This curiosity led us to the Standard Model of particle physics, currently the best theory that describes nature's constituents and their interactions. The absence of the description of gravity and recently discovered phenomena such as neutrino oscillations show us that the Standard Model is an incomplete theory and needs further improvements. Modern particle accelerators are continuously testing the Standard Model and looking for new discoveries which would help in its further development. The Large Hadron Collider (LHC) at the European Organization for Nuclear Research (CERN) is the leading particle accelerator in the world with the highest collision energies. Detectors at the LHC produce enormous amounts of data which are analyzed by an army of scientists all over the world.

Jets are collimated streams of particles produced in the high energy collisions. They can be seen as an experimental manifestation of the quarks or gluons that are created in the primary process of the collision. These quarks and gluons, commonly denoted as partons, undergo through the process of fragmentation where multiple emissions of other partons occur. During this process, the energy of partons is getting smaller and they form hadrons in the final state via hadronization process. These hadrons are detectable and scientists have access to their properties. Multiple techniques and methods are devised to cluster these hadrons in the final states into jets. The kinematic properties of the jet is then assigned to the single parton created in the primary process. Although many parton properties are relatively easily reconstructed this way, an information about the parton flavour remain unknown. A lot of effort is invested in the parton flavour tagging. The first introduced techniques involved tagging of the jets initiated by the heavier quarks, such as charm and bottom quarks. Knowledge of the parton flavour can be very useful for the more precise measurements in Standard

Model a good parton flavour tagger can be crucial in obtaining it. The discrimination of the light quark initiated jet from the gluon jet background proved to be even harder task due to the similar emission patterns in the fragmentation process. There are some variables that show a particular discrimination power. However, if one wants perform even more precise measurements he must include some other techniques. Difficulty arises from the high-dimensionality of jets paired with non-perfect knowledge of the fragmentation and the hadronization process. One possible approach to solve this issues is to use modern machine learning techniques which showed very good performance when dealing with high-dimensional data in problems with the insufficient domain knowledge. Neural networks proved to be especially powerful in capturing non-linear correlations between the different features in data. Aim of this research is to develop a quark-gluon jet discriminator suitable for the applications in data collected by the ALICE detector at CERN. Also, we introduced a technique to recover emissions pattern in such data.

This thesis is organized as follows: This chapter describes the theoretical background of the Standard Model and Quantum Chromodoynamics and the theoretical overview of jets with jet finding algorithms and the most common variables used in the description of the jets. The experimental setup consisting of the LHC and the ALICE detector is described in Chapter 2. An overview of the machine learning techniques with the special attention to the neural networks is presented in the Chapter 3. Chapter 4 provides the methodology of the presented research which contains descriptions of the introduced algorithms and models. The results and evaluations of the developed models are presented in Chapter 5. Finally, the overview of introduced models and methods along with the corresponding results are discussed in the Chapter 6.

1.1 Standard Model

The Standard Model (SM) of particle physics is widely regarded as the best description of physics at subatomic level. It successfully describes strong and weak nuclear interactions as well as electromagnetism. The remaining fundamental force, not described in the SM, is gravity, which is the weakest of the interactions and it's effects can be well observed only on the scale of large masses and distances. Currently, gravity is best described by the theory of General Relativity (GR) which describes how matter affects the fabric of spacetime and vice versa. These two frameworks are proven to be

mutually incompatible at the moment, but a lot of research is aimed towards their unification which would provide the so called Theory of Everything (ToE). One example of such a theory would be the famous String theory, along with its variations. The Standard Model is based on the Quantum Field Theory (QFT) framework in which all of the elementary particles are described as excitations of quantum fields. An important feature of the SM is gauge symmetry. A consequence of this symmetry is the existence of the so called spin 1 gauge bosons, which are particles that mediate forces between elementary particles.

According to the SM, the matter content of the universe consists of 12 fundamental particles. They can be divided into two groups, quarks and leptons. Unlike the gauge bosons, quarks and leptons are fermions with a spin of exactly $\frac{1}{2}$ and they obey the Fermi-Dirac statistic. The six flavours of quarks are the down (d), up (u), strange (s), charm (c), bottom (b) and top (t). The six leptons are the electron, the electron neutrino (ν_e), the muon (μ), the muon neutrino (ν_μ), the tau lepton (τ) and the tau neutrino (ν_τ). Quarks interact with all three known fundamental forces, while leptons do not interact via the strong nuclear force. All of the mentioned particles have their antimatter counterparts which have the opposite electric charge. These twelve fermions can also be divided into three generations. Each generation consists of four particles, two quarks with electric charges of $+\frac{2}{3}$ and $-\frac{1}{3}$, a lepton with charge of -1 and a corresponding neutral neutrino. All the charges are expressed in units of the elementary charge e. The three generations of particles differ only in particle masses, while the other properties remain the same. The particles from a higher mass generation spontaneously decay into lower mass generation particles. As a consequence of this, the Universe mostly consists of the particles of the first generation, the up and down quarks, and the electron and the electron neutrino. Two u quarks and one d quark make up a proton and one u quark and two d quarks make up a neutron. It must be noted this is a very simplified view of the proton and the neutron. In a particular moment in time there can be a lot of quark-antiquark pairs inside the proton or neutron, as well as heavier quarks from a higher mass generation. The matter particles interact with each other by three fundamental interactions. Each of the three has its own force carrier particles.

The electromagnetic interaction is described in terms of Quantum Electrodynamics (QED). It is a quantum field theory with $U(1)$ gauge symmetry. This causes the existence of a massless neutral gauge boson called photon. Charged particles interact by exchanging virtual photons. Out of the fundamental matter particles, only the neutrinos don't interact via the electromagnetic interaction. Since the photon does not have

any verifiable mass, the electromagnetic interaction has an infinite range. Due to the small coupling constant which allows perturbative calculations, QED predicted a variety of precise results, like the g factor of an electron or the hyperfine structure of the electron levels in a hydrogen atom [1].

Both quarks and leptons also interact via the weak nuclear interaction. This interaction is responsible for radioactive decays. A particle can change its flavour via the weak interaction. For example, a muon can spontaneously decay into an electron, an electron antineutrino and a muon neutrino. The first theoretic approach to a description of the weak interaction was done by Enrico Fermi in 1933, where he suggested that the beta decay can be explained by a four-fermion interaction, known as Fermi's interaction [2]. After the discovery of parity violation and CP symmetry breaking, a more accurate theoretical description was needed [3]. This was done in the work of Sheldon Glashow, Abdus Salam and Steven Weinberg [4]–[6]. Their approach unifies the weak interaction and QED into a single Electroweak framework. The electroweak theory is the quantum field theory with a gauge symmetry group $SU(2) \times U(1)$. It gives rise to four massless gauge fields: three weak isospin fields, W_1, W_2 and W_3, and a weak hypercharge field B. However, these are not physical fields. With the concepts of spontaneous symmetry breaking and associated Higgs mechanism one can imply an existence of four physical gauge bosons: the photon, the $W^{\pm}$ and the Z bosons. The photon is, as mentioned, responsible for the long range electromagnetic interactions. Charged $W^{\pm}$ bosons have a mass of 80.4 Gev$/c^2$ and the neutral Z boson 91.2 GeV$/c^2$. They have been experimentally confirmed in 1983 by the UA1 and UA2 collaborations at European Center for Nuclear Research (CERN). Since these bosons are not massless, the weak interaction has a short range. A consequence of the Higgs mechanism is the existence of the scalar Higgs boson [7]. The interaction with the Higgs field gives masses to other fundamental particles. This final piece of the SM was experimentally confirmed in 2012 by the ATLAS and CMS collaborations at CERN [8].

The strong nuclear interaction describes the interactions between quarks. It is responsible fur nucleon stability (it holds the quarks inside the proton or neutron), and, as a residual effect it is also responsible for the interaction between the nucleons. The strong nuclear force is theoretically described by Quantum Chromodynamics (QCD), which is analyzed in detail in the following section.

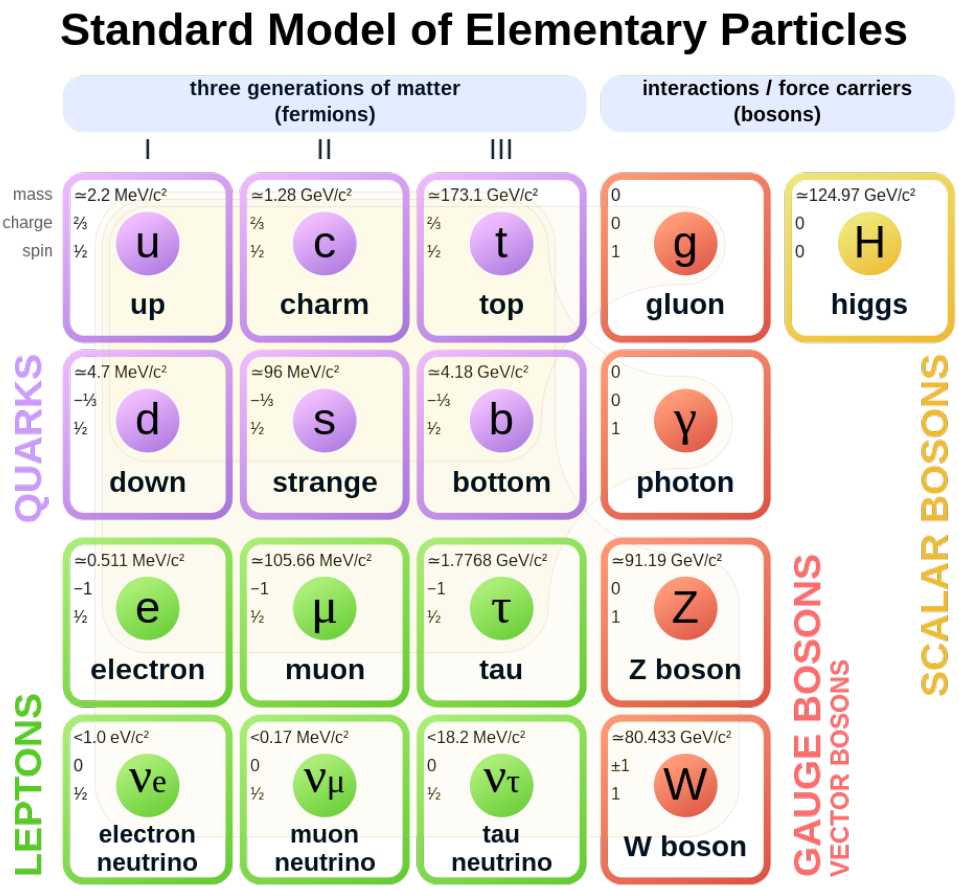


FIGURE 1.1: The particles of the Standard model [9].

1.2 Quantum Chromodynamics (QCD)

The Quantum Chromodynamics is a quantum field theory with $SU(3)$ gauge symmetry. As a consequence, quarks have additional three degrees of freedom called the color charge, which can be red, green or blue, hence the name Quantum Chromodynamics. There are 8 different generators of the group $SU(3)$. Therefore, there are eight gauge bosons which mediate the strong nuclear force. These bosons are called gluons. The Lagrangian density of the QCD is given by 1.1:

$$\mathcal{L} = -\frac{1}{4}F^{\alpha}_{\mu\nu}F^{\mu\nu}_{\alpha} + \sum_i \bar{\psi}_i\gamma^{\mu}(\partial_{\mu} - igA^{\alpha}_{\mu}t_{\alpha})\psi_i - \sum_i m_i\bar{\psi}_i\psi_i\,. \tag{1.1}$$

In the Lagrangian density the index i iterates on all quark flavours. ψ and A^{α}_{μ} denote quark and gluon fields, respectively. The index α iterates over 8 different gluons and t_{α}

are the 3×3 Gell-Mann matrices, which are the $SU(3)$ group generators. The indices μ, ν, α follow the Einstein summation convention. The first term in 1.1 describes the free gluon fields defined by eight four-potentials A^{α}_{μ} in the following way:

$$F^{\alpha}_{\mu\nu} = \partial_{\mu} A^{\alpha}_{\nu} - \partial_{\nu} A^{\alpha}_{\mu} + C^{\alpha}_{\beta\gamma} A^{\beta}_{\mu} A^{\gamma}_{\nu} \,, \tag{1.2}$$

where $C^{\alpha}_{\beta\gamma}$ are the SU(3) structure constants. In contrast to photons, the gauge bosons of QED, gluon field tensors depends on the gluon fields, and not only on its derivatives. As a result, the gluons interact with each other since the third term in 1.2 implies color charged gluons which interact mutually. There are four possible processes in the QCD. One is where a quark emits a gluon, one is where a gluon splits into a quark-antiquark pair and the remaining two are the 3-point gluon interaction and the 4-point gluon interaction. The first two processes are analogous to QED processes, unlike the latter two which are specific only to QCD. Another big difference between QCD and QED is the value of their coupling constants. In QCD, the value of this dimensionless parameter is much higher than in QED ($\alpha \approx 1/137$) and it forbids the application of perturbative calculations. There are several non-perturbative approaches to do calculations and predictions in the QCD, with the Lattice QCD being the best established [10]. This approach discretizes spacetime points in order to simplify the calculation of path integrals. It is a slow method and needs a lot of computational resources, but it has high applicability. A peculiarity of QCD is that the coupling constant g becomes lower at high energy scales. Then, one is allowed to use a perturbative approach.

1.2.1 Vacuum polarization

The first order perturbations of the QCD vacuum involve a process where a quark-antiquark loop is created from the gluon field. An analogue process appear in QED. This causes a stronger interaction on shorter distances. But, there is also the possibility that a pair of gluons is created from the gluon fields. The latter process causes anti-screening of the color charge. It makes the coupling stronger at longer distances. The combined effect of these causes the strong interaction to be stronger at large distances and low transferred momenta. However, the interaction is weaker in the opposite case. A first order of the perturbative QCD calculation of the strong coupling constant gives:

$$\alpha_s(Q^2) = \frac{12\pi}{(22 - n_f) \ln \frac{Q^2}{\Lambda^2_{QCD}}} \,, \tag{1.3}$$

where n_f denotes the number quark flavours, $\Lambda_{QCD} \approx 0.22$ GeV is parameter of the QCD and Q is the transferred four-momentum. This gives rise to the two important features of the QCD, asymptotic freedom and color confinement. The comparison of the QED and QCD running coupling constants with marked asymptotic freedom and color confinement are shown in Fig. 1.2.

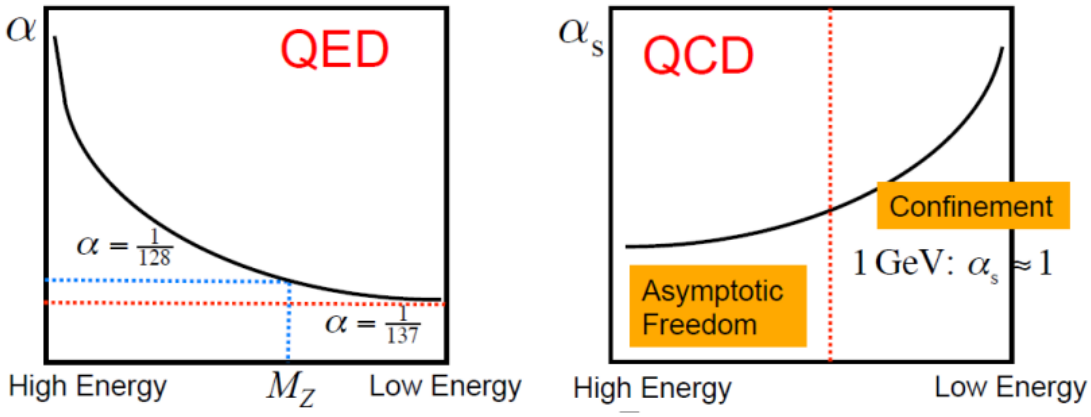


FIGURE 1.2: The comparison of the running coupling constants for QED and QCD [11].

1.2.2 Asymptotic freedom

Asymptotic freedom is the QCD property where it is implied that very close quarks does not „see" each other [12]. According to 1.3, quarks and gluons are almost free of each other when the energy density is high enough. This allows for the usage of perturbative calculations. Because of asymptotic freedom, the scientific community proposed a new phase of matter, the quark gluon plasma (QGP), which is believed to have been the dominant phase in the early universe, before cooling down. The QGP is considered to be a soup of deconfined quarks and gluons. The existence of QGP was first experimentally confirmed by the RHIC experiment in high energy heavy ion collisions [13]. In such experiments the conditions of the early universe are simulated.

1.2.3 Color confinement

In contrast to the high energy densities and closer distances, at lower energy densities and larger distances a strong nuclear interaction grows stronger. As a quark distances itself from the other quark, a gluon field between them can contain sufficient energy to

create a new quark-antiquark pair. These processes continue until all the quarks group itself into color-singlets, a composition of particles with zero total color charge. This process is called hadronization and the color-singlets are called hadrons. An isolated single quark cannot be detected. Only hadrons are detected. In most cases, hadrons are made up of 2 or 3 quarks and they are accordingly divided into two groups, mesons and baryons, respectively. Pions and kaons are examples of mesons, while nucleons are examples of baryons. In heavy ion collisions, as the created QGP cools down, a phase transition occurs and results in a hadronic gas. Very energetic quarks and gluons created in hard processes with high transferred momenta manifest themselves as collimated showers of hadrons. These objects are called jets and they are studied to infer knowledge about initial quark or gluon and the hard process they participated in. The jets are analyzed more thoroughly in the next section since their are the main topic of a research in this thesis.

1.3 QCD Jets

A QCD jet is a collimated stream of particles. It can be used as an experimental probe with which one can deduce some fundamental properties of the QCD by measuring its total energy, momentum, mass and other observables. Quarks and gluons, created in a hard process, move as quasi-free particles at short distances (of the order of 10^{-2} fm) due to the asymptotic freedom. When these partons separate themselves to the distance of the order of 1 fm, they radiate gluons preferentially with small angles relative to their direction. Subsequently, the gluons split into two gluons or quark-antiquark pairs. This can be repeated numerous times so that the original parton cascades into a shower within a narrow cone. This process is called jet fragmentation [14]. Afterwards, when quarks and gluons become separated by larger distances, hadrons are non-perturbatively created via the process of hadronization which preserves the kinematic properties of the original parton.

The description of the initial stages of parton fragmentation is possible in a perturbative way. Let us consider the cross-section for a quark emitting a gluon calculated to first order:

$$\frac{\mathrm{d}^2\sigma}{\mathrm{d}z\mathrm{d}\sin\theta} = C_F \frac{\alpha_S}{2\pi} \frac{2}{\sin^2\theta} \frac{1+(1-z)^2}{z}, \tag{1.4}$$

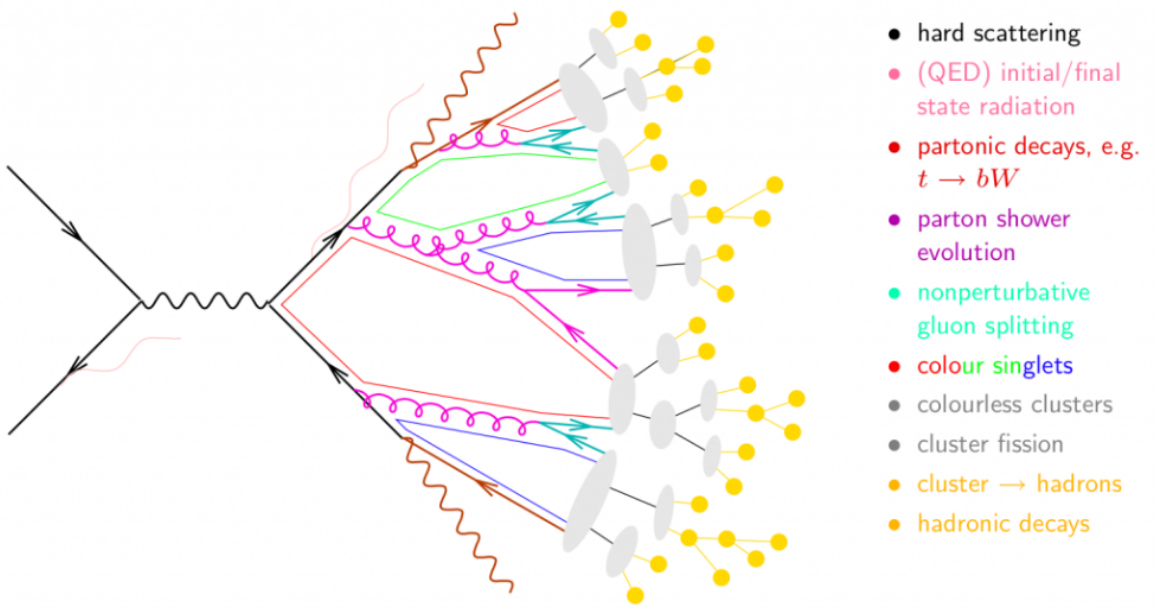


FIGURE 1.3: Visualization of the QCD jet with the fragmentation and hadronization processes [16].

where θ is the angle between the initial quark and the emitted gluon, z is the energy fraction of the gluon with respect to the quark and $C_F = 4/3$ is a color factor which quantifies the coupling between the quark and the gluon. One can notice that the cross-section diverges for „soft" z ($z \approx 0$) and in the collinear limit ($\theta \approx 0, \pi$). These two divergences, taken together, are called the Infrared divergence. Preferably, in an experimental study of jets one wants to use observables which are infrared safe, i.e. observables which are not sensitive to effects in the collinear and soft limits. The splitting functions $P_{ij}(z)$, also called Altarelli-Parisi functions [15], are interpreted as the probability that a parton of type j emits a parton of type i, with a fraction z of the energy of the parent parton. The spin-avaraged splitting functions in the collinear limit for the QCD processes are:

$$P_{qg}(z) = C_F \frac{1 + (1-z)^2}{z} \qquad P_{qq}(z) = C_F \frac{1 + z^2}{1 - z} \tag{1.5}$$

$$P_{gg}(z) = 2C_A \frac{(1 - z(1-z))^2}{z(1-z)} \qquad P_{gq}(z) = \frac{1}{2}(z^2(1-z)^2) \tag{1.6}$$

1.3.1 Jet algorithms

In high-energy collisions, the final states consist of many particles. To identify QCD jets, the final state particles must be grouped in some way. Because of the collimated nature of the jets, qualitatively they are easy to classify. However, if highly precise measurements are to be made, algorithms that define the jets in a rigorous manner must be used. Also, when comparing experimental measurements with some theoretical predictions, the algorithms used in both analyses must be mutually compatible. There are several algorithms used to perform this task that, and can be classified into two categories, recombination and cone algorithms. Important requirements that jet algorithms must fulfill are infrared and collinear safety. In other words, collinear splittings and soft emissions should not alter the clustered jets. Also, jet finding algorithms should have minimal sensitivity to hadronization, since the process belongs to a non-perturbative regime of the QCD and is not modelled properly.

In the past, cone algorithms have been widely used in hadron colliders due to their fast computational performance. However, they are usually not infrared and collinearly safe and are generally not favored by theorists. With the development and improvement of the speed of sequential recombination algorithms, cone algorithms are used less and less in modern analyses. Examples of the cone algorithms used at the LHC are the Iterative Cone Algorithm and SISCone [17].

Sequential recombination algorithms attempt to mimic the QCD fragmentation process. Particles from the final state recombine iteratively. First, a pair of particles with a minimum distance between them is searched for. The distance metric is defined as follows:

$$d_{ij} = \min(k_{Ti}^{2p}, k_{Tj}^{2p}) \frac{(y_i - y_j)^2 + (\phi_i - \phi_j)^2}{R^2}, \tag{1.7}$$

with k_T denoting the transverse momenta of the particles with regard to the beam axis. (y, ϕ) denote the particle's rapidity and the azimuthal angle. The parameter R scales the calculated distance and is typically chosen to have value of 0.4. The parameter p is typically chosen to be $1, 0, -1$ and implies an algorithm which is used. These algorithms are called the k_T, Cambridge-Aachen (C/A), and the anti-k_T, respectively. After the pair with the minimum distance d_{ij} is found, the two particles are combined so that their four-momenta are summed into so-called pseudojets. The same procedure is repeated for the remaining particles and pseudojets. The algorithm stops when each pair of particles has a distance larger than the minimal distance to the beam defined

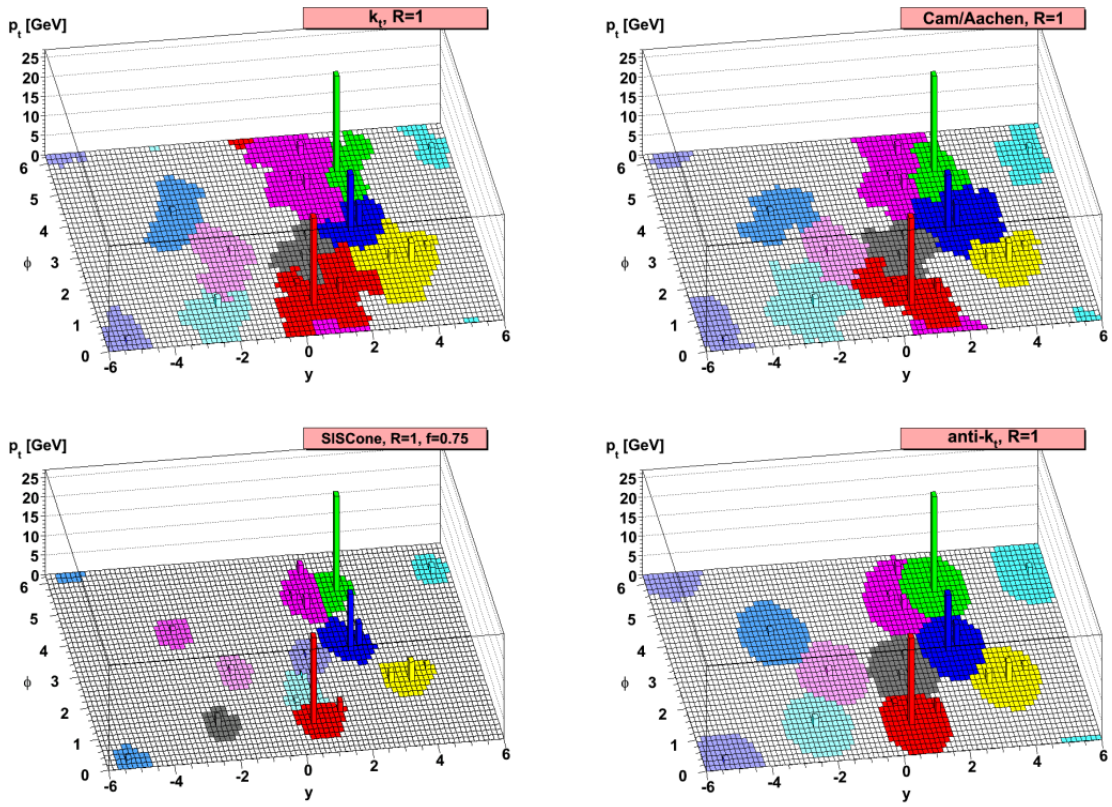


FIGURE 1.4: Event clusterings of different jet finding algotithms [19].

as $d_{iB} = k_{Ti}^{2p}$. When this condition occurs, the procedure is stopped and recombined pseudojet is called a jet.

The k_T algorithm prefers to cluster low momenta particles first according to 1.7. As a result, the area of the reconstructed jet is prone to the significant fluctuations and the algorithm is susceptible to the Pile-Up (PU) and Underlying Event (UE) effects [18]. In the C/A algorithm, the distance variables are independent of momentum and so its area fluctuates somewhat and is moderately susceptible to the UE and PU effects. This algorithm is proven to be best in the resolving of the jet substructure. The anti-k_T clusters high energy particles first and is only slightly sensitive to the UE and PU effects. The reconstructed jets have an almost perfect conical shape and their areas experience only small fluctuations. However, this algorithm is worst for studying the jet substructure.

1.3.2 Jet substructure

Jet finding algorithms aim to identify a jet which will represent an initial quark or gluon participating in the primary hard process of the collision. However, an analysis of the jet substructure also proves to be useful in studying QCD. The most common task of such an analysis is to distinguish different kinds of jets. For example, one wants to separate W/Z jets from much more common quark or gluon initiated jets. Jet substructure tools are designed to study the internal kinematic characteristics of the jets. These methods can be grouped in three broad categories, prong finders, radiation constraints and groomers.

Prong finders are designed to identify hard emissions inside a jet. Typically, quark and gluons emit soft partons and are considered to be a 1-prong object. On the other side, boosted massive particles such as $W^{\pm}$, Z and Higgs bosons decay into two high-momenta partons. Then these objects are called 2-prong. Boosted top jets would be 3-prong objects. Prong finder tools are used to distinguish between these different kinds of jets.

Groomers are generally used when studying large-radius jets. Because of their large area they are very sensitive to PU and UE effects. These tools are designed to minimize the impact of these effects. They usually work by removing soft particles far from the jet axis where these effects will likely show up, hence the name grooming. Groomers share similarities with prong finders since removing soft background reveals hard prongs. Examples of the prong finding and grooming tools are Mass-drop tagger, SoftDrop, I and Y-pruning and others [20], [21].

The other main difference between jets is in their soft-gluon radiation patterns. For example, jets initiated by quarks are expected to have less soft gluons than jets initiated by gluons. In the radiation constraint approach one typically wants to study jet shapes. In most cases, a jet shape observable is constructed and is then used to impose a cut. There is a variety of introduced jet shape observables. The simplest group of such observables are the so-called generalized angularities [22], which are defined as:

$$\lambda_{\beta}^{\kappa} = \sum_{i \in jet} z_i^{\kappa} \theta_i^{\beta}, \tag{1.8}$$

where z_i is the fraction of the jet transverse momentum carried by the particle i and θ_i roughly represent the angle of the i-th particle relative to the jet axis:

$$z_i = \frac{p_{T,i}}{\sum_j p_{T,j}} \qquad \theta_i = \frac{(y_i - y_{jet})^2 + (\phi_i - \phi_{jet})^2}{R^2}\,. \tag{1.9}$$

The angularities λ_0^0 and λ_0^1 are simply the total number of constituents (jet multiplicity) and the jet total transverse momentum, respectively. Generalized angularities are not infrared safe except in a special case when $\kappa = 1$. The special case $(\kappa, \beta) = (1, 1)$ is a known jet shape observable called the jet width or girth. The observable λ_1^2 in the limit of massless particles reduces to the jet mass. The more radiation there is present in a jet, the higher the values of angularities. Therefore, it is a measure of QCD radiation relative to the jet axis. Typically, gluon initiated jets have higher angularity values than quark initiated jets and can be used to discriminate between the two.

Finally, jet shape observables called N-subjettines are widely used in jet substructure analyses [23]. Firstly, for a given jet, one needs to specify a set of N axes (subjets). Then the N-subjettines are introduced as follows:

$$\tau_N = \frac{1}{d_0} \sum_{i \in jet} p_{T,i} \min(\Delta R_{ij})\,, \tag{1.10}$$

where the index i runs over all jet constituents and the index j runs over the set of specified axes. The expression $\Delta R_{ij} = \sqrt{(y_i - y_j)^2 + (\phi_i - \phi_j)^2}$ represents the distance between the i-th jet constituent and j-th axis. There are several ways to define a needed set of axes. One way is to use the k_T algorithm to re-cluster jet constituents into N distinct subjets. Also, one can choose axes which minimize the value of τ_N. Often, the minimisation process is done iteratively. Similarly to the angularities, τ_N measures the amount of the radiation relative to the specified axes. For a jet with N prongs, a higher value τ_i is expected for the $i < N$ and a lower value for the $i \geq N$. A ratio of the consecutive subjettines $\tau_{N,N-1} = \tau_N / \tau_{N-1}$ can be useful in a discrimination of the N-prong jets against the QCD background (quark and gluon initiated jets). For example, one can use a cut on $\tau_{2,1}$ to identify $W/Z/H$ jets from the QCD background. Also, τ_N exhibits a slightly higher values for a gluon initiated jet in contrast to a quark initiated jet.

1.3.3 The Lund jet plane

The Lund jet plane is introduced as a useful graphical representation of emissions in a jet [25]. The variables $(\ln 1/\theta, \ln z\theta)$ are in the most cases the axes of the Lund diagram,

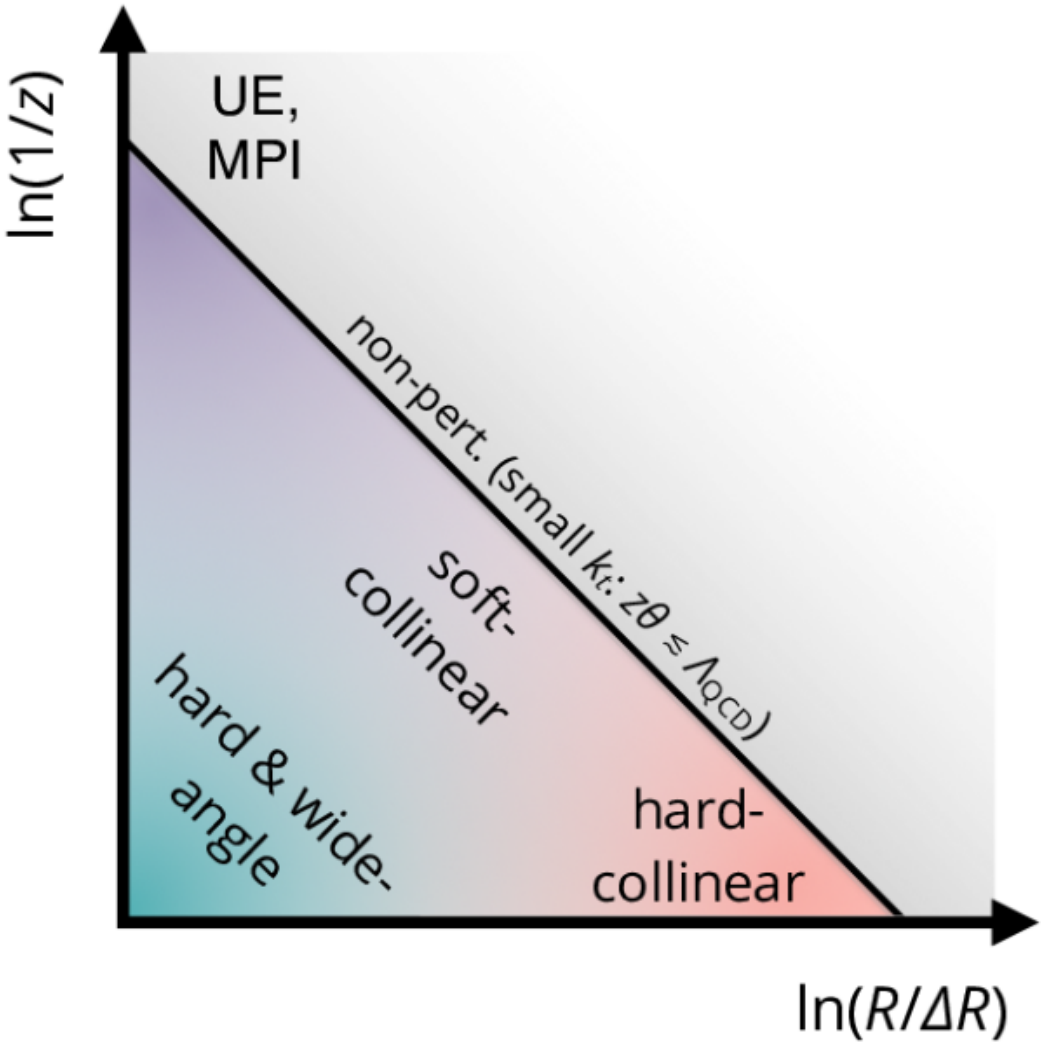


FIGURE 1.5: The Lund jet plane with the distinct regimes [24].

where z denotes the transverse momentum fraction and θ denotes the angle of emission as defined in 1.9. This way, different kinematic regimes are clearly separated as seen in 1.5. Also, emissions in soft and collinear limits are distributed uniformly in the Lund plane according to 1.4. Mass, angle and momentum can easily be determined from a Lund plane. These considerations can be expanded to the jet as a whole. In such an analysis one usually declusters the full jet into two subjets using the C/A algorithm. Knowing the four momenta of the full jet and subjets, $p = p_i + p_j$, variables z and θ can be constructed. This procedure is then repeated for the harder jet p_i. Such a procedure provides a list of pairs $(z_i\theta_i, 1/\theta_i)$, where index i iterates over all primary emissions in a jet. In a way, one can access to the internal structure of a jet. Using the given pairs,

an average Lund plane density can be obtained:

$$\rho = \frac{1}{N_{jet}} \frac{\mathrm{d}^2 N_{emissions}}{\mathrm{d}(\ln z\theta)\mathrm{d}(\ln 1/\theta)} \,. \tag{1.11}$$

This distribution of emissions has been studied analytically in [26]. Experimental values can be seen in measurements done by the ATLAS Collaboration [24]. The Lund jet plane also can be useful for constraining Monte-Carlo generators and studying how the medium in heavy ion collisions impacts the jet. Also, jet discriminators and taggers can be built with the help of the Lund jet plane using log-likelihood approach or a machine learning technique of some kind.

1.3.4 Quark and gluon initiated jets

Hard scattering processes emit partons which produce hadronic jets, as explained. These jets can be found by using the tracking and calorimeter systems. From experimental observations and theoretical insights, these reconstructed jets can show somewhat different properties depending on the flavour of the initial parton. Generally, if a gluon initiated a jet, it will have higher particle multiplicity and softer fragmentation functions. Also, gluon jets tend to be wider than quark jets. One of the reasons for this is the higher gluon color factor. These differences can be used to construct jet taggers which can be used in physics analyses where one wants to eliminate the QCD background, in which the gluon component is predominant.

The ATLAS Collaboration used a method based on data-driven extraction of quark and gluon jet properties. They also tested a number of discriminant variables [27]. The ATLAS jet multiplicity measurements are shown in 1.6.

Quark/gluon tagging in the CMS Collaboration used observables based on performances in the Monte Carlo simulated QCD events. They constructed a likelihood-based discriminator and validated it using data. They also employed a data-driven method to do corrections for observed differences with the data. The use of different parton shower models and their effect on the performance of the classifier is analyzed in [29].

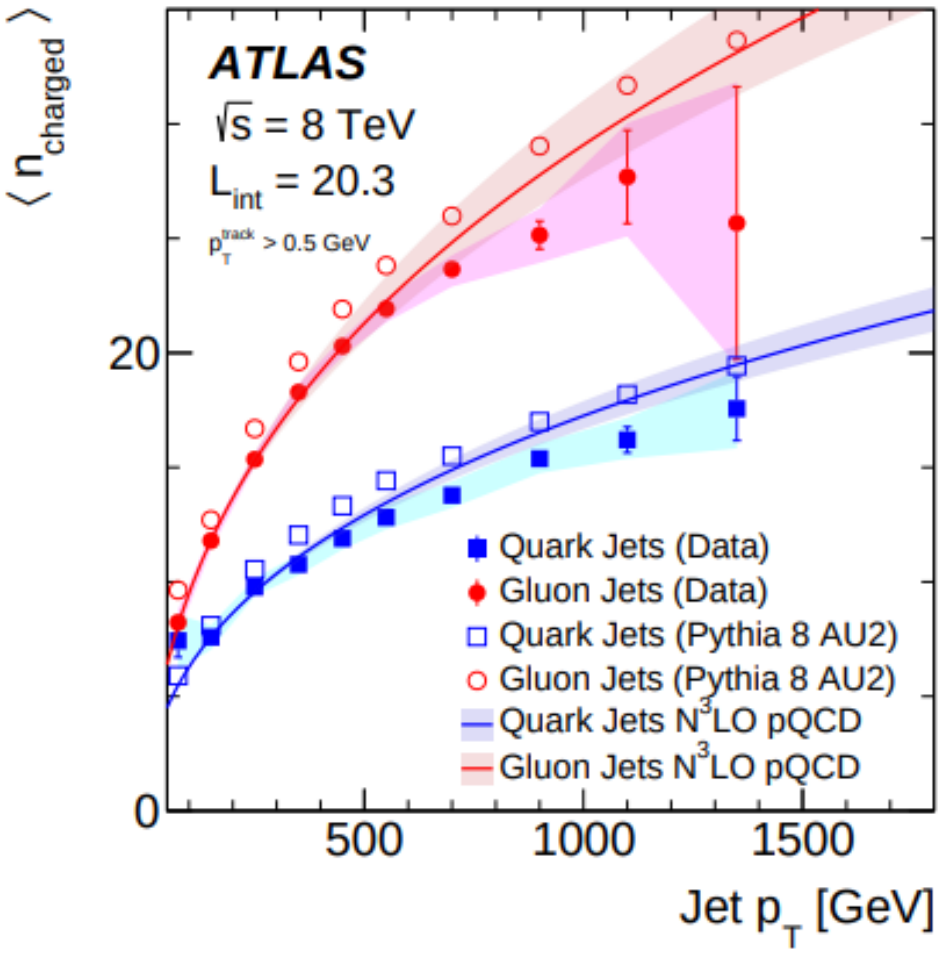


FIGURE 1.6: p_T- dependence of charged-particle multiplicity for quark- and gluon-initiated jets, as measured by the ATLAS collaboration [28].

Chapter 2

Experimental Setup

This chapter describes the LHC (Large Hadron Collider) and ALICE (A Large Ion Collider Experiment). The LHC is the particle accelerator which houses ALICE, the detector used to collect data used to develop the methods presented in this thesis.

2.1 The Large Hadron Collider

The LHC is currently the world's largest particle accelerator. It was built by the European Organization for Nuclear Research (CERN) with an intent to test various predictions in particle physics, primarily the existence of the Higgs boson. It is the latest addition to the CERN accelerator complex. The LHC first started in September 10, 2008, but the run soon terminated due to a malfunction in some of its magnets. After a period of repairs, it produces high energy particle collisions since march 2010 without any major issues. One of the most significant breakthroughs came in 2012, when the major experiments at the LHC confirmed the existence of the Higgs boson with a mass of approximately 125 GeV/c^2. The LHC is located in the tunnel with a circumference of 27 km which lies up to 175 m underground beneath the France-Switzerland border near Geneva.

At the LHC, large superconducting magnets are used to contain particles in their circular trajectories. The particle acceleration takes place at a number of radiofrequency cavities located along the LHC ring. Particle beams travelling in opposite directions intersect at four points. These interaction points are occupied by large detectors which are run by the four large collaborations of scientists. The largest ones are general-purpose detectors, ATLAS (A Toroidal Lhc Apparatus) and CMS (Compact Muon Solenoid). They investigate a wide range of physics, from the search for the Higgs boson to extra dimensions and particles that could make up dark matter. The

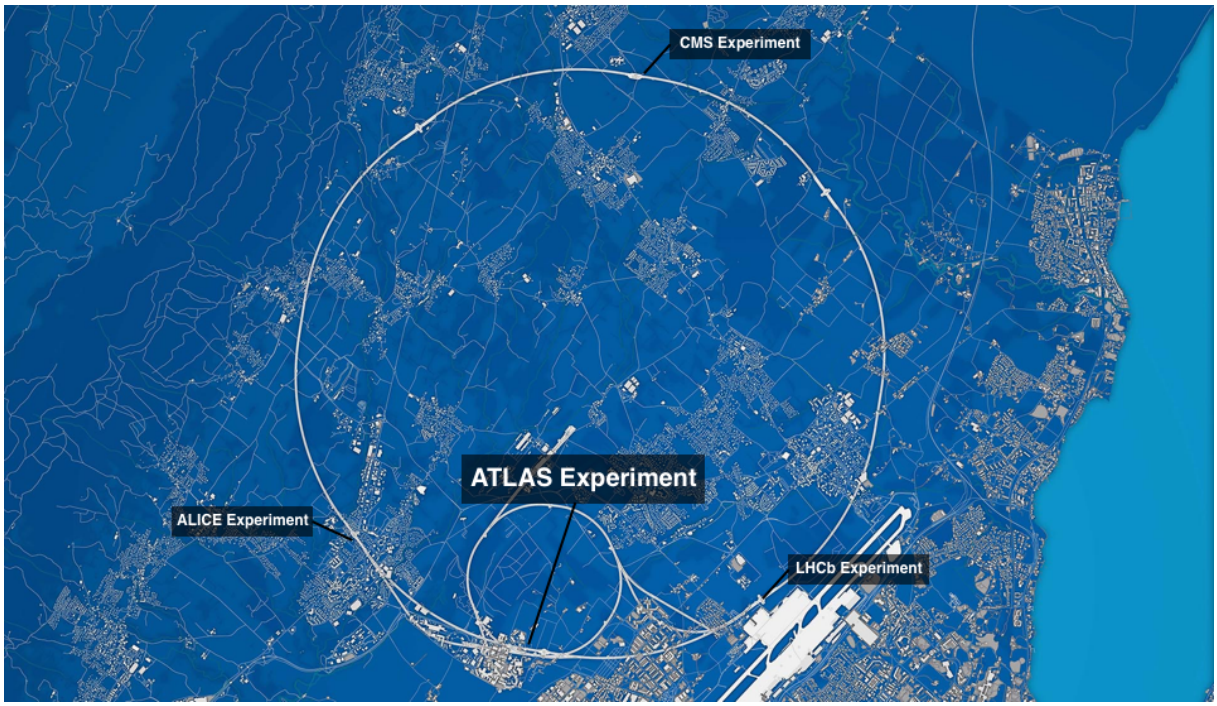


FIGURE 2.1: Location of the major LHC experiments [30].

LHC includes more specialized experiments such as ALICE, an experiment dedicated to heavy-ion physics, and LHCb (Large Hadron Collider), which specializes in investigating the slight differences between matter and antimatter by studying „beauty" quarks. Also, three smaller experiments are located at the LHC. TOTEM (TOTal Elastic and diffractive cross section Measurement) shares an interaction point with CMS and measures total cross sections, elastic scatterings and diffractive processes. MoEDAL (Monopole and Exotics Detector at the LHC) shares an interaction point with the LHCb and is involved in search for the magnetic monopoles or other highly ionizing particles. Finally, LHCf (LHC forward) shares an interaction point with ATLAS and studies particles thrown forward in collisions in order to simulate cosmic rays in laboratory conditions.

2.1.1 The CERN accelerator complex

The CERN accelerator complex is a collection of successive particle accelerators. Each machine boosts the energy of particles before injecting it into the next accelerator in the sequence. In the LHC, which is the last part of the complex chain, particles are accelerated up to the energy of 6.5 TeV per beam. The complex consists of six accelerators and

one decelerator. Since 2020, Linear accelerator 4 (Linac 4) is the source of proton beams by accelerating hydrogen ions, H^- to the energy of 160 MeV. The ions are stripped of their two electrons during injection from Linac4 into the Proton Synchrotron Booster (PSB) where they are accelerated to the energy of 2 GeV. Next elements in the chain of the complex are the Proton Synchrotron (PS) and Super Proton Synchrotron (SPS) which accelerate the beam of protons to energies of 26 Gev and 450 GeV, respectively. Then, the protons are finally injected to the two beam pipes of the LHC. Under normal operating conditions the beams circulate for few hours inside the LHC beam pipes after which the cycle is repeated. As already mentioned, the two beams are brought into collision inside four detectors, ALICE, ATLAS, CMS and LHCb, where the total energy per nucleon at the collision point is equal to 13 TeV. Although they are the most common particles to collide in the LHC, the protons are not the only particles accelerated. Lead ions used in the LHC start from a source of vaporised lead and are injected into the Linear Accelerator 3 (Linac3) before being collected and accelerated in the Low Energy Ion Ring (LEIR). After the transfer into PS they follow the same route to maximum energy of 2.76 TeV per nucleon.

2.1.2 The LHC superconducting magnets

A strong magnetic field is needed to keep the high energy particles in their circular path. For this purpose, a total of 1 232 of superconducting dipole magnets are installed at the LHC. Each of these magnets is 15 meters long and weighs 35 tonnes. They generate powerful 8.3 T magnetic fields. These magnets consists of two coils of copper-clad niobium-titanium wire cooled using superfluid helium-4 to their operating temperature of 1.9 K. Particles are accelerated in bunches which have a tendency to spread out since the they consist of positively charged entities. The imperfections in the magnetic fields on their edges also cause further spreading of the bunches. Several other factors also contribute to this drifting apart, such as initial conditions at the injection of the particles into the beam pipes. Spreading out in direction along the beam is annulled by the radiofrequency cavities. In the transverse direction the beam is focused by a total of 858 quadropole magnets.

Since quadrupoles focus the beam in one plane and defocus in the other, the quadrupole magnets are arranged in a FODO pattern to achieve focusing in both transverse planes. A FODO cell is a combination of focusing (F) and defocusing (D) quadrupoles separated by some drift space (O). The main dipole magnets and other multipole magnets,

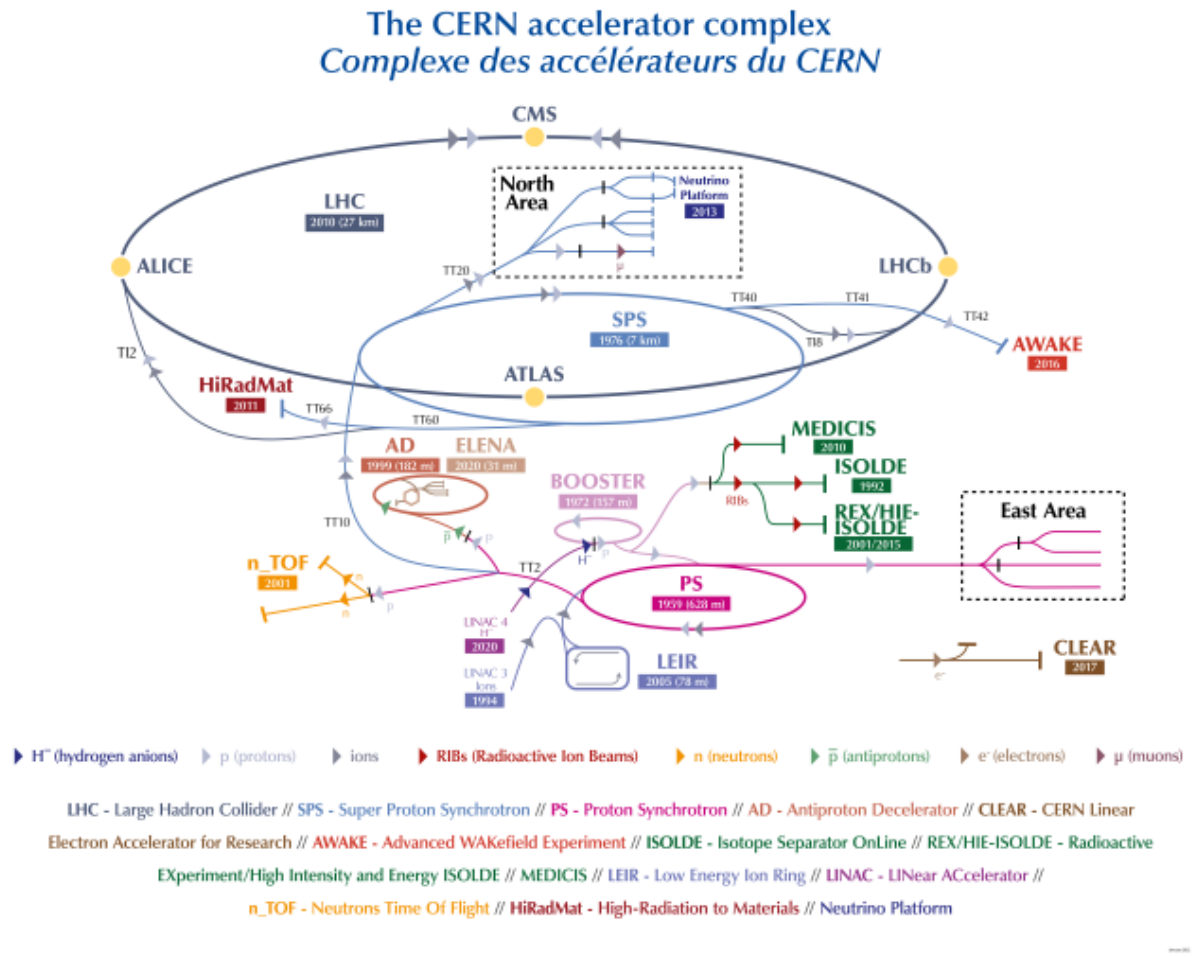
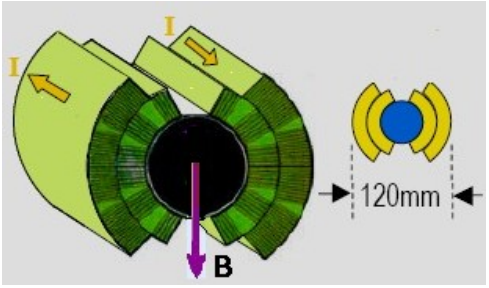
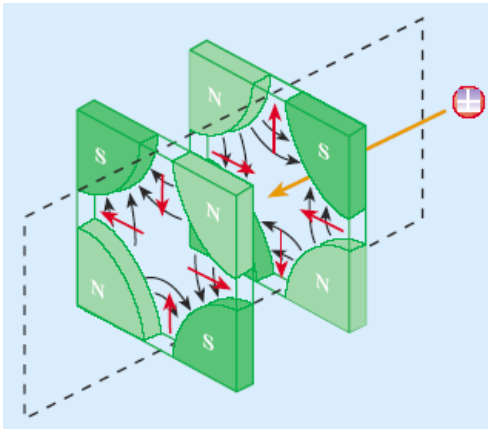


FIGURE 2.2: A schematic of the CERN's accelerator complex [31]



(A) Dipole magnets



(B) Quadrupole magnets

FIGURE 2.3: A schematic of the LHC's dipole and quadrupole magnets.

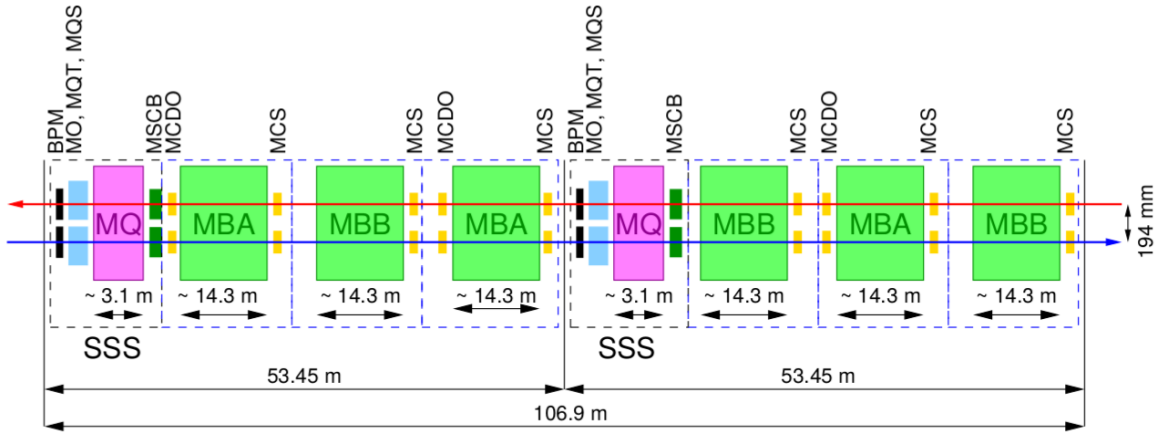


FIGURE 2.4: A standard LHC arc FODO cell [32].

used to correct imperfections in magnetic fields, are usually installed into the drift spaces in order to optimise the space on the accelerator's circumference. On the ends of each of the quadrupoles and dipoles are smaller corrector magnets of higher order. The basic magnetic cell (FODO) at the LHC is approximately 110 m long and consists of two perpendicular quadrupoles, six dipoles and a number of multipolar correction magnets [32]. There are approximately 10 000 superconducting magnets installed at the LHC in total which require almost 100 tonnes of superfluid ^{4}He.

2.1.3 Radiofrequency cavities

A radiofrequency (RF) cavity is a metallic chamber that contains an electromagnetic field. Its primary purpose is to accelerate charged particles. An electromagnetic field is supplied by klystrons and the energy is transferred by a waveguide. Charged particles passing through the cavity interact with the resulting electromagnetic field and are accelerated. The shape and size of the cavities is such that their resonant frequency corresponds to that of the generated periodic of the electric field. In the LHC, each RF cavity is tuned to oscillate at 400 MHz. The ideally timed proton, with the right amount of energy, will experience zero accelerating voltage when the LHC is at full energy. Protons with slightly different energies arriving earlier or later will be accelerated or decelerated so that they stay close to the energy of the ideal particle. This way, the beam is divided into discrete bunches of particles. At the LHC, there are two sets of four RF cavities per beam. A maximum electrical potential difference is set to 2 MV per cavity or a total of 16 MV per beam. At 7 TeV each bunch has an averege length of 7.6 cm

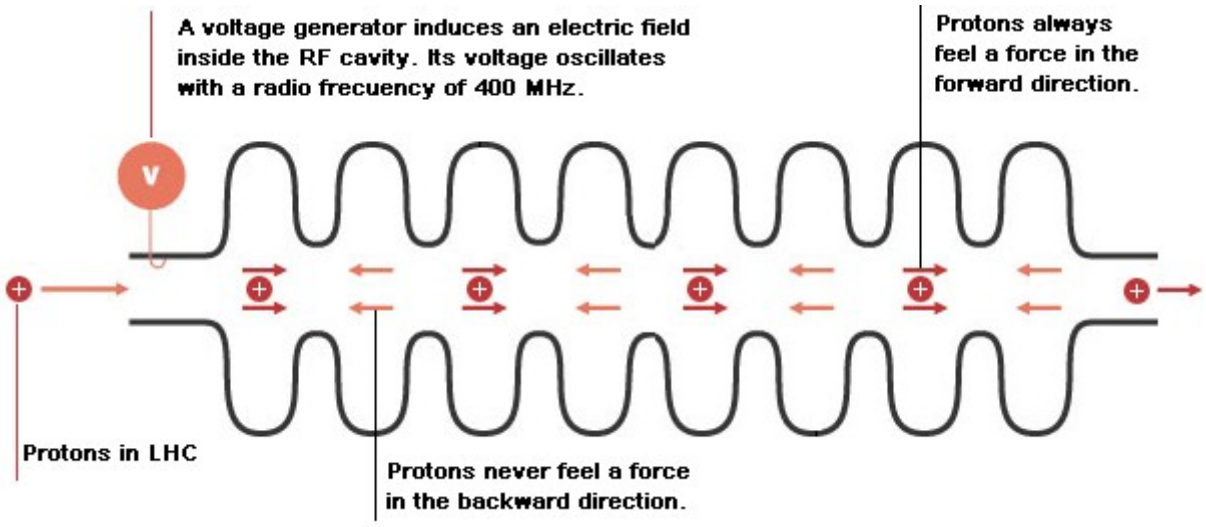


FIGURE 2.5: The principle of work of an RF cavity at the LHC.

and an average spread in energy of 0.011 %. The cavities are housed in cylindrical cryomodules which keep them working in a superconducting state, without losing energy to electrical resistance.

2.2 The ALICE detector

The ALICE detector is specialized for research of heavy-ion physics at the LHC. It is specifically designed to study the physics of strongly interacting matter at extreme energy densities and temperatures. Collisions in the LHC can generate temperatures more than 100 000 times hotter than the centre of the Sun. When the LHC provides collisions between lead ions, laboratory conditions similar to those just after the Big Bang can be recreated. Under such extreme conditions, protons and neutrons disintegrate into a soup of quarks and gluons. This peculiar phase of matter is called the quark-gluon plasma (QGP). Studying such a phase and its properties can prove to be very important for the understanding of the theory of quantum chromodynamics (QCD). The ALICE collaboration studies the QGP as it expands and cools down, observing its evolution and its phase transition to the hadronic phase of matter, the phase of matter the every-day Universe is mostly made of.

The ALICE collaboration uses the 10 000-tonne ALICE detector, which is 26 m long, 16 m high, and 16 m wide, to study QGP. The detector design enables efficient tracking of the huge number of particles which can occur in heavy-ion collisions. Also, it has

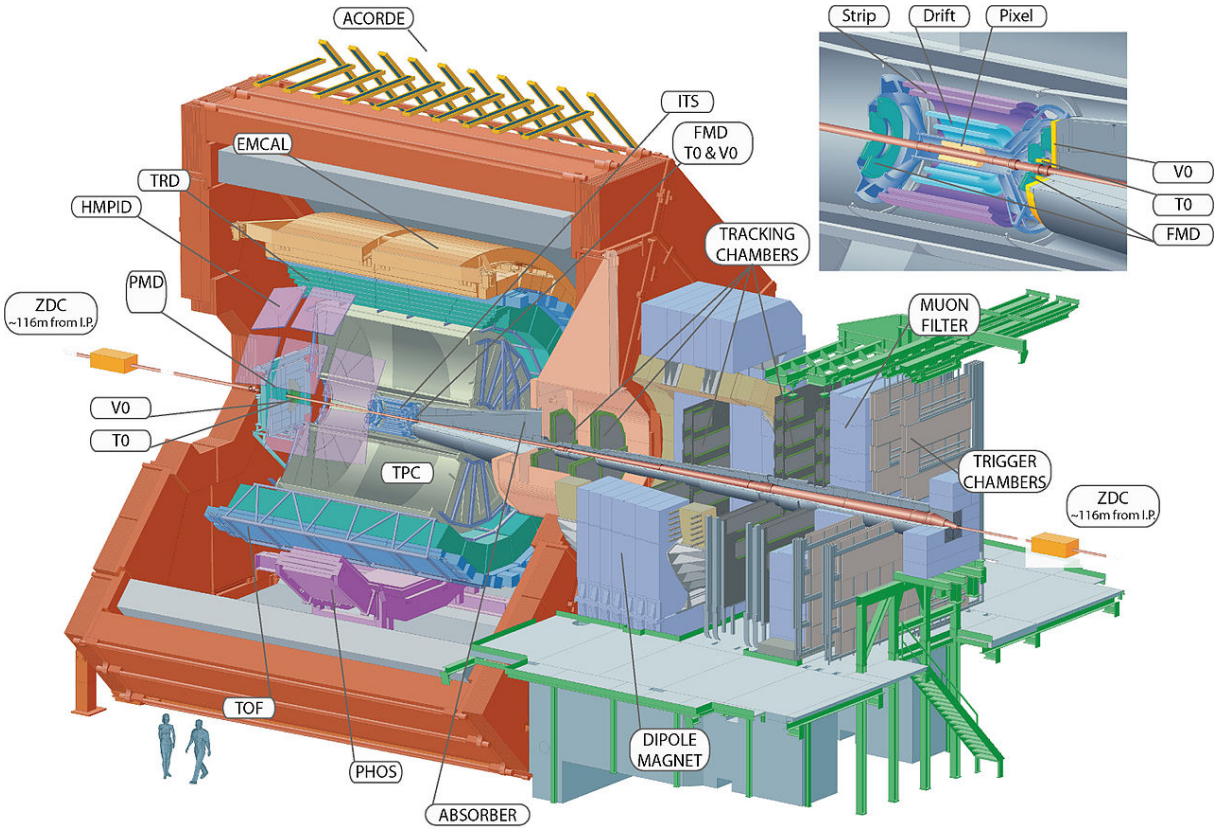


FIGURE 2.6: Layout of the ALICE detector.

very powerful particle identification capabilities. The detector is located in a cavern 56 m underground, close to the village of St.Genis-Pouilly.

As of 2022, the collaboration involves almost 2 000 scientists from 174 physics institutes in 40 countries.

The detector itself consists of two main parts: the Central barrel and the forward muon spectrometer (muon arm). The Central barrel is centred around the interaction point and is composed of detectors designed for the study of hadrons, electrons and photons. The central barrel covers the full azimuthal range and a pseudorapidity range of $|\eta| < 0.9$. The muon arm is located off to the side and is dedicated to the study of quarkonia behaviour in dense matter. It covers the pseudorapidity range of $-4.0 \leq \eta \leq 2.5$. Just as the central barrel, it also covers the full azimuthal range. The detectors of the central barrel are embedded in an L3 solenoidal magnet which generates a magnetic field 0.5 T in order to bend the tracks of charged particles for their momentum and charge determination. The ALICE detector is composed of a number

of smaller sub-detectors which are designed to do a specific task. Together they produce a precise and reliable measurement. The main tasks of the ALICE detector are particle tracking and particle identification, among others. A brief description of the most important sub-systems of the ALICE detector is given in the following sections.

2.2.1 The tracking system

A composition of cylindrical detectors located around the nominal interaction point is used to track the particles that are generated in the collision. The Inner Tracking System (ITS), the Time Projection Chamber (TPC) and the Transition Radiation Detector (TRD) detect the positions of charged particles along a number of points from which the particle trajectory can be reconstructed.

The Inner Tracking System (ITS)

The ITS is the innermost and the closest detector to the interaction point. It consists of 6 layers of semiconductor silicon detectors. Its primary task is precise tracking of particles and determining the positions of the primary vertex of the collision and the secondary vertices in the case of short-lived heavy particles. It consists, from inside out, of two layers of Silicon Pixel Detector (SPD), two layers of Silicon Drift Detector (SDD) and two layers of Silicon Strip Detector (SSD). Charged particles passing through each layer leave a signal which is used to reconstruct the particle tracks. Also, the ITS is used for the improvement of the momentum and angle measurements of other detectors. Additionally, it can be used for the identification of low-momentum particles.

The Time Projection Chamber (TPC)

The TPC is the principal component of the ALICE detector and its main tracking device. It is a cylindrical drift chamber with a length of 5 m, an inner diameter of 85 cm and an outer diameter of 250 cm. The total working volume of the TPC is about 90 m^3. This volume is filled with a Ne-CO_2-N_2 (90:10:5) gas-mixture. The chamber is divided into two parts by the central electrode which is located in the middle of the detector. Read-out planes are located at the both ends of the volume. A voltage difference between the central electrode and the read-out planes is equal to 100 kV and generates an electric field inside the volume. Charged particles travelling through the chamber

ionize the gas along their trajectory. Then, electrons drift towards the read-out plane which is divided into more than 570 000 pads. Each pad detects a drifting electron signal amplified by multi-wire proportional chambers located in front of the readout planes. This way, the radial component of the particle position can be measured. The longitudinal component can be determined by measuring the time needed by a drifting electron to reach the readout plane. The track position resolution of the TPC is about 1 mm in both the radial and the longitudinal directions. In the normal operating conditions the TPC can measure particles with momenta in the range from 200 Mev/c to 100 GeV/c. Although particle tracking is its primary purpose, the TPC can be very useful in particle identification by measuring the energy loss per unit of path length ($\mathrm{d}E/\mathrm{d}x$). Using this value and comparing with the well known Bethe-Bloch formula can be indicative of the type of the measured particle.

The Transition Radiation Detector (TRD)

Electrons and positrons can be distinguished from other charged particles by looking at the transition radiation (TR), which are X-rays emitted when a particles cross a boundaries of materials with differing dielectric constants. To amplify the produced transition radiation, the TRD uses a radiator containing a foam-like substance which causes the particles to pass over many such boundaries, thereby increasing the probability of TR emission. The TRD consists of 540 chambers arranged in 6 layers at a radial distance from 2.90 m to 3.68 m from the beam axis surrounding the TPC. In each chamber, the radiator is followed by a Multi Wire Proportional Chamber filled with Xe-CO_2 gas-mixture which precedes a drift region of 3 cm. The extracted temporal information represents the depth in the drift volume at which the ionisation signal was produced. Each chamber has over 2 000 readout pads. Although the main purpose of the TRD is to provide electron and positron identification and fast triggering of the charged particles, the TRD also contributes significantly to the track reconstruction and calibration.

2.2.2 Particle Identification

One of the best features of the ALICE detector is its capability to identify particles. Although the ITS, the TPC and the TRD give information about particle identity, more specialized detectors are needed for more reliable measurements. Such detectors are

the Time-Of-Flight (TOF) and the High Momentum Particle Identification Detector (HMPID).

The Time-Of-Flight detector (TOF)

The TOF of the ALICE detector measures time needed by a particle to reach the detector from the collision point. Since the length of the particle's trajectory can be obtained from the measurements from other detectors, the velocity of the particle can be determined. Also, momentum information can be retrieved, which, when combined with the velocity value, can give the mass of the particle. Charged particle mass uniquely determine its identity. The TOF is composed of Multigap Resistive Plate Chambers (MRPCs). They consist of two stacks of 400 μm thick glass plates separated by 250 μm thick gas gaps. Two electrodes create an electric field in the gas which is ionized by the passing particle. The MRPCs are organized into a cylindrical surface at a radius of 370 cm from the beam line. The TOF has total active area of 141 m^2. There are approximately 160 000 MRPC pads. The time resolution of the TOF is about 100 ps which gives very accurate indication of the particle type. For example, kaons can be distinguished from the pions with a 2 sigma statistical significance up to momenta of 3 Gev/c.

The High Momentum Particle Identification Detector (HMPID)

The HMPID is a Ring-imaging (RICH) detector used to determine the velocity of particles beyond the momentum range available through energy loss and time-of-flight measurements. The speed of the high momentum particle is obtained by the Cherenkov radiation it emits by travelling faster than light in a particular material. The HMPID is the world's largest caesium iodide RICH detector, with an active area of 11 m^2. The momentum range of kaon-pion discrimination is up to 3 Gev/c and the momentum range of kaon-proton discrimination is up to 5 GeV/c.

2.2.3 Calorimeters

Calorimeters are the detectors responsible for energy measurements. All particles except muons and neutrinos deposit all their energy in the calorimeters by means of electromagnetic or hadronic showers. Photons, electrons and positrons deposit all their energy in an electromagnetic calorimeter. Their showers cannot be distinguished, but, since the photon carries zero charge, it can be identified by the non-existent track in

the tracking system of the ALICE detector. The photons can tell us information about the temperature of the system (thermal radiation). In order to measure such radiation, other special detectors are also necessary.

The Photon Spectrometer (PHOS)

The PHOS is a high resolution electromagnetic calorimeter specialized for photon identification. It can provide measurement for the research of the thermal and dynamical properties of the initial phase of the collision. It is made up of fast scintillating lead tungstate ($PbWO_4$) crystals and uses Avalanche Photodiodes (APD) for readout. ($PbWO_4$) is extremely dense and stops most of the photons. The energy range detectable by PHOS is from 0.5 Gev up to 10 GeV.

The Electro-Magnetic Calorimeter (EMCal)

The EMCal improves the high momentum particle measurement capabilities of the ALICE detector. It is a shashlik-type lead-scintillator calorimeter composed of 4 416 modules that are grouped into 20 super-modules (SM). Each of the modules consists of 4 optically isolated towers which gives a total of 17 664 individual towers. The readout from the EMCal is done by wavelength shifting fibers. The EMCal is the outmost detector in the central barrel. It is 24 cm thick and weighs about 100 tonnes. It covers an azimuthal angle of 110 degrees and a pseudorapidity range of $|\eta| < 0.7$. Also, it is used for the reconstruction of high energy jets, which can give an insight in the physics of hard processes.

The Photon Multiplicity Detector (PMD)

The PMD measures the multiplicity and the spatial distribution of photons produced in the collisions. It consists of two layers, each with 24 gas-filled modules. Each module consists of an array of closely packed hexagonal proportional counters, with wire readouts. Charged hadrons are rejected using a charged particle veto (CPV) in front of the converter. The photons pass through the converter, initiating an electromagnetic shower in a second detector layer.

The muon spectrometer

The muon spectrometer is designed to study the spectra of heavy quarkonia by analyzing their decay in the $\mu^+\mu^-$ channel. The design of the muon spectrometer is based on the fact they are the only charged particles able to pass almost undisturbed through any material. Muons with momenta under a few hundred GeV/c do not produce electromagnetic showers. For this reason a set of absorbers is installed in order to isolate muons. There are three absorbers, a passive front absorber, which absorbs hadrons and photons from the collision, an inner beam shield to protect the chambers from particles produced at large rapidities, and a passive muon-filter wall. It also consists of a tracking system of 10 detection plates, four planes of trigger chambers, and one large dipole magnet. The muon spectrometer is located outside of the central barrel in the pseudorapidity region of -4.0 $\leq \eta \leq$ -2.5.

2.2.4 Forward detectors

The ALICE detector is also equipped with so-called forward detectors located in the forward region which are used to measure the global characteristics of the collision event.

The Forward Multiplicity Detector (FMD)

The FMD consist of 5 large silicon discs, each with 10 240 individual detector channels. It detects charged particles emitted at small angles relative to the beam. The FMD provides a measurement of charged particle multiplicity. It also can give insight on the inclination of the event planes and shape variables of the event.

The V0 Detector

The V0 detector is a common name for two detectors installed on both sides of the interaction point. The two parts, V0-A and V0-C are located 340 cm and 90 cm from the nominal interaction point. The V0 detector mainly serves as a trigger detector, as it provides minimum-bias triggers for the other detectors. It is also able to independently estimate the multiplicity of the event. Each of the V0s is made up of 32 scintillator counters.

The T0 Detector

The T0 detector is used as a trigger and luminosity detector. The measured interaction time is used as the reference signal and is crucial for the functioning of the TOF detector that is used for particle identification. Also, it is used to measure the vertex position of the collision. The T0 detector consists of two arrays of Cherenkov counters (T0-A and T0-C) which are located at both sides of the interaction point at distances of 375 cm and 72.7 cm, respectively. Each array has 12 cylindrical counters equipped with a quartz radiator and a photomultiplier.

The Zero Degree Calorimeter (ZDC)

There are two ZDCs located 115 m on both sides of the nominal interaction point. They measure the energy of the spectator nucleons in order to determine the overlap region in the nucleus-nucleus collisions. This way, centrality of the collision can be measured. The ZDC has a distinct proton calorimeter (ZP) and a neutron calorimeter (ZN). ZPs are made of stack of brass plates grooved to allocate a matrix of quartz fibers. The ZNs are made of tungsten alloy plates.

Chapter 3

Machine learning

Machine learning is a field of study in programming and computer science which aims to build and understand adaptable methods and models that „learn" without being explicitly „told" what to do. The basic assumption behind machine learning is that algorithms, methods or models which worked in past are likely to work in the future. More specifically, given some training data, a model will learn important features and properties from which it could predict some future behavior. Such algorithms closely resemble optimization algorithms which aim to minimize a specific loss of some kind to better describe a dataset. However, the goal of machine learning algorithms is a generalization in which one is concerned with minimizing loss on the unseen examples. Machine learning techniques can be very powerful in tasks where human knowledge is not complete, like in science, or in tasks where humans posses knowledge but cannot explain it, for example in speech recognition.

The idea of self-teaching algorithms exists for a long time, but with recent development of suitable hardware, machine learning and other artificial intelligence frameworks experienced a rapid development in the last few years. Today, machine learning algorithms are involved in a wide variety of applications in medicine, speech recognition, computer vision, etc. Also, there is growing number of scientists that use machine learning methods in their fields of study. Given the enormous amount of the data provided by the high energy physics experiments one can assume that such an environment is suitable for the usage of machine learning algorithms in data analysis. There are many examples of applications of such an algorithms in the scientific collaborations that work on modern particle accelerators. For example, such algorithms are already in use in tagging of jets initiated by the „beauty" and „charm" quark [33]. There is a growing interest in utilizing machine learning in search for rare unknown particles (dark matter, supersymmetric particles etc.)[34].

3.1 Approaches to Machine learning

There are three main approaches to the machine learning algorithms. In supervised learning, the machine is presented with inputs and the desired outputs. The machine then tries to find a general rule or a function that maps inputs to outputs. In unsupervised learning, only the input data is given to the machine. In principle, the goal in that case is to find hidden patterns in the data. Finally, in reinforcement learning, the algorithm interacts with some dynamic environment where it aims to achieve a specific goal. Feedback for the actions of the algorithm is given by the environment itself.

3.1.1 Supervised learning

Supervised learning algorithms are the easiest to understand of the three mentioned. They use labeled training data with pairs of input data and the desired outputs. The goal is to find an algorithm that creates a function which maps the data from the input space to the output space. For example, if we want to distinguish dog pictures from cat pictures, assuming the pictures are labeled (dog - 1, cat - 0), the algorithm should find a function that maps inputs (pictures, however they are represented) into interval from zero to one. Ideally, the output of the function for a given picture would represent the probability of seeing a dog picture. In most cases, learning is done by minimizing some *loss* function averaged over the training data. This is some metric that describes the difference between the predicted and desired output. The main two categories of supervised learning tasks are classification, where the outputs of the model are discrete and associated with desired class, and regression, where outputs are continuous rather than discrete.

There is a wide range of supervised learning algorithms, such as support vector machines, decision trees or neural networks (described in detail in later sections). Each of them posses strengths and weaknesses. However, there is not a learning algorithm that works best for all supervised learning problems. The choice of a learning algorithm depends of the type of the problem one wants to solve.

There are several generalizations of the traditional supervised learning. Semi-supervised learning denotes the case where only a subset of data is provided with the desired outputs. In weak supervision one must deal with noisy, limited labeling of the training data. Active learning assumes that all of the training examples are not given at the start and the algorithms interactively collect new data.

3.1.2 Unsupervised learning

Unsupervised learning is a framework where an algorithm is given some input data without any desired outputs. The main goal of such an algorithm is to find some hidden patterns and correlations in the data. There is a variety of tasks that demand some form of unsupervised learning. For example, in clustering, one wants to group data points in some unknown categories. Also, some algorithms are used for dimensionality reduction of the input data. In that case, one wants to represent input data with the vectors of lower dimension, but with the minimal loss of information. For such a task, the most important independent features in the input data must be extracted. Principal Component Analysis (PCA) is an example of the such an algorithm [35]. Some generative models which generate data that mimics given training data are also examples of unsupervised learning. Anomaly detection problems and wide range of other tasks often use unsupervised learning algorithms.

3.1.3 Reinforcement learning

Reinforcement learning (RL) is a field of machine learning concerned with the notion of how to take actions in an environment. In reinforcement learning one does not need labelled input/output pairs. Also, it does not need an explicit correction of some wrong action. The goal is find a balance between exploration, where actions are taken sometimes randomly to investigate some unexplored possibilities, and exploitation, where the best action is taken based on the current knowledge. Partially supervised RL algorithms combine the advantages of supervised and RL algorithms.

The RL environment is usually modelled by the Markov decision process (MDP) [36]. However, reinforcement learning algorithms do not assume knowing an exact mathematical model of the MDP, in difference with the classical dynamic programming methods. In general, the goal is to find the optimal „policy", i.e. the probability of transition from the one state to another in the environment under some action by maximizing the „reward" function which represents some feedback from the environment. RL algorithms can be powerful in problems where the model of the environment is known, but there is no analytic solution, and in problems where you gather knowledge by interaction with the environment. An example of the latter is the game of chess where Google Deep Mind developed AlphaZero algorithm using RL [37].

3.2 Data overview

Since machine learning models use examples to „learn", it is important to collect good data which should be represented accordingly to the used algorithm. Quality data can be the key to a successful machine learning model. Data is regarded as a physical representation of some information. A collection of data instances is called the dataset. The data instances are usually represented by a number of features which capture some piece of the corresponding information. There are various ways how the data can be represented since many factors enter its form, such as equipment used to collect the data, processing operations and storage requirements.

3.2.1 Types of data

In most cases, the data is represented as a set of independent records described by certain pre-specified features. For instance, an iris flower can be represented by four features: sepal length, sepal width, petal length and petal width. In this case, each feature is described by a continuous numeric value. A feature can be described as a categorical entity as well. This example is fairly simple, but if we want to describe pictures of an iris flower with the size of 256×256 pixels, each pixel is regarded as a feature with its numeric value. Mathematically, data is often represented by the means of the vectors and tensors, in which each component describes a particular feature. Data from the above examples would be represented by a vector with a size of 4 and a rank 2 tensor with a size of 256×256, respectively.

In sequenced data, such as in texts or in time-series data, the order of information entities or features is relevant and contains additional information. It can also be represented as a simple vector, but one should be careful to take ordering information in the design of the algorithm.

In some cases data is structured, so links and interactions between information entities or features are very important. An example is a description of chemical molecules. This kind of the data is regarded as graph data. There are various ways to mathematically describe graph data, one of them being adjacency matrix or matrices.

3.2.2 Data quality

Real world datasets will almost certainly contain errors. These errors can be caused by various things, like human errors, imperfect devices, bad data processing and so

on. They can be grouped into various categories, such as noise, outliers, missing values, inconsistent feature values or duplicates. These errors can significantly degrade the learning performance. Also, in many cases one deals with sparse data. The feature space is large but only small number of features are effective. This also presents a challenge in learning performance. Dealing with imbalanced data in supervised learning problems, where data is labelled mostly in one or a few ways, is an issue too. It is important to analyze the dataset before starting machine learning tasks. Thorough cleaning of the dataset and transforming or preprocessing operations can be crucial to the performance of a machine learning algorithm.

3.3 Standard issues in ML algorithms

Machine learning has found success in many disciplines, but sometimes machine learning techniques fail to deliver expected results. This is caused by a number of possible reasons, like lack of suitable data, biased data, wrong choice of algorithms and methods, lack of resources and others. Often, machine learning models have to be specifically tuned in order to work properly, which is often not a very easy task. This section will deal with the most common general issues one can encounter when dealing with a machine learning task.

3.3.1 Underfitting

In some cases, used models and algorithms simply do not have a capacity to solve a particular task. The variety of data is too big for a model to map inputs to desired outputs. In other words, the task is too hard for the given model and it did not learn anything. One says that the model underfits.

3.3.2 Overfitting

Overfitting appears when a trained model reflects the noise in the given training data too much. Such models does not generalize well. It is as if someone learns (remembers) the whole book but does not know anything outside of it. This often happens when a model is overly complex. In that case, there are a lot of ways a model can fit the data and no one can guarantee that its solution is the real solution. One of the ways to overcome overfitting is an early stopping of the training process. Also, there exist a

number of regularization techniques which dynamically decrease the complexity of the model. In most cases, the principle that the simplest solution is the best solution works. Since neural networks are especially prone to overfitting due to their complexity, some of the regularization techniques will be explained in the section that describes neural networks.

3.3.3 The curse of dimensionality

Sometimes, when one must deal with highly dimensional data it becomes very challenging to handle it due to limited computational resources, among other reasons. Also, most of machine learning algorithms have a complexity of the order at least $\mathcal{O}(n^2)$. This causes the usage of unnecessarily complex models. To deal with such a challenge, some data processing techniques are used, like dimensional reduction and feature selection. They effectively reduce dimensionality by projecting the data instances to a lower dimensional space or by selecting fewer features for their description.

3.4 Machine learning algorithms

There is a wide range of machine learning algorithms. Choosing which one to use depends on the specific characteristics of the task one wants to address. There is no exact way to tell which algorithm will perform the best. Data scientists often rely on their intuition and some general rules to pick the right algorithm. The most commonly used machine learning algorithms are described in this section. Since a neural network algorithm, as an example of a machine learning algorithm, is widely used in this thesis, they will be described in detail in a separate section.

3.4.1 Linear regression

The most basic form of a machine learning algorithm is the linear regression. It is a linear model where the relationships are modeled using linear functions whose parameters are estimated from the data. Let's say there is a dataset consisting of a pair of variables, one independent and one dependent. The goal is to find a line which describes data the best, $y = Ax + B$. The parameters A and B are estimated by the

minimization of some error, in most cases the Mean Squared Error (MSE). Linear regression can also be generalized to vectors, where some set of variables ($\mathbf{y}$) depends on multiple independent variables ($\mathbf{x}$). In that case A is a matrix, and B is a vector. Its basic assumption is that variables depend linearly and it is unable to capture nonlinear correlations.

3.4.2 Logistic regression

Despite its name, logistic regression is in fact a classification algorithm. It wants to predict the probability of the some binary outcome, like in coin flipping. Suppose a dataset consisting of independent variables x and associated outcomes y (denoted by 0 or 1) is presented. The goal of the regression is to estimate the probability that the outcome will be one for a given x, $p(1|x)$. Then, the probability of the outcome of zero is simply $p(0|x) = 1 - p(1|x)$. In the algorithm, the probability is modelled by the means of the logistic function defined as:

$$p(x) = \frac{1}{1 + e^{-\frac{x-\mu}{\sigma}}} \tag{3.1}$$

where μ is a location parameter and σ is a scale parameter. The special case of logistic function where $\mu = 0$ and $\sigma = 1$ is called the *sigmoid* function. In logistic regression, the parameters μ and σ are usually estimated by maximizing the log-likelihood. This is identical to minimizing the *binary crossentropy* loss defined as

$$L = -y \ln p(x) - (1 - y) \ln (1 - p(x)) \tag{3.2}$$

where $p(x)$ is the output of the model for a given input variable x, and y is a desired outcome. In practice, an averaged loss over the all data points is minimized.

3.4.3 Support Vector Machine (SVM)

SVM is one of the most popular algorithms in supervised learning tasks. It is primarily used for classification problems, although it too can be used in regression tasks. The goal of the SVM is to find a hyperplane which classifies the data points the best. The dimension of the hyperplane depends on the number of features in data instances. The estimated hyperplanes are used as decision boundaries. Data points positioned on either sides of the hyperplane are considered to belong to different classes. The support

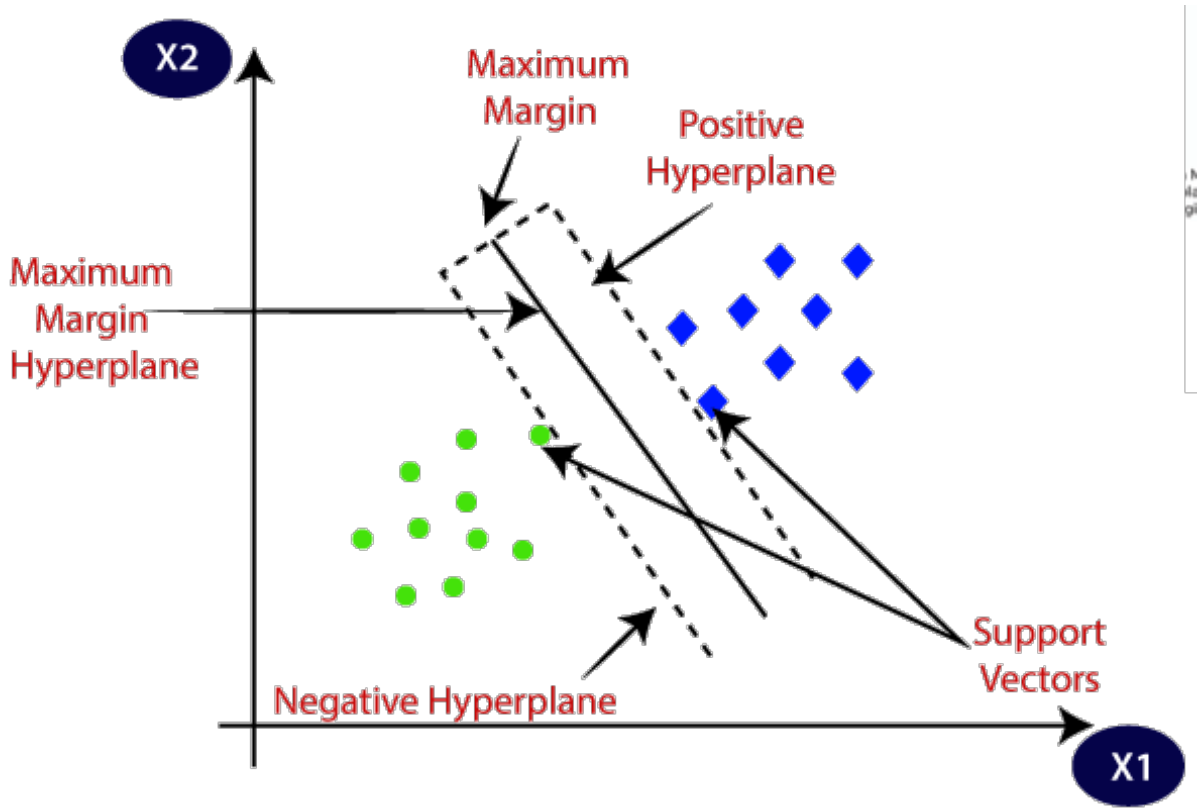


FIGURE 3.1: A visualization of support vectors [38].

vectors are the vectors closest to the hyperplane. The position and the orientation of the hyperplane depend on these support vectors. The objective of the SVM algorithm is to maximize the distance between support vectors and the hyperplane, also called the margin. Often, this is done by minimizing the *hinge* loss which can be defined as:

$$L = \sum_i \text{Max}(0, 1 - y_i d_i)\,, \tag{3.3}$$

where y_i is the desired output for a given data point $\mathbf{x}$ and d_i is the distance between a given data point $\mathbf{x}_i$ and the hyperplane. If a hyperplane is defined by parameters $\mathbf{w}$ and b as a collection of points $\mathbf{x}$ that satisfy $\mathbf{w}^T\mathbf{x} - b = 0$, the distance d_i is equal to $\mathbf{w}^T\mathbf{x}_i - b$.

3.4.4 K-Means

K-Means is a clustering algorithm and belongs to the category of unsupervised learning algorithms. This algorithm aims to partition collection of data points into k clusters,

where k is a predefined number. It wants to find k centroids which are points that represent a center or a mean of a cluster. Then, it allocates each data point to the nearest cluster center. Centroids are found by minimizing the average distance between data points of a particular cluster and its centroid. It is often done in an iterative way where the initial locations of centroids are set randomly.

3.4.5 K-Nearest Neighbours (kNN)

The kNN is a simple supervised learning algorithm which can be applied on both classification and regression problems. The algorithm assumes that similar things are closer to each other, where „distance" is defined by some metric. The most popular choice used is the Euclidean distance. k stands for a user-defined number of neighbours. The kNN works in the following way. Assuming k is determined and a metric is defined, for each data point it calculates the distances to other data points. These values are then sorted and k data points with smallest distances are chosen. Next, labels of the nearest k neighbours are used. In classification, for a given data point, the algorithm outputs the mode of the k labels, i.e. it outputs the label that appears most often. In regression, it outputs the mean of the k labels.

3.4.6 The Decision Tree

The Decision Tree algorithm aims to create a model which makes a prediction by learning simple decision rules. These rules have a hierarchical, tree-like structure, hence the name decision tree. It starts from the root node with a goal to find an attribute or a feature of the data point which splits the data the best according to some labels. If the values of the features in a data point are continuous, they need to be discretized. Selecting the best attribute is done by means of the information gain which is defined as the difference in Shannon entropy and the average Shannon entropy of the subsets created by the decisions on a given attribute [40]. The Shannon entropy of a set S is in this case defined by:

$$H(S) = -\sum_i p_i \ln p_i \,, \tag{3.4}$$

where the index i iterates over the number of classes and p_i is the probability that a particular class appears in the set S. In other words, p_i is a fraction of data points in the i-the class with respect to the total number of data points in the set S. The best attribute

is considered as the one with the highest information gain. This procedure is done recursively on each node of the tree. The algorithm continues to split on attributes until either it classifies all the data points or there are no more attributes to split on. In other words, there is no more information gain. After the tree is built in the previously described way, the decisions are made on the basis of the calculated best attribute for each node. Decision trees provide a clear insight into fields which are the most important to make some prediction. A model built this way is highly interpretable. It also does not require much computational power to make predictions. However, it can be computationally expensive to train them. Also, it is prone to overfitting. Decision trees can

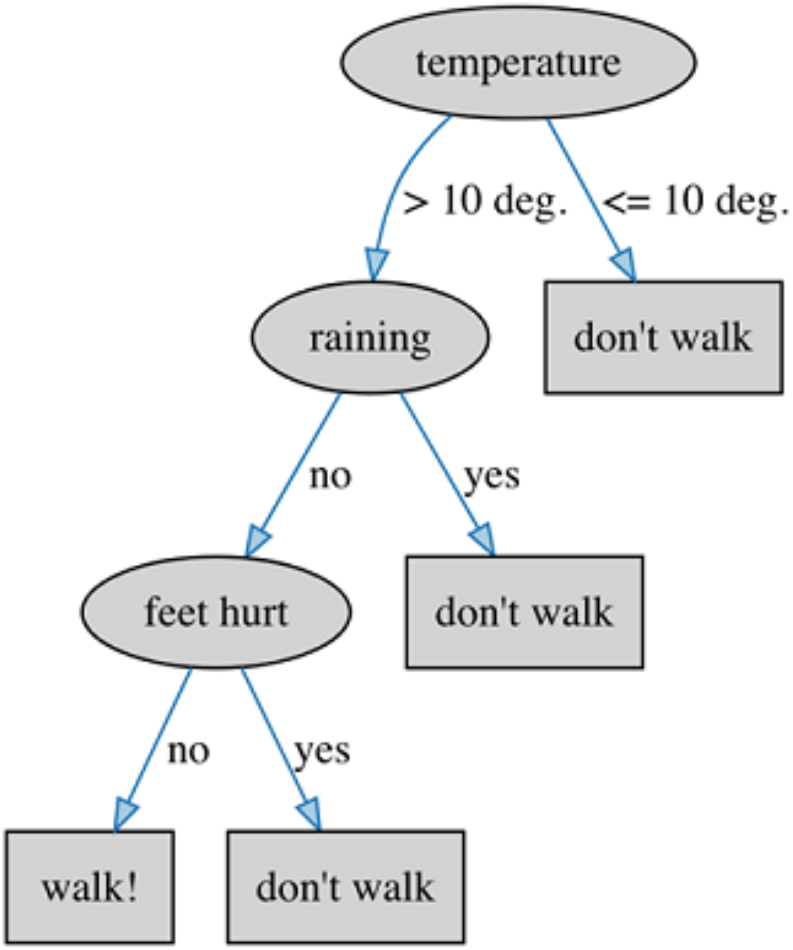


FIGURE 3.2: An example of a decision tree [39].

be sequentially combined to boost their performance, in which case we have Boosted Decision Trees (BDT) [41]. Using the techniques of bagging or bootstrap aggregating it is possible to create an ensemble of decision trees to construct the Random forest algorithm, which in general has a better performance than an individual tree [42].

3.5 Neural networks

A neural network is an algorithm based on a set of inter-connected objects called artificial neurons. They loosely model the neurons in a biological brain. Like neurons and synapses in an animal brain, artificial neurons receive some signal, usually a real number, from other neurons and applies some, usually non-linear, function to the weighted sum of the inputs from the other neurons and transmits it to the next neuron, also in the form of the real number. The history of neural networks started in 1943 when Warren McCulloch and Walter Pitts developed a computational model for neural networks []. Since then, a lot of researchers worked on the topic, but there was not sufficient computer support and only pretty small and primitive networks were used. Over decades, steady progress was made in a field, with a rapid development observed by usage of modern GPUs and distributed computing, which allowed an application of very complex and deep neural networks. Such networks proved to be very successful in fields like computer vision and speech recognition. Use cases of neural networks are multiple, such as function approximation, pattern recognition, data processing, etc. They are suitable for both classification and regression tasks. Also, they can be used in supervised, unsupervised and reinforcement learning approaches.

3.5.1 Artificial neurons

An artificial neuron is the basic building block of a neural network. It receives some, often multiple, inputs, whether from external user input or from other neurons and outputs some function of the received inputs. Typically, each input is weighted and summed with others. A certain bias value is added to the sum. In the end, a neuron outputs some function of that value. Mathematically, it can be described as:

$$y = f\left(\sum_i w_i x_i + b\right), \tag{3.5}$$

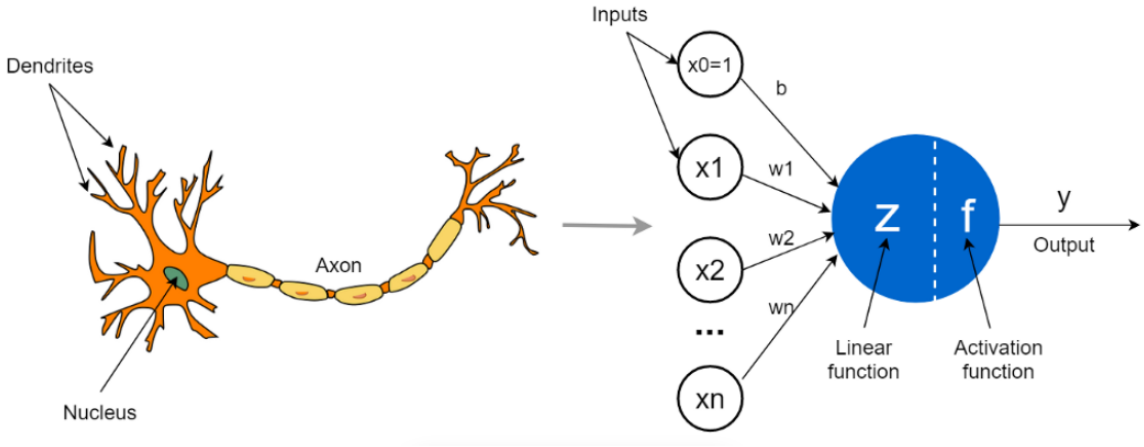


FIGURE 3.3: The neural network neuron in comparison to the human brain neuron [43].

where x_i denotes inputs, w_i weights associated with given inputs and b a bias value. $f(x)$ is some activation function. The most popular activation functions are the *sigmoid* function and the Rectified Linar Unit (ReLU) function, defined as $f(x) = Max(0, x)$. The activation function is important and is needed in order to enable a neural network to capture non-linearities in data which increases its learning capacity. The neurons are typically organized in one or multiple layers. The Input layer is a collection of neurons that receive external data. The Output layer is a collection of neurons that output a final result. In between them, there can exist any number of hidden layers, which are collections of neurons that neither receive input data nor output the final result. A neural network without a hidden layer is called a simple neural network. Complex neural networks with multiple hidden layers are called deep neural networks and are associated with the deep learning paradigm. In general, each neuron from one layer is connected to each neuron in the next layer. Then, the neural network forms a directed a-cyclic graph and is called a *feed-forward* neural network. When layers have connections in the same layer or to a previous layer, we talk about *recurrent* neural networks. Mathematically, a layer can be described as vectorized analogue of 3.5:

$$\mathbf{y} = f\left(\mathbf{W}\mathbf{x} + \mathbf{b}\right), \tag{3.6}$$

where $\mathbf{x}$ denotes the input vector and $\mathbf{y}$ the output vector of the layer. $\mathbf{w}$ is a weight matrix which transforms the input vector $\mathbf{x}$ by operation of matrix multiplication and

b is a bias vector which contains a bias value of each neuron in the layer. The activation function f is applied element-wise on the vector $\mathbf{W}\mathbf{x} + \mathbf{b}$. Neural networks, as described above, are called fully-connected and display the most general case of a *feed-forward* neural network. The goal of the neural network is to find the parameters of the weight matrix **W** and the bias vector **b** for each layer, which solves a problem the best. Assuming these parameters are found, predictions are made simply by using 3.6 where each layer receives inputs from the previous layer and transmits the output to the next layer.

Sometimes, fully-connected networks are too complex and prone to overfitting. In order to address that issue and achieve better performance, various architectures of neural networks are suggested. An architecture is defined by a constraint put in connection weights suitable for a specific problem. This kind of neural network design reduces the degrees of freedom a neural network can have and directs the neural network to learn in the right way. The best example of the neural network architecture is a Convolutional Neural Network (CNN), where a large number of weights are shared by its neurons.

3.5.2 Optimization algorithms

In this section we describe how a neural network learns. Optimization algorithms are designed to find the right weights for a given neural network. They are based on the concept of minimizing some *loss* function which somehow describes the difference between what we have and what we want to have. The idea is to use gradients of the neural network to iteratively find minima of the *loss* function, since neural networks consist of differentiable multivariate functions (assuming all of their activation functions are differentiable). At first, one needs to find how much a change of a particular parameter of the neural network affects the *loss* function. Then, this parameter is updated accordingly to the amount of change it generates. Basically, one wants to make larger changes to parameters which reduce the value of the *loss* function the most. Calculating gradients of the *loss*-function is done by the algorithm of *backpropagation*. There are several algorithms which use calculated gradients somewhat differently, all of them trying to stabilize the training process. These algorithms, along with the *backpropagation* algorithm, are described in the following sections.

Backpropagation

Differentiating a composition of functions is done using the chain rule. Since a neural network is basically a composition of functions the chain rule is used as a basis for the *backpropagation* algorithm.

Given an input/output pair (x, y), the scalar *loss* function is given by $L(y, f(x))$, where f represents the whole neural network. As described above, a neural network is a composite function $f(x) = f_N(f_{N-1}(..f_1(x)))$, where f_i represents a transformation in the i-th layer. One wants to find the gradient of a *loss* function in terms of weight parameters. Assuming the output of the i-th layer is equal to

$$\mathbf{f}^i = \sigma(\mathbf{z}^i) = \sigma(\mathbf{W}^i \mathbf{f}^{i-1} + \mathbf{b}^i) \tag{3.7}$$

we have:

$$\frac{\partial L}{\partial W^i_{jk}} = \frac{\partial L}{\partial f^i_j} \frac{\partial f^i_j}{\partial z^i_j} \frac{\partial z^i_j}{\partial W^i_{jk}} \,. \tag{3.8}$$

The last element in the product is easy to calculate since there is an explicit connection between z^i_j and the weight matrix element W^i_{jk}. The first two elements of the product are denoted by δ^i_j:

$$\delta^i_j = \frac{\partial L}{\partial z^i_j} \,. \tag{3.9}$$

Since δ is a vector, this expression can be written in terms of the gradient, $\delta^i = \nabla_{\mathbf{z}^i} L$. Now, one wants to see in what way the gradients between two consecutive layers connect. We will assume the vector δ^{i+1} is known. Since each component of the vector $\mathbf{z}^{i+1}$ is a function of all components of the vector $\mathbf{z}^i$ and there is an explicit dependence between two vectors,

$$\mathbf{z}^{i+1} = \mathbf{W}^{i+1} \sigma(\mathbf{z}^i) + \mathbf{b}^{i+1} \,, \tag{3.10}$$

the j-th component of the vector δ^i can be written as

$$\delta^i_j = \frac{\partial L}{\partial z^i_j} = \sum_k \frac{\partial L}{\partial z^{i+1}_k} \frac{\partial z^{i+1}_k}{\partial z^i_j} = \sum_k \delta^{i+1}_k W^{i+1}_{kj} \sigma'(z_j) \,. \tag{3.11}$$

It can be seen that the right side of the above expression is in fact a matrix multiplication of the transposed weight matrix, $(\mathbf{W}^{i+1})^T$ and the vector δ^{i+1}. The full vector form

of the 3.11 is written as

$$\delta_j^i = (\mathbf{W}^{i+1})^T \delta^{i+1} \circ \sigma'(\mathbf{z}) \,. \tag{3.12}$$

The symbol $\circ$ denotes the Hadamard product, an element-wise product between vectors. It can be noticed that each δ^i can be calculated iteratively if the last, δ^N, is known. Indeed, it is easily calculated since the *loss* function explicitly depends on the output of the neural network. The gradients of the loss function in terms of weights and biases are, according to 3.8, simply calculated if the inputs from the previous layer are used. Inputs of the first layer are external data inputs.

$$\frac{\partial L}{\partial W_{jk}^i} = \delta_j^i \sigma(z_k^{i-1}) \quad \frac{\partial L}{\partial b_j^i} = \delta_j^i \tag{3.13}$$

This algorithm is called *backpropagation* since it propagates the error on the *loss* function backwards from the last layer.

Gradient Descent

The Gradient Descent algorithm is designed to iteratively update the parameters of a neural network to minimize a given convex *loss* function. In the beginning, the weights are parameterized randomly by some small numbers. In each step, the weights are updated in the following way:

$$W_{jk}^i \to W_{jk}^i - \alpha \frac{\partial L}{\partial W_{jk}^i} \,, \tag{3.14}$$

where the parameter α is called the *learning rate*. It is a very important parameter and it should be neither too large nor to small. It is assumed that the *loss* function is calculated over the entire dataset. In contrast, when using the Stochastic Gradient Descent (SGD), the *loss* function is calculated only on one data instance in each step of the algorithm. In practice, the most commonly used gradient descent algorithm is the Mini-Batch Stochastic Gradient Descent (MB-SGD) which, in each step, evaluates the *loss*-function on some small subset of data called a batch. Typically, there would be 32 or 64 data instances in a mini-batch. It is less noisy than a regular SGD does not have a large memory requirement for data storage. In order to reduce noise, the MB-SGD

with momentum is sometimes used. It introduces a momentum defined by

$$V_{jk}^{i}(t) = \gamma V_{jk}^{i}(t-1) - \alpha \frac{\partial L}{\partial W_{jk}^{i}}, \tag{3.15}$$

where the parameter γ is pre-defined and usually takes on the value of 0.9. The variable t represents the iteration of the algorithm. Consequently, each iteration reduced the value of $V_{jk}^{i}(t)$. The idea is to denoise the gradients in this way by giving more importance to the recent updates compared to the previous updates.

Adaptive Momentum Estimation (AdaM)

The principle of work of the AdaM algorithm was introduced in a paper from 2015 by Diederik Kingma and Jimmy Ba [44]. It combines the advantages of AdaGrad and RMSProp algorithms [45], [46]. It is used for the neural network optimizations in this thesis. Empirically, it provides very good performance. It is straightforward to implement and computationally efficient. It is well suited for neural networks with a large number of parameters and problems with sparse or noisy gradients.

3.5.3 Regularization techniques

Regularization techniques are used to overcome the problem of overfitting. Generally, fully connected neural networks are very powerful, but sometimes too complex and prone to overfitting. In this section some commonly used regularization techniques are described.

L2 regularization

The L2 regularization is the most common type of all regularization techniques and is also commonly known as *weight decay*. It aims to control the weight parameters by minimizing their Euclidean norm. It is done by adding the term $\Omega(W) = \|W\|_2^2 = \sum_{ij} W_{ij}^2$ weighted by some parameter α to a regular loss function one wants to minimize. The parameter α is a new hyperparameter of the neural network and is called the *regularization rate*. It quantifies how much one wants to regularize. The consequence of this method is that weights are on average smaller after each update. This way, the complexity of a neural network is reduced.

L1 regularization

The L1 regularization, also known as Lasso regression, is very similar to the $L2$ regularization in implementation. In it, the sum of the absolute values of the weight parameters in a weight matrix is added to a *loss* function, $\Omega(W) = \|W\|_1 = \sum_{ij} |W_{ij}|$. The aim of the L2 regularization is to minimize the weights as much as possible, but the goal of the L1 regularization is to have as many of them equal exactly zero. One needs to be careful in choosing the parameter α. If it is too high, it will overshadow the regular loss and the model will not converge. If it is too low, the regularization will be negligible and very likely overfitting will take place.

Dropout

In addition to the above described L2 and L1 regularization techniques, another widely used and powerful technique is the dropout regularization. The implementation of the dropout regularization is quite simple. During the training of the neural network a random neuron gets turned off, i.e. it outputs zero regardless of its input. The probability p of a neuron to turn off is set by the user and is another hyperparameter of the neural network. The dropout prevents neurons from adapting to each other too much, and as a consequence one has sparsely populated weights across the neural network. Therefore, it effectively reduces the complexity of the neural network [47].

3.5.4 Convolutional Neural Network (CNN)

The architecture of the Convolutional Neural Network is typically constructed from a few convolutional layers, each paired with a pooling layer. The last few layers are generally fully-connected layers. The network aims to capture local patterns in some high dimensional data, for example pictures represented by pixels, and transform them to some abstract features which are then fed to the fully-connected layers. The convolutional layer is a special case of a fully connected layer where the weights are shared between neurons. The pooling layers are often non-parametric layers which reduce the dimension of a vector or a tensor by combining adjacent elements into one. These kinds of neural networks proved to be very successful in fields of computer vision, image and video recognition, significantly improving performance and speed of convergence.

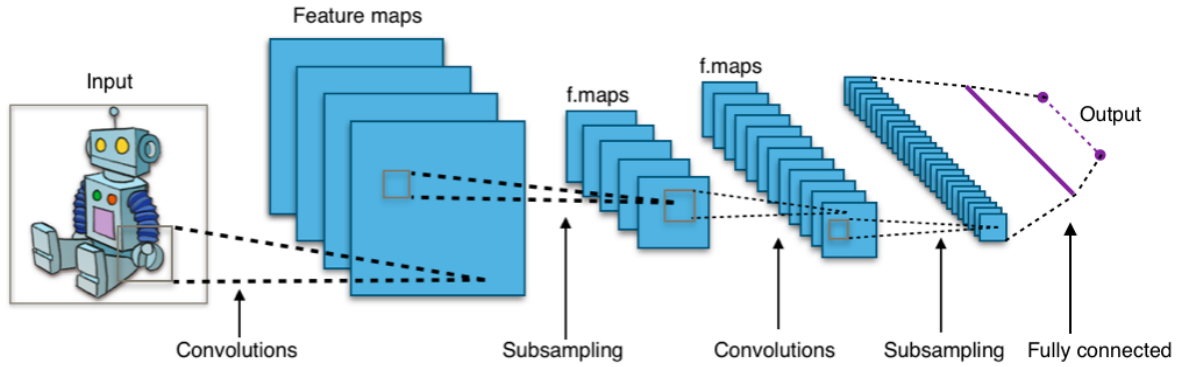


FIGURE 3.4: A typical architecture of the convolutional neural network [48].

The convolutional layer

The convolutional layer often takes tensor of rank 3 as an input. The axes of the tensor correspond to width, the height, and the number of channels. For example, a two-dimensional picture can have a shape of $256 \times 256 \times 3$, where an image has a size of 256 pixels in width, 256 pixels in height and there are three channels for color, assuming the RGB coloring scheme is used. It transforms the input into an abstracted feature map. Weight sharing in the convolutional layer is done by using filters or kernels. They can be regarded as fully-connected layers which receive only a low-dimensional part of the original input and output a single number. In general, a filter of size 3×3 is used. Each filter with its weights slides over an input image and constructs a feature map by applying itself on different parts of the image. This operation is called convolution since it closely resembles a mathematical convolution. A convolutional layer can have multiple different filters, with their number corresponding to the number of channels of the feature map which the layer outputs. Assuming the input tensor has a size of $256 \times 256 \times 3$ and the layer has 8 filters, the size of the output tensor will be $254 \times 254 \times 8$. The height and the width of the input image are slightly reduced because a filter of size 3×3 can move itself $256 - 2 = 254$ times in each direction. This way, one tends to recognize a local pattern in an image. Another important benefit of the convolutional layer is the introduction of translational symmetry. In other words, a specific important pattern will be recognized regardless of its position in the image.

The pooling layer

The purpose of the pooling layer is to reduce the height and the width of the feature map obtained from the convolution layer. Typically, 2×2 pools are used. It means that this layer takes a part of the feature map with a size of 2×2 and combines it into one singular value. The most common way is to take the maximum value of the numbers in a given pool. In that case the pooling layer is called Max Pooling. Sometimes, the way to combine values in a pool is to take an average of the values in the pool. Then the layer is called Average Pooling. After the application of a pooling layer of dimensions 2×2 the data size is reduced by a factor of 4. For example, assuming an input of $254 \times 254 \times 8$, the pooling layer outputs a tensor with a size of $127 \times 127 \times 8$. The goal of such an operation is to disregard non-important or redundant features in a feature map.

Layer organization

The first two layers of a CNN are usually a convolutional layer followed by a pooling layer. This stack of layers outputs a tensor that is less wide and high but more long than the input tensor. Generally, such a stack is repeated a few times producing even longer and very thin tensors. Gradually, the feature maps (outputs of each convolutional layer) become more and more abstract. In the end a feature vector is sent to a fully connected layer. The first convolutional layer typically recognizes edges and shapes. Deeper layers recognize connections and correlations between the shapes and objects in the picture. This way, fully connected layers in the end have an easier job because they receive a low dimensional feature vector with the most important features. The schematic of a CNN is shown in 3.4. The CNN have a significantly smaller number of free parameters than the fully connected neural network, which directly takes an image as input. With this specific design, the CNN's complexity is reduced and is less prone to overfitting. Also, it achieves faster convergence to a minimum of a *loss* function.

3.5.5 Autoencoders

Autoencoders are neural network models that try to approximate the identity function. They consist of two parts, the encoder and the decoder. The encoder aims to encode input data into some latent vector. Then, the goal of the decoder is to reconstruct

the original input from the latent vector. Training is done by minimizing some *loss* function between input and output, usually mean squared error,

$$L = \frac{1}{n}\sum_{i}^{n}(\mathbf{x}_i - D(E(\mathbf{x}_i)))^2 . \tag{3.16}$$

Here, D denotes the decoder and E denotes the encoder. In general, the latent vector has a much lower number of elements than the input vector. It forms an information bottleneck. Taking into account that the training process succeeded, the latent vector has most of the information needed to reconstruct the input, but with a much lower number of elements than the input. Since it needs only the input data, an autoencoder is used in unsupervised learning tasks. It is applied for the purpose of dimensionality reduction or feature extraction. Any architecture of a neural network can be used to design an encoder or a decoder. If encoder and decoder use convolutional layers, it is called an convolutional autoencoder.

3.5.6 Generative models

Until so far, we were describing discriminative models. In principle, given some input data $\mathbf{x}$, one wants to find a probability distribution of outputs, $P(\mathbf{y}|\mathbf{x})$. In generative models, the goal is to find the real probability distribution $P(\mathbf{x})$ and to be able generate some data from random noise according to this distribution. Generally, models don't explicitly calculate $P(\mathbf{x})$, but imitate this probability to generate data using some statistical methods. In this thesis, a generative model is presented which in fact estimates underlying distributions of the input data. In the following text, two of the most popular generative models based on neural networks are presented, the Variational Autoencoder and the Generative Adversarial Network.

The Variational Autoencoder

In a regular autoencoder, the elements of the latent vector are mutually entangled and highly correlated. This makes it difficult to interpret the features in the latent vector. If the elements of the latent vector are independent and follow some simple probability distribution, it could enable the usage of the decoder network to sample data from the latent vector by feeding it with some noise modelled by some simple distribution, for example a multivariate Gaussian distribution, $\mathcal{N}(0, \mathbf{I})$. The variational autoencoder is the regularized version of an ordinary autoencoder in which the latent vector is

forced to have independent and untangled elements. The variational autoencoder also consists of the encoder and the decoder. In contrast to an ordinary autoencoder, it uses two vectors. The latent vector $\mathbf{z}$ is then sampled by introducing some Gaussian noise

$$\mathbf{z} = \mu + \sigma\mathcal{N}(0, \mathbf{I})\,. \tag{3.17}$$

This way, the latent vector follows the multivariate normal distribution with a mean and variance μ and σ, $\mathcal{N}(\mu, \sigma)$ [50]. Then, the vector is fed to the decoder which tries to reconstruct the input vector x. In the training process one wants to minimize the difference between the input and the output vectors, but also wants that the probability distribution of the latent vector gets closer to the $\mathcal{N}(0, \mathbf{I})$. This is done by minimizing the Kullback-Leibler (KL) divergence between $\mathcal{N}(0, \mathbf{I})$ and $\mathcal{N}(\mu, \sigma)$. The KL divergence is a specific measure of the difference between probability distributions defined in []. The difference between $\mathcal{N}(0, \mathbf{I})$ and $\mathcal{N}(\mu, \sigma)$ in terms of the KL divergence is expressed by

$$KL(\mathcal{N}(0, \mathbf{I}), \mathcal{N}(\mu, \sigma)) = -\frac{1}{2}\ln\sigma + \sigma^2 + \frac{1}{2}\mu^2 - \frac{1}{2}\,. \tag{3.18}$$

This regularization term, averaged over the dataset, is added to the *loss* function shown in 3.16. The variational autoencoder simultaneously wants to achieve good reconstruction of the input and regularization of the latent vector. Sometimes, the KL term in the variational autoencoder loss is weighted by a parameter β. In that case, it is called β-VAE [51]. If trained properly, the decoder of the variational autoencoder can be fed with a data sample from $\backslash(0, \mathbf{I})$ and it would produce output data that resemble the input data $\mathbf{x}$. Because of this, the variational autoencoder is considered to be a

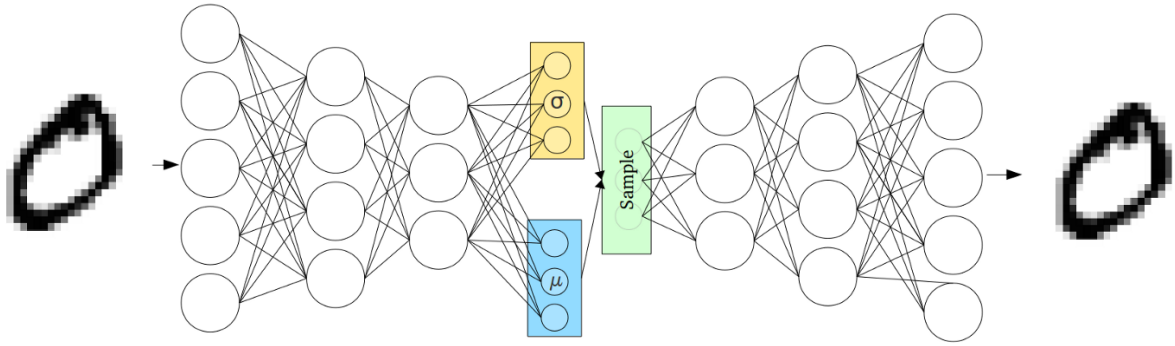


FIGURE 3.5: An architecture of the variational autoencoder [49].

generative model.

Generative Adversarial Network (GAN)

A generative adversarial network (GAN) is a generative model designed by Ian Goodfellow et. al. in June 2014 [52]. It consists of two neural networks, the generator and the discriminator, that are in contest with each other, hence the name adversarial. The basic idea is this: the generator samples some data vectors $\mathbf{z}$ from random noise, usually modelled by a multivariate normal distribution $\mathcal{N}(0, \mathbf{I})$. The outputs of the generator are fed to the discriminator together with the external real data $\mathbf{x}$. The discriminator is then trained to classify real data and data created by the generator into different categories. In other words, it wants to distinguish between real and fake data the best it can. Usually, this is done by the means of the *binary crossentropy* loss:

$$L = -\sum_i \ln D(\mathbf{x}_i) + \ln(1 - D(G(\mathbf{z}_i)))\,, \tag{3.19}$$

where the generator is denoted by G and the discriminator by D. In the next step the generator is trained to produce data which the discriminator will see as real. In this step, random noise is fed to the generator, and the objective is that a combined model consisting of both, the generator and the discriminator, outputs a value of one, assuming the real data is labeled by one in the previous step. The optimization is done by a gradient descent through the combined model, but only the generator's weights are allowed to be updated. The weights of the discriminator remain constant in this step.

$$L = -\sum_i \ln(1 - D(G(\mathbf{z}_i)))\,. \tag{3.20}$$

These two steps are repeated until the discriminator is not capable to distinguish between the real data and the data provided by the generator. Then its desired output is $\frac{1}{2}$, so that it assigns an equal probability to each input. The generator is then used as generative model to sample data instances that mimic the real data. Since the desired working point is not a minimum of the objective function, but a saddle point also called the Nash equilibrium [], the training process of the GAN proved to be unstable and hard to achieve. One can encounter a variety of issues, such as vanishing gradients, exploding gradients, mode collapse, a discriminator that is too strong, a discriminator that is to weak, etc. [52]. A wide range of solutions are proposed to address these

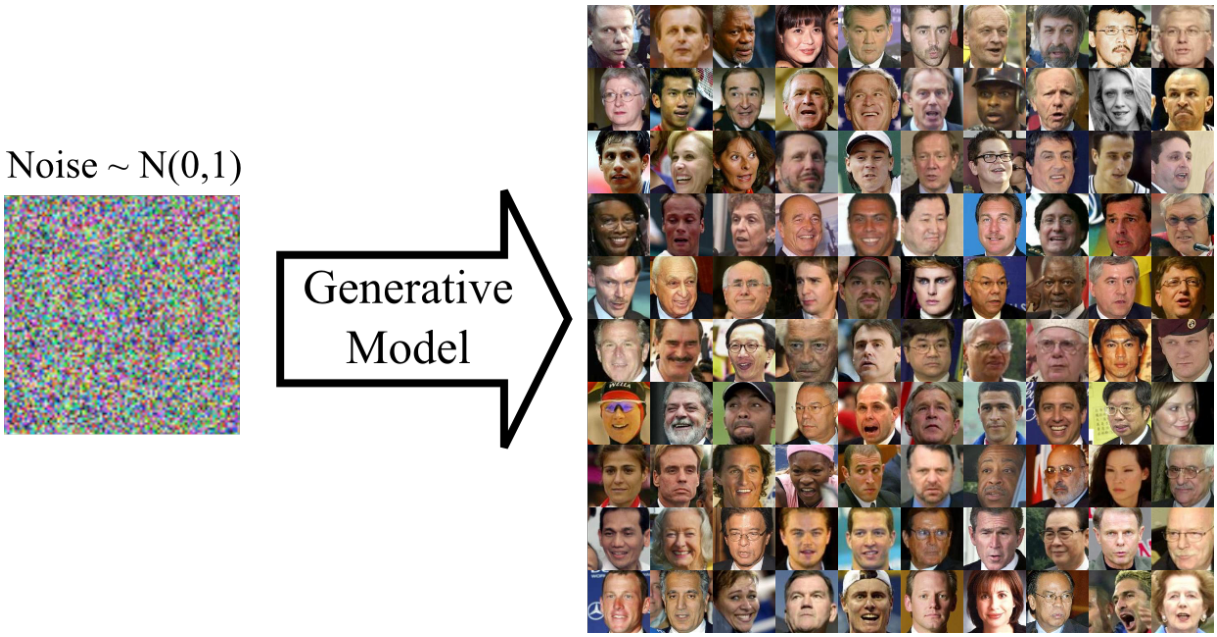


FIGURE 3.6: Noise passed through a GAN generates fake celebrity images [53].

issues. Because of that there are many variations which improve above the described GAN, like Wasserstein GAN (WGAN) and InfoGAN [54], [55]. The schematic of a GAN and some images of non-existent fake celebrities generated by a GAN are shown in 3.6.

Chapter 4

Methodology

This chapter contains a detailed description of the methodology used in this thesis. It can be divided into two parts. First, an analysis of the quark-gluon jet discrimination is presented along with corresponding machine learning models used for it. Also, we present the overview of the data used to train, validate and apply the models. Next, we present a novel method which calculates the underlying probability distributions of emissions in jet-like data which can potentially be used in measurements of the fragmentation functions.

4.1 Quark-gluon jet discrimination

The quark and gluon jet discriminators developed in this research are meant to be suited for the application at the ALICE experiment. In order to do this, a large amount of data describing jets is simulated to mimic data gathered by the ALICE detector. Then, this data is used to train, validate and apply proposed models.

4.1.1 The ALICE data

One of the goals of this thesis is to develop a method for identification and tagging of the light quark and gluon jets in the ALICE experiment. The method is based on a machine learning approach and uses the data obtained by the ALICE detector. Training of the models is done by using the data simulated by Monte Carlo (MC) generators. The data used is chosen to be from proton-proton collisions recorded in 2018 during the LHC run 2. The centre-of-mass energy of these collisions is $\sqrt{s} = 13$ TeV. Pythia 6 with a Perugia0 tune and GEANT3 were used to generate the events and simulate the detector response. The settings of the MC generator correspond to the state of the LHC collider and the ALICE detector at the time of the data taking. The jets from are clustered by

the anti-k_T algorithm with the E-recombination scheme and the parameter R set to 0.4. This is done by the FastJet package incorporated in the AliPhysics framework [].

Event and track selection

The ALICE detector is equipped with a number of triggers in order to reject unwanted events and reduce background signals. In this analysis, the selected events are required to satisfy several criteria. First, they must pass the minimum bias trigger. This trigger rejects background interactions such as beam-gas and beam-halo interactions. Also, in order to pass the minimum bias trigger, an event is required to have a signal in one of the two inner layers of the SPD detector or in one of the V0 detectors. Furthermore, the distance of the primary collision from the center of the detector along the beam line.

There are also several selection criteria for tracks in the selected events. First, to avoid too small a radius of particle trajectories in the detector's magnetic field, we require that the particle transverse momentum is greater than $0.15\,\mathrm{GeV}/c$. Also, the particles are required to have a pseudorapidity of $|\eta| < 1.0$ due to the limited detector's acceptance outside this region. Additionally, the trajectory of the particle in the TPC must be reconstructed with at least 70 measured points. The distance of the closest approach (DCA) of the track relative to the primary vertex must be no more than $0.018 + 0.035 p_T^{-1.01}$ċm, where p_T is particle's transverse momentum measured in GeV/c.

Jet reconstruction

The jets are clustered and reconstructed by the anti-kT algorithm as said above. The jets used in the analysis are required to have a total transverse momentum greater than $1\,\mathrm{GeV}/c$. Also, the number of jet constituents, i.e. jet multiplicity, is selected to be 10 or higher. The minimal jet area of the jets is set to be 0.001 in (y, ϕ) space. After applying selection criteria, the number of the obtained MC jets is $2.67 \cdot 10^7$. A jet as a data instance is saved as an array of size of 50×4. The values of the array represent each jet constituents four momentum. Size 50 is chosen since it is the maximum multiplicity of a jet in the dataset. Data represented in this way contain the maximum amount of information and can be easily suitably preprocessed for a model used in the analysis. Furthermore, jets are divided into bins according to their total transverse momentum, since the p_T-dependence of the quark and gluon fractions is of particular interest in

this thesis. Detailed specifics of the binning and properties of each of the subsamples are shown in table 4.1.

TABLE 4.1: Number of jets in each p_T bin.

p_T	N	N_q	N_g	N_{train}	N_{test}
0-25 GeV	5352459	509416 (9.5 %)	4843043 (90.5%)	4817213	1338114
25-50 GeV	5059608	872273 (17.2%)	4187335 (82.8%)	3794706	1264902
50-75 GeV	4774522	1011871 (21.2%)	3762651 (78.8%)	3580891	1193630
75-100 GeV	3951685	840789 (21.3%)	3110896 (78.7%)	2963763	987921
100-125 GeV	3051618	661499 (21.7%)	2390119 (78.3%)	2288713	762904
125-150 GeV	2080107	487988 (23.5%)	1592119 (76.5%)	1560080	520026
150-175 GeV	1246585	335412 (26.9%)	911173 (73.1%)	934938	311646
175-200 GeV	681821	217856 (32.0%)	463965 (68.0%)	511365	170455
200-250 GeV	536163	211876 (39.5%)	324287 (60.5%)	402122	134040

Jet labelling in MC

A parton flavour definition of the jets is needed to train the machine learning models. Monte Carlo generators give access to the additional information other than final state particles. In our analysis a particular jet is matched to the highest energy parton present within the radius ΔR used by the jet finding algorithm. This way, only a small fraction of jets remains unlabeled since jets with more than one high-energy partons in the shower are rare. Jets initiated by the heavier c-quarks and b-quarks are dismissed from the analysis. Bottom quark originated jets are defined by the requirement that at least one b-quark with $p_T > 5$ GeV is present in the parton shower. If there are no b-quarks and there is at least one c- quark with $p_T > 5$ GeV, these jets are labelled as charm jets.

4.1.2 Data preprocessing

There is a variety of ways and representations of data which can be used as an input to a machine learning model, namely a neural network. In the case of jets, there is a

number of possibilities one can consider. For example, global jet observables can be used. As an upside, this approach implies a low dimensionality of a model and the used observables can be designed specifically by using existing knowledge to extract the most important features. However, one is risking to omit information in the data which can be very important for the discriminating power. Examples of such observables are total jet momentum, jet mass, jet width, generalized angularities and many more. Another approach is to keep most of the features on the particle level. The downside of this approach is high dimensionality, but the information loss remains low. Also, one needs to be careful and recognize underlying symmetries in data. For example, if raw data (50×4 arrays of particles) is used as an input to a model, there exists a permutation symmetry since the ordering of the constituents physically does not change a jet. This can be partially addressed by employing some rule on the ordering. For example, arrays can be sorted according to the energy or momentum of a constituent. Another possibility is to use jet images, which resolves the problem of the ordering symmetry but introduces a spatial translational and rotational symmetry. Also, physical symmetries can be considered, such as Lorentz symmetry.

In this analysis three distinct representations are used. A list of particles' kinematic variables, a vector of generalized angularities and jet images. These representations are derived from the array of four-vectors with some kinematic transformations. The considered representations are chosen due to their generality and performance.

List of kinematic variables

As mentioned, data is stored as collections of jet constituent particle four-vectors (E, p_x, p_y, p_z). Since a neural network model implies inputs with a constant size, a tensor of size 50×4 is constructed. The first axis runs over the particles in a jet. The second axis runs over the components of the four-vector. The size 50 is chosen as it is the maximum jet multiplicity in the datasets. Most of the jets have a lower jet multiplicity, so the remaining part of the tensor is filled with zeros. This representation is used by the model pPFN described below. We also use four different kinematic variables to describe a particle, $(z, \Delta y, \Delta\phi, m)$, defined by:

$$z = \frac{p_T}{\sum_{i \in jet} p_{T,i}}, \tag{4.1}$$

$$\Delta y = y - y_{jet} = \frac{1}{2} \ln \frac{E + p_z}{E - p_z} - \frac{1}{2} \ln \frac{E_{jet} + p_{z,jet}}{E_{jet} - p_{z,jet}}, \tag{4.2}$$

$$\Delta\phi = \phi - \phi_{jet} = \arccos \frac{p_x p_{x,jet} + p_y p_{y,jet}}{p_T p_{T,jet}}, \tag{4.3}$$

$$m = \sqrt{E^2 - p_x^2 - p_y^2 - p_z^2}. \tag{4.4}$$

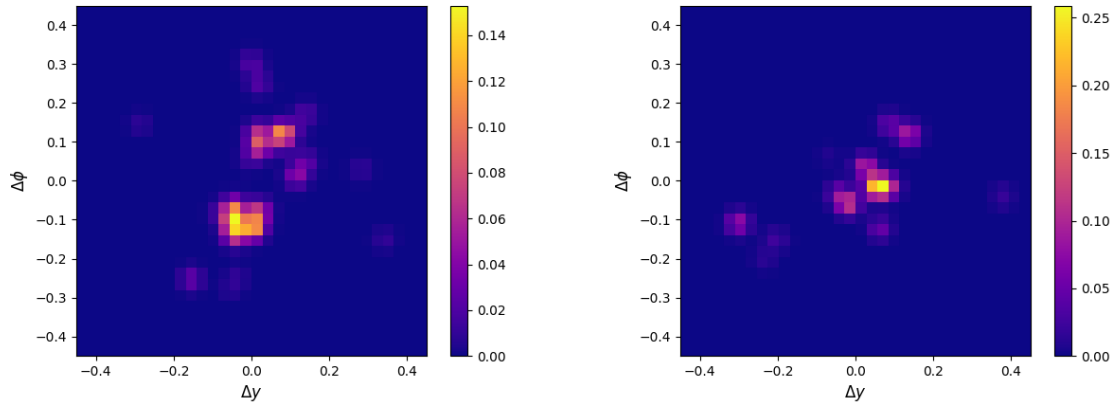
These variables are calculated from four-vectors for each particle. They represent the fraction of total transverse momentum, relative rapidity, relative azimuth and mass of the particle. Total four-vectors of a jet are calculated by adding together four-vectors of its constituents. Identically as the described above, a tensor with size 50×4 is constructed which contains variables $(z, \Delta y, \Delta\phi, m)$ for each particle.

Generalized angularities

Generalized angularities are calculated according to equations 1.8. They are easily retrieved using the already calculated $(z, \Delta y, \Delta\phi, m)$ representation of data. The angularities λ_β^κ, where $\kappa = 0, 1, 2$ and $\beta = 0, 1, 2$ are considered in the analysis. These combinations of indices give us a total of 9 angularities. This way, a jet is represented by a vector with the size of 9, filled with real values. Since we are dealing with the powers of variables that range from 0 to 1, the generalized angularities of higher orders are significantly smaller in value than the generalized angularities of lower orders. In order to prepare the data, a standard scaling is used. This means each variable is scaled so that its mean is zero and its standard deviation is one across the dataset.

Jet images

The usage of the jet images as a representation of the data is motivated by the performance of the CNNs in tasks of computer vision. The basic idea is to put each constituent particle on the right spot in the $(\Delta y, \Delta\phi)$ grid. Both of the variables are constrained in interval [-0.4, 0.4], since the parameter R of the anti-k_T algorithm is set to 0.4. These intervals are divided into 32 bins. This way, a $2D$ grid is constructed with $32 \times 32 = 1024$ bins. The central bin of the image represents the jet axis and the particles are usually found in the vicinity of that point. This way, a translational symmetry of the image is eliminated. If a particle has an η and a ϕ which belong to a particular bin, it is filled with the particle energy E in units of GeV. If a bin contains multiple particles, their energy is summed together. Jet images can also be constructed by other variables such as the total momentum of a particle, p, or a fraction of total transverse momentum, z. These different jet images can be stacked together so that



(A) (B)

FIGURE 4.1: Jet image examples in (z) representation.

the different channels represent variables E, p and z. In this analysis we use stacked (E, p) representations and standalone z representations. Constructed this way, jets are represented by tensors with sizes of $(32 \times 32 \times 2)$ and (32×32), respectively. The one arising problem of a such representation is sparsity. In our dataset the jet multiplicity does not exceed 50 which means that only a maximum of 50 bins out of 1024 will be populated. The rest of them will be zero. In order to overcome this, a Gaussian filter is used. The parameter σ (width) of the filter is set to be 3% of the height and width of the image. This parameter is adjustable and can be gradually decreased during the training process. Examples of jet images are shown in Figure 4.1. The model denoted by CNN uses (E, p) jet images and the model denoted by zCNN uses (z) jet images. Both models are discussed in detail in 4.1.4 below. Jet images (E, p) and (z) do not explicitly contain information of the particle identities. Additional versions of jet images are constructed in order to give some information about particle flavours. These images contain three channels. Fractions z of particles with $m < 0.3$ only are shown in the first channel. Particles with mass in the range [0.3, 0.6] are shown in the second channel and finally, $m > 0.6$ particles are show in the third channel. This representation is denoted as (z_π, z_K, z_p) since the first channel mostly consists of pions, the second channel consists mostly of kaons and the third channel consists mostly of protons.

4.1.3 Data balancing

An imbalance between quark and gluon initiated jets is present in the dataset as shown in Table 4.1. There are several methods to avoid this issue. For example, data instances can be weighted according the probability of its category. Another possibility is simply to cut down number of instances in the larger category so it is equal to the number of instances in the smaller category. In our case, this procedure would be problematic for jets in low p_T bins since there, the number of gluon labelled jets is much greater. As a consequence, the trained models will be either too simple or prone to an overfit. In this analysis the data is balanced by the requirement that the training process runs over the sample which contains the same number of quark labelled jets and gluon labelled jets. In each epoch, a subsample of the gluon labelled jets is randomly selected for the training dataset. The number of these gluon jets is equal to the total number of quark labelled jets in the corresponding p_T bin.

4.1.4 Discrimination models

In this section, different models used for the discrimination of quark and gluon initiated jets are discussed. Models that use jets represented by generalized angularities are the simple multivariate logistic regression model and the fully connected neural network. Models that use jet images are convolutional neural networks. Finally, kinematic properties of constituents are used by the ParticleFlow and EnergyFlow networks described below.

Fully connected network

A fully connected *feed-forward* network uses a vector of generalized angularities as an input. The architecture of the network consists of three hidden layers with 100 neurons each. The last layer has one neuron that is activated by the *sigmoid* function to constrain the network output in the interval between zero and one. Other layers are activated by the ReLU function. Such a network has approximately 20 000 parameters, depending how many generalized angularities are used as an input. A multivariate logistic regression is also used in an attempt to discriminate between jet flavours. For this, a neural network with only one neuron activated by the *sigmoid* function is used.

Such a model is clearly less capable than the neural network with multiple hidden layers, but offers a possibility of interpretation since the number of its parameters is low and equals the number of used generalized angularities.

Convolutional neural networks

Convolutional neural networks are employed to make use of jet images described in section 4.1.2. All of the used CNNs consist of 3 consecutive convolutional layers coupled with MaxPooling layers. Convolutional layers use a 3×3 kernel and have 32, 64 and 128 filters, in that order. They are regularized by the Dropout technique. MaxPooling layers use 2×2 pools. The output of the last MaxPooling layer is flattened and fed into a block of three fully connected layers with 100, 50 and 1 neuron, respectively. These layers use the ReLU activation function, except the last one which uses the *sigmoid* activation function. CNNs constructed this way contain approximately 300 000 parameters. There are three used CNN models which differ in type of jet images they use as an input. pCNN uses jet images with (E, p) channels and zCNN uses jet images with the z channel. Finally, zCNN-ID uses (z_π, z_K, z_p) jet images with 3 channels.

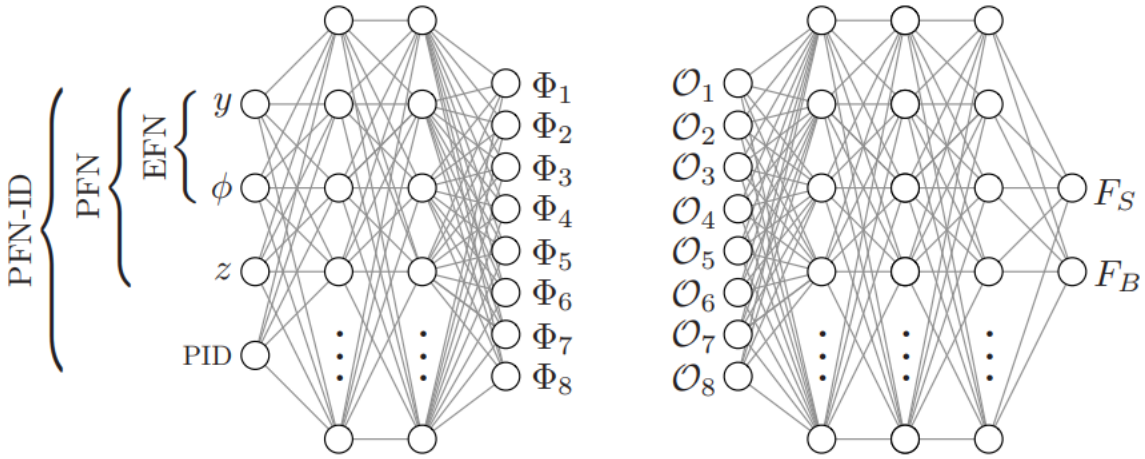


FIGURE 4.2: Architecture of the EnergyFlow and ParticleFlow networks introduced in [56].

4.1.5 EnergyFlow and ParticleFlow networks

EnergyFlow and ParticleFlow networks are introduced in [56]. The basic idea behind these architectures is the fact that any jet observable $\mathcal{O}$ can be approximated arbitrarily well by the following decomposition:

$$\mathcal{O}(p_1, ..., p_n) = F(\sum_i^n \Phi(\hat{p}_i)) \,, \tag{4.5}$$

where p_i is some particle level information of the i-th jet constituent, such as their four momentum. Also, one can use some other kinematic variables, such as fraction of the transverse momentum z, rapidity y and the azimuthal angle ϕ. Φ is particle level mapping of p_i into some observable and F is some continuous function. These mappings can be represented by networks. Such a model with two neural networks is called the ParticleFlow network (PFN). Infrared safety of such a decomposition can be enforced by

$$\mathcal{O}(p_1, ..., p_n) = F(\sum_i^n z_i \Phi(\hat{p}_i)) \,, \tag{4.6}$$

where $\hat{p}_i$ is usually (y_i, ϕ_i). Such a model would be called EnergyFlow network (EFN). In our analysis we used both models, the PFN and the EFN. Also, since the ALICE detector has a strong particle identification capability, particle flavour information is used in the PFN. The architecture of the neural networks Φ and F is based on the architecture used [56]. A schematic of the architecture is shown in 4.2. The neural network Φ consists of two layers with 100 hidden units. The size of its output vector (latent vector) is varied. The neural network F consists of three layers with 100 hidden units. The last layer of the network F outputs a scalar value with the *sigmoid* activation function. All of the other layers are activated by the ReLU activation function. In this analysis we used EFN as described, PFN with $(z, \Delta y, \Delta\phi)$ kinematic properties, PFN-ID with variables $(z, \Delta y, \Delta\phi, m)$, where m is the mass of the given particle and gives particle identification information. Lastly, pPFN use four-vectors (E, p_x, p_y, p_z) as kinematic representations of the particle.

4.1.6 The training process

The optimization of the described neural network is done by minimization of the *binary crossentropy* loss. This way, in fact, the log-likelihood is maximized. Samples in each p_T bin are divided into training and test datasets. The size of training dataset is

set to be 75% of the total dataset. This differs for jets in the first p_T bin (0-25 GeV) where the training sample consists of 90% of the total number of jets. As discussed in section 4.1.3, each epoch is run over the same number of quark and gluon labelled jets. All of the quark jets and the corresponding number of randomly selected gluon jets from the training dataset are used in each epoch. A good convergence for each model is obtained after 20 epochs, with CNNs being the exception, needing 40 epochs to reach satisfying convergence. Each neural network model is optimized by the ADAM algorithm.

4.1.7 Performance metrics

In this section some of the used performance metrics of the discriminating models are explained such as ROC and AUC. Also, permutation feature importance is used as a technique to interpret trained models.

Receiver Operating Characteristic (ROC)

One of the performance metrics used in the analysis of discriminating models is the ROC (Receiver Operating Characteristic) curve, defined with respect to a given class, which in our case that would be quark initiated jets. In this case, gluon initiated jets would be considered as the background. For a given jet j, we aim to find a model that outputs the probability that this jet belong to a class of quark initiated jets, $P(q|j)$. According to this probability, one can set a decision rule where we consider the jet a quark jet if a probability is greater than a certain threshold value, $P(q|j) > T$. Each threshold value generates a fraction of true positives (quark efficiency) and also a fraction of true negatives (gluon efficiency). For example, if $T = 0$, each jet is considered as a quark jet. In that case quark efficiency is equal to one since each quark jet is tagged correctly, but gluon efficiency is equal to zero since none of the gluon jets is recognized correctly. The ROC curve consists of the ensemble of (ϵ_q, ϵ_g) pairs generated by each possible threshold value T. This curve is in most cases a continuous function with start- and end-points being (0,1) and (1,0). The Area under this curve (AUC) is considered to be a good performance metric of a classifier, since the greater area under the curve, there are more points (ϵ_q, ϵ_g) with good quark and gluon reconstruction efficiency. The ROC curve can also be plotted by quark efficiency against gluon rejection (1 - ϵ_q). The ROC curve can help visualize the point where a model has very good background rejection with a small sacrifice in signal efficiency.

Feature importance

Feature importance measures how important a particular feature is for the predictive value of a model. It evaluates how the the prediction error increases when a feature is absent. This can be done by removing a particular feature and retraining the model. This procedure can be computationally exhausting. Also, the architecture of a model very often changes if the size ofda the input is decreased. To avoid these issues, we use permutation feature importance where retraining of the model is not necessary. In permutation feature importance one shuffles the values of a particular feature along samples in the, usually, test dataset. If a feature is important, this random reordering causes less accurate predictions. If the model does not rely on this feature, the predictions will have the same accuracy. Any scoring metric can be used in this procedure. In our analysis, we define a metric called PFI:

$$PFI = \frac{L - L_i}{L}\,, \tag{4.7}$$

where L is *binary crossentropy* loss averaged over the whole test dataset without any reordering of features. L_i represents the *binary crossentropy* loss averaged over the whole dataset if the values of the i-th feature are shuffled. This metric shows the relative feature importance. Defined this way, if the PFI equals zero, a feature does not have any importance to the performance of the model. As the PFI increases, a feature is more and more important for the performance of the model.

Shapley values

Another way to explain and interpret machine learning models is to calculate the Shapley values for each feature [57], [58]. The Shapley values are a game theory concept, but can be used in purpose of explaining a machine learning model, so the model represents a game and a specific feature represents a player of the game. These values show how much a feature contributes to a prediction in total. The prediction $f(\mathbf{x})$ of the model, where $\mathbf{x}$ is an input feature vector, can be linearly decomposed as follows:

$$f(\mathbf{x}) = \phi_0 + \sum_{i}^{M} \phi_i(\mathbf{x})\,, \tag{4.8}$$

where $\phi_i(\mathbf{x})$ denotes the Shapley value of i-th feature for the input $\mathbf{x}$ in the total number of M features. The value of ϕ_0 is the averaged mean of the predictions over the whole

phase space of the feature vector **x**. The global averaged importance of a feature can be obtained by taking the mean of absolute Shapley values over the whole dataset. In practice, in order to calculate Shapley values, the computational time grows exponentially with the total number of features and only a subset of the test dataset is used, usually 1 000 – 5 000 data instances. In this research Shapley values are calculated using the SHAP Python module (SHapley Additive exPlanations).

4.2 The *2NN* algorithm

In this section we present a method which iteratively recovers the underlying probabilities of emissions in some hypothetical physical system where the particles only decay into two daughter particles. The idea is to design a method with a high level of generalization, so that in be applicable in a particular real physical system in some degree. Two basic assumptions used in the method are the conservation of four-momenta and relativistic invariance. Such a method could possibly give an insight to radiation patterns in QCD jets. We present the 2NN method introduced in [59] which is a generalization and a continuation of the method presented in [60]. Also, further generalizations and improvements of these methods are suggested.

4.2.1 The physical system

In particle physics, jets are detected as collimated streams of particles. The jet production mechanism is in essence clear: partons from the initial hard process undergo the fragmentation and hadronization processes. In this work, we develop a simplified physical model in which the fragmentation process is modeled as cascaded $1 \to 2$ independent decays of partons with a constant number of decays. We represent each decay of a mother parton of mass M by four real numbers $(\frac{m_1}{M}, \frac{m_2}{M}, \theta, \phi)$, where m_1 and m_2 are the masses of the daughter particles, and θ and ϕ are the polar and azimuthal angle of the lighter particle, as measured from the rest frame of the mother particle. For simplicity we make all the decays isotropic, which is not necessarily true in real processes. Using conservation laws, the energies and the momenta of the daughter particles for a given mother particle's mass M can be calculated in the rest frame of the mother particle in the following way:

$$E_1 = \frac{1}{2M}(M^2 + m_1^2 - m_2^2)\,, \quad E_2 = \frac{1}{2M}(M^2 + m_2^2 - m_1^2)\,, \tag{4.9}$$

$$p_1 = \sqrt{E_1^2 - m_1^2}\,, \quad p_2 = \sqrt{E_2^2 - m_2^2}\,, \tag{4.10}$$

$$\begin{aligned} p_{1x} &= -p_{2x} = p_1 \sin\theta \cos\phi\,, \\ p_{1y} &= -p_{2y} = p_1 \sin\theta \sin\phi\,, \\ p_{1z} &= -p_{2z} = p_1 \cos\theta\,. \end{aligned} \tag{4.11}$$

The four-momenta of the daughter particles in the laboratory frame are obtained by performing a Lorentz transformation from the rest frame of the mother particle.

We note that this setup also covers the case in which a particle does not decay by setting m_1 equal to zero and $m_2 = M$. This produces a daughter particle with zero energy and a daughter particle with the same four-momentum as the mother particle. Physically, this corresponds to a non-decayed mother particle. Furthermore, we observe that for any pair of daughter masses in which $m_1 + m_2 < M$, some mass has converted to energy. The calculations described here are performed for each decay. This way, in each step we obtain the daughter particles' four-momenta and use them as mother particles' four-momenta in the next step. When this procedure is repeated N times, after $2^N - 1$ decays, we obtain a jet with 2^N particles in the final state (some of which may have mass zero). Assuming that the initial particle's properties are fixed, this means any single jet is described by $4 \times (2^N - 1)$ parameters. We call these parameters the degrees of freedom of a jet, and can sample them from some probability distribution.

To fully define our physical system we set a decay probability distribution function $p(m_1, m_2|M)$, the details of which are given in the following subsection. The aim of our proposed algorithm is to recover these underlying probability distributions, assuming we have no information on them, using only a dataset consisting of jets described with final particles' four-momenta, as one would get from a detector.

4.2.2 The particle Generator

To generate the jets, we developed an algorithm where we take a particle of known mass that undergoes three successive decays. We consider only the possibility of discrete decays, in the sense that the decay product masses and decay probabilities are well defined. We consider a total of 10 types of particles, labeled A–J, which can only decay into each other. The masses and the decay probabilities of these particles are given in Table 4.2. In this scenario, the „decay probabilities" p are given by the ratios

TABLE 4.2: Allowed particle decays in the discrete model. The designation p/channel shows the probability that a mother particle will decay into specific daughters.

Particle	A	B	C	D	E
Mass	0.1	0.6	1.3	1.9	4.4
p/channel	1 A	0.7 B 0.3 A+A	1 C	0.4 D 0.3 A+A 0.3 A+C	0.6 C+C 0.4 E
Particle	F	G	H	I	J
Mass	6.1	8.4	14.2	18.1	25
p/channel	0.5 A+A 0.5 B+C	0.9 B+B 0.1 A+F	0.6 D+D 0.25 D+E 0.15 E+F	1 F+G	0.5 F+I 0.4 G+H 0.1 E+E

of decay amplitudes. Thus, the total sum of the probabilities for a given particle to decay into others has to be one, and the probabilities describe the number of produced daughters per N decays, scaled by $1/N$.

Particles A–E are set to be long lived and can thus be detected in the detector, which only sees the decay products after several decays. This can be seen in Table 4.2 as a probability for a particle to decay into itself. In this way, we assure two things: first, that we have stable particles and second, that each decay in the binary tree is recorded, even if it is represented by a particle decaying into itself. Particles A and C are completely stable, as they only have one „decay" channel, in which they decay back into themselves. On the other hand, particles F–I are hidden resonances: if one of them appears in the i-th step of the decay chain, it will surely decay into other particles in the next, (i+1)-th step of the chain.

To create a jet, we start with particle J, which we call the mother particle, and allow it to decay in one of the decay channels. Each of the daughter particles then decays according to their decay channels, and this procedure repeats a total of 3 times. In the end, we obtain a maximum of 8 particles from the set A–E, with known momenta measured from the rest frame of the mother particle.

4.2.3 Mathematical background of the method

The starting point of the method is the Neymann-Paerson lemma which states that the likelihood ratio is the most powerful hypothesis test between two random variables at a given significance level [61]–[63]. In other words, the optimal discriminator of the

two random variables would be their likelihood ratio. Let us assume there is a dataset of real numbers x. These values follow some unknown probability density $p(x)$. This dataset is denoted as the real dataset. If a test dataset is constructed so that it contains values x according to probability density $q(x)$, the Neymann-Pearson lemma states that the best discriminator between the two datasets is the following ratio:

$$\Lambda(x) = \frac{p_{\text{real}}(x)}{p_{\text{test}}(x)}\,. \tag{4.12}$$

Assuming one is equipped with an optimal classifier and p_{test} is known, this enables determination of the probability p_{real} for a given x. In practice, machine learning models and other techniques used in classification tasks aim to output probability of the category that the given input x will be part of. In the case of a binary classification between the real and the test dataset this can be written as conditional probability density $p(\text{real}|x)$. This quantity can be calculated from the likelihood ratio by employing the Bayes' theorem

$$p(\text{real}|x) = \frac{p(x|\text{real})p(\text{real})}{p(x|\text{real})p(\text{real}) + p(x|\text{test})p(\text{test})}\,. \tag{4.13}$$

If the datasets are of the same size, $p(\text{real})$ and $p(\text{test})$ are both equal to $1/2$. Quantities $p(x|\text{real})$ and $p(x|\text{test})$ are probabilities $p_{\text{real}}(x)$ and $p_{\text{test}}(x)$ that the given value x will appear in the real and the test dataset, respectively. Combining of the expressions 4.12 and 4.13 results in the following expression:

$$p_{\text{real}}(x) = \frac{p(\text{real}|x)}{1 - p(\text{real}|x)} p_{\text{test}}(x)\,. \tag{4.14}$$

In order to simplify equations, values $p(\text{real}|x)$, $p_{\text{real}}(x)$ and $p_{\text{test}}(x)$ will be denoted as $C_{NN}(x)$, $p(x)$ and $q(x)$ in future considerations. In ideal conditions, the unknown probability density $p(x)$ can be calculated with the help of the known probability density $q(x)$ and a classifier. In practice, one does not deal with an optimal classifier. In further text we present a proof that the iterative application of sub-optimal classifiers indeed converges to the recovery of the real probability density $p(x)$, assuming classifiers are neural networks trained by minimization of a binary crossentropy loss. It can be assumed that the averaged binary crossentropy loss will be higher than the loss of the optimal classifier loss and lower than the value of $\ln 2$, being the averaged loss of the random classifier. These inequalities can be written as follows:

$$-\frac{1}{n}\sum_{i\in\text{real}}\ln C_{NN}(x) < -\frac{1}{n}\sum_{i\in\text{real}}\ln C'_{NN}(x) < \ln 2 \tag{4.15}$$

$$-\frac{1}{n}\sum_{i\in\text{test}}\ln\left(1-C_{NN}(x)\right) < -\frac{1}{n}\sum_{i\in\text{test}}\ln\left(1-C'_{NN}(x)\right) < \ln 2\,, \tag{4.16}$$

where C_{NN} denotes an optimal and C'_{NN} a sub-optimal classifier. Both inequalities can be expanded and simplified in the following way:

$$\sum_{i\in\text{real}}\ln\frac{C_{NN}(x)}{1-C_{NN}(x))} > \sum_{i\in\text{real}}\ln\frac{C'_{NN}(x)}{1-C'_{NN}(x))} > 0 \tag{4.17}$$

$$\sum_{i\in\text{test}}\ln\frac{1-C_{NN}(x))}{C_{NN}(x)} > \sum_{i\in\text{test}}\ln\frac{1-C'_{NN}(x))}{C'_{NN}(x)} > 0 \tag{4.18}$$

When the sub-optimal classifier is applied in 4.14 it results in some probability density $p'(x)$ which is not the real probability density $p(x)$. The inequalities written in terms of these probability densities become:

$$\sum_{i\in\text{real}}\ln p(x) > \sum_{i\in\text{real}}\ln p'(x) > \sum_{i\in\text{test}}\ln q(x) \tag{4.19}$$

$$\sum_{i\in\text{test}} p(x) < \sum_{i\in\text{test}} p'(x) < \sum_{i\in\text{test}}\ln q(x)\,. \tag{4.20}$$

The obtained result tells us, although the calculated $p'(x)$ is not the desired probability $p(x)$, it is on average closer to the desired result than the probability density $q(x)$. As a consequence, the same procedure can be repeated using the probability density $p'(x)$ instead of $q(x)$. This process can continue iteratively until the satisfactory convergence to the real probability density $p(x)$ occurs.

Generalization to Jets

The 2NN algorithm aims to recover underlying probabilities of particle emissions which occur in a cascading manner. Such a process generates a final state that is detected in an experiment. Since only the final states are detected in particle accelerators, the densities have to be retrieved only from the final states. In order to apply the 2NN algorithm, the data probability of the final state must be modeled somehow. We represent jets as a binary tree of depth N which consists of $2^N - 1$ independent decays producing a

maximum of 2^N particles in the final state. This way, a jet is described by a vector $\vec{x}$ with size $4 \times (2^N - 1)$. Then, the probability distribution function given by:

$$p\left(\vec{x}\right) = \prod_{i}^{2^N-1} p(m_1^i, m_2^i|M) p_\theta(\theta^i) p_\phi(\phi^i), \tag{4.21}$$

where i denotes the decay index and $(m_1^i,\ m_2^i,\ \theta^i,\ \phi^i)$ are the components of the vector $\vec{x}$ and $p_\theta(\theta^i)$ and $p_\phi(\phi^i)$ are the angle probability distributions. We assume that both angles are uniformly spatially distributed. Therefore, they contribute to the probability distribution with a constant factor. After plugging in $p\left(\vec{x}\right)$ from Eq.4.21 into Eq.4.14, the angles can be omitted, since the constant factors cancel each other out:

$$\prod_{i}^{2^N-1} p(m_1^i, m_2^i|M) = \frac{C_{NN}(\vec{x})}{1 - C_{NN}(\vec{x})} \prod_{i}^{2^N-1} q(m_1^i, m_2^i|M). \tag{4.22}$$

$$\sum_{i}^{2^N-1} \ln p(m_1^i, m_2^i|M) = \ln C_{NN}(\vec{x}) - \ln(1 - C_{NN}(\vec{x})) + \sum_{i}^{2^N-1} \ln q(m_1^i, m_2^i|M). \tag{4.23}$$

This linear system of equations has to be solved to obtain the probability $p(m_1, m_2 \mid M)$. This is a very computationally exhausting task due to the size of the dataset. To avoid this issue, a neural network f is introduced to approximate $\ln p(m_1, m_2|M)$, which can then be optimized by minimizing the *mean squared error* between the two sides of Eq.4.23

4.2.4 The workflow of the 2NN algorithm

The analysis done in the above section is used to design the steps of the 2NN algorithm. The workflow of the algorithm is constructed as follows: firstly, the parameters of both neural networks are initialized. Then, a test dataset is generated using the probability distributions according to the neural network f. In the next step, the classifier network is trained to distinguish between jets from the test dataset and the jets from the real dataset. This produces a set of linear equations in the form of Eq.4.23. These are not solved, but their solutions are approximated by fitting the neural network f, which in turn produces a new test dataset. This procedure is then repeated iteratively. The algorithm stops when there are no noticeable changes in the difference in the real and

the test distributions. The individual steps of the algorithm workflow are described in the next subsections.

Generating the test dataset

After the parameters of the neural network f are initialized, a test dataset of jets is generated with known decay probabilities $q(\vec{x})$. The input of the neural network f is a vector consisting of 3 real numbers: $a = m_1/M$, $b = m_2/M$ and M. We denote the output of the neural network with $f(a, b, M)$. Due to conservation laws, the sum $a + b$ needs to be strictly less or equal to 1. We can assume $a \leq b$ without any loss of generality. In order to manipulate with probabilities a partition function, defined by

$$Z(M) = \int_{\Omega} e^{f(a,b,M)} \, \mathrm{d}a\mathrm{d}b \tag{4.24}$$

needs to be calculated first. Here, Ω denotes the entire probability space and is shown as the gray area in the left panel of Fig. 4.3. To calculate the integral in the above expression, the probability space is discretized into 650 equal areas shown in the right panel of Fig.4.3. These areas are obtained by discretizing the parameters a and b into equidistant segments of length 0.02. After the discretization, the partition function $Z(M)$ then becomes equal to

$$Z(M) \approx \sum_j \sum_k e^{f(a_j,b_k,M)} \, . \tag{4.25}$$

To generate the jets which form the test dataset, we must generate each decay in the cascading evolution using the neural network f. Each of the decays is generated by picking a particular pair of parameters (a, b) from the 650 possible pairs which form the probability space for a given mass M. The decay probability is then given by:

$$q(m_1, m_2 \mid M) = \frac{e^{f(a,b,M)}}{Z(M)} \, . \tag{4.26}$$

This probability is used to generate a single decay of a given mother particle of mass M. Each decay is generated by randomly sampling a pair of parameters (a, b) according to the calculated probability (Eq. 4.26) from 650 possible pairs which form the probability space. The kinematical parameters of the decay are calculated according to Eqs. 4.9–4.11 and setting $m_1 = aM$ and $m_2 = bM$, to obtain the four momenta of the

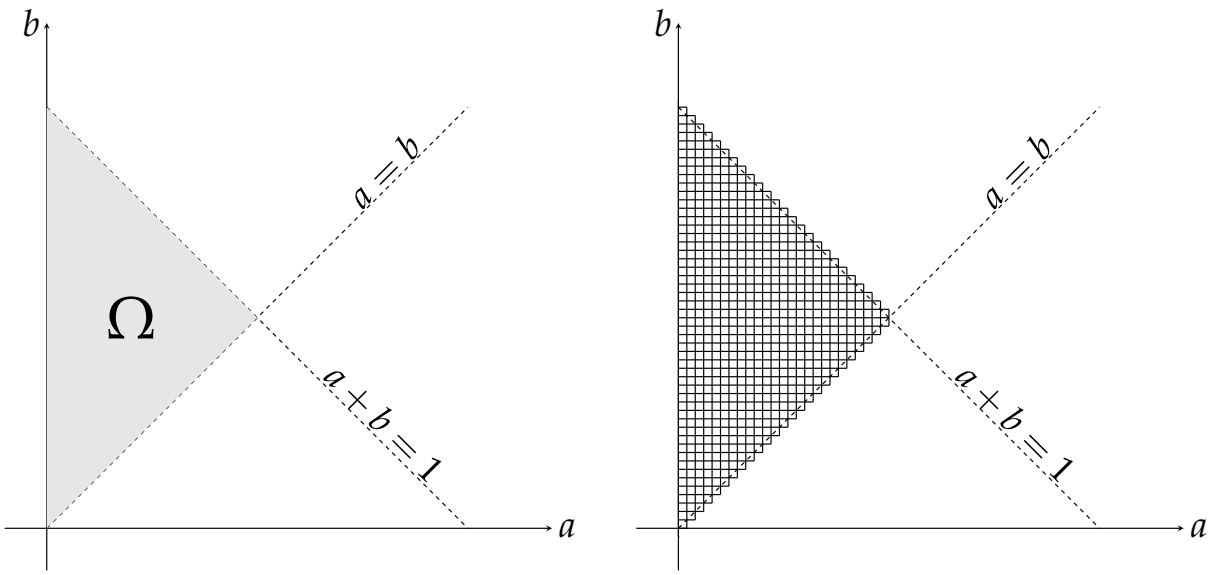


FIGURE 4.3: (**a**) The left panel shows the entire allowed probability space of the parameters a and b, designated by Ω. Due to conservation of mass-energy, $a + b \leq 1$ needs to hold true. To describe our system, we selected the case where $a \leq b$, which we can do without loss of any generality. (**b**) The right panel shows the discretized space Ω, as used to evaluate the partition function.

daughter particles. To produce a full event, i.e. a jet, this procedure is done iteratively N times. The four-momentum of the initial parton in it's rest frame is sampled from the distribution of the total jet mass in the „real" dataset which is, in turn, obtained from the total jet mass histogram. It must be noted that the jets in the „real" dataset have to be preprocessed by applying suitable Lorentz transformations, so that the total momentum of each jet in it's rest frame equals zero. In the presented physical system the mass of the initial parton takes only on one possible value, equal to $M = 25$ (in dimensionless units).

After applying this procedure a test dataset is obtained in which each jet is represented as a list of 2^N particles and their four-momenta. For each decay, we also store the pairs (a^i, b^i) as well the corresponding decay probabilities. For each jet in the test dataset we store $2^N - 1$ triplets (a, b, M) and their corresponding probabilities calculated by Eq. 4.26 for each decay in the jet. Since all the decays are independent, the total probability of a jet appearing in the test dataset is a direct product of the probabilities for each particular decay. The value of N is a parameter of the algorithm which needs to be guessed, since we can not foresee how many consecutive decays are needed to

produce a specific real dataset. Since our kinematic setup allows non-decays, the value of N has to be larger or equal to the degree of the longest chain of consecutive decays in a physical system.

Classifier optimization

The classifier used in this work is a convolutional neural network. The input to these types of network are typically sets of images. For this purpose, all the jets are preprocessed by transforming the list of particles' four-momenta into jet images. This way, 32×32 images are produces for a single jet. In the images, the axes correspond to the decay angles θ and ϕ, while the pixel values are either the energy or the momentum of the particle found in that particular pixel. If a pixel contains two or more particles, their energy and momenta are summed together and stored as pixel values. The transformation of jet representations is done on both the real and the test datasets. We label the „real" jet images with the digit 1 and the „test" jet images with the digit 0. The classifier is then optimized by minimizing the binary crossentropy loss between the real and the test datasets. The optimization is performed by the ADAM algorithm.

Optimization of the neural network f

After the classifier is optimized, a new jet dataset is generated by using the neural network f as described in 4.2.4. Just as earlier, the generated jets are first transformed into jet images and then fed to the classifier. Since we have access to each of the decay probabilities for each jet, the right side of Eq.4.23 can be easily calculated for all the jet vectors $\vec{x}$ in the dataset. This way we can obtain the desired log value of the total probability for each jet $p(\vec{x})$:

$$\ln p(\vec{x}) = \ln C_{NN}(\vec{x}) - \ln(1 - C_{NN}(\vec{x})) + \sum_{i}^{2^N-1} \ln q(m_1^i, m_2^i|M). \tag{4.27}$$

The parameters of the neural network f are then updated by minimizing the expression given by:

$$L = \frac{1}{n}\sum_{i}^{n}\left[\sum_{j}^{2^N-1} f(a_i^j, b_i^j, M_j) - \ln p_i(\vec{x})\right]^2, \tag{4.28}$$

where i denotes the jet index and j denotes the decay index in a particular jet. In this purpose we introduce a model which takes a sequence of $2^N - 1$ triplets (a^j, b^j, M_j) as

an input, and simply gives the sum of neural network f outputs for each of the triplets. We store all the triplets when generating a dataset. After this step, the weights of the neural network are updated in such a way that the network output values $f(a, b, M)$ are on average closer to the real log value of $p(m_1, m_2 \mid M)$. The updated network f is then used to generate the test dataset used as input for the next iteration.

4.2.5 Evaluation of the 2NN algorithm

Upon completion of each iteration of the algorithm, the underlying probability densities can be obtained from the output values of the neural network f according to Eq.4.26. In the Results section, the 2NN algorithm is evaluated in terms of the Kullback-Leibler divergence (KL) in the following way:

$$KL(M) = \sum_{j,k} p_{\text{real}}(m_1^j, m_2^k \mid M) \left[\ln p_{\text{real}}(m_1^j, m_2^k \mid M) - f(a^j, b^k, M) + \ln Z(M) \right] \quad (4.29)$$

where the sum is performed over the whole probability space. The KL-divergence is a non-negative measure of the difference between two probability densities defined on the same probability space. If the probability densities are identical, the KL divergence is zero.

4.3 Hardware and software

The code used for calculations in this research is written in the Python programming language using *Tensorflow 2* and *Numpy* modules. An NVIDIA Quadro p6000 GPU Unit obtained from the NVIDIA Grant for academic research was used to increase the speed of the performed calculations.

Chapter 5

Results

The results presented in this chapter contain evaluations of various neural network models used for the discrimination of the quark and gluon initiated jets. Each model is separately trained on jets from the different p_T bins. Separability of quark and gluon samples for different bins is analyzed. Interpretability and possible applications of the trained models are discussed. Furthermore, the $2NN$ algorithm is applied on the data simulated by the particle generator described in 4.2.2. Performance evaluation of the algorithm is analyzed and presented. The obtained probability distributions are compared to the expected values. Also, the convergence of the algorithm is analyzed and discussed.

5.1 The Quark-gluon model performance

The performance of our models is evaluated by means of ROC curves and, more importantly, AUC metrics evaluated over the test dataset. The discriminating power of the models is compared to known discriminating variables such as jet multiplicity and jet mass. The model evaluations are divided into three parts, ANG model evaluation, ParticleFLow models evaluation (EFN, PFN, PFN-ID, pPFN) and convolutional models evaluation (CNN, zCNN, zCNN-ID).

5.1.1 The ANG model

In this section we evaluate the ANG model and the logistic regression using generalized angularities. Fig. 5.1 shows the ROC curves for the ANG model evaluated for different p_T bins. It can be qualitatively seen that the model performance increases with increasing p_T. For jets in the lowest p_T bin, the model shows only a small improvement relative to a random classifier. The ANG model trained on jets that belong

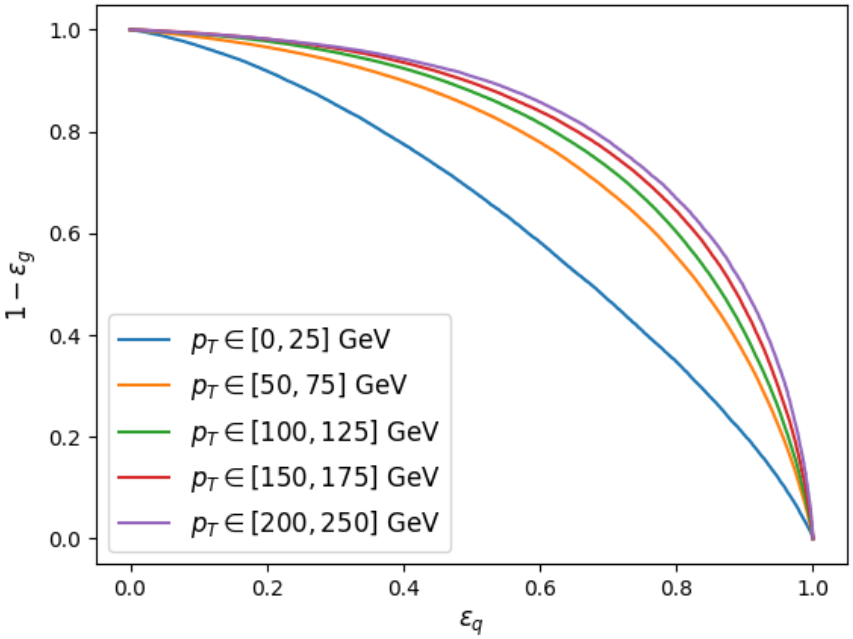


FIGURE 5.1: The ANG model ROC curves for the different p_T bins.

to the highest p_T bin shows much better discriminating power. To quantify the performance, the area under the ROC curve is calculated (AUC). Fig. 5.2 shows the p_T dependence of the AUC metric for the ANG model and the logistic regression of generalized angularities in comparison to AUC metrics of jet multiplicity and jet mass. The logistic regression shows an improvement relative to jet multiplicity and jet mass. This improvement is expected since jet multiplicity is explicitly located in the vector of the generalized angularities and the jet mass closely relates to the variable λ_1^2.

The ANG model shows even greater improvement as a neural network with a couple of hidden layers which can capture non-linear correlations between different generalized angularities. Also, one can clearly see that the AUC value increases monotonically with p_T. Using logistic regression, weights assigned to each generalized angularity can be extracted. These weights roughly give an information of how much a particular angularity contributes to the prediction. These weights can be seen in Fig 5.3 for the lowest and the highest p_T bin. In the lowest bin (0-25 GeV) the generalized angularity $(\kappa, \beta) = (1, 1)$ contributes the most. This variable is known as jet width or jet girth. In the highest p_T bin the jet multiplicity is the most important along with significant contributions from jet width and generalized angularity $(\kappa, \beta) = (2, 0)$. The

ANG model is a neural network with approximately 20 000 parameters and it acts like a black box. One cannot extract meaningful information from its weights. In order to do analysis of how a specific variable affects its predictions we use PFI metrics defined in 4.7. In Fig. 5.4 a PFI metric can be seen for jets in the lowest and highest p_T bins. In the lowest p_T bin, similar feature importance is observed as in the logistic regression model. However, in the highest p_T bin some differences occur. The most contributing variable is still jet width, but one also can see that a lot of other variables are important as well. The ANG model captures these and performs significantly better than simple logistic regression on jets with high p_T. At lower jet p_T values it performs similarly since there is additional information available, i.e. quark initiated and gluon initiated jets are very similar in when represented with generalized angularities.

In addition to the PFI metric, the Shapley values of the features were calculated. The mean absolutes of the Shapley values are qualitatively in agreement with the calculated PFI metric. The jet width is found to be the most important feature overall. Since a particular prediction of the ANG model can be decomposed as the sum of individual Shapley values according to 4.8, we are able to plot the Shapley values of the

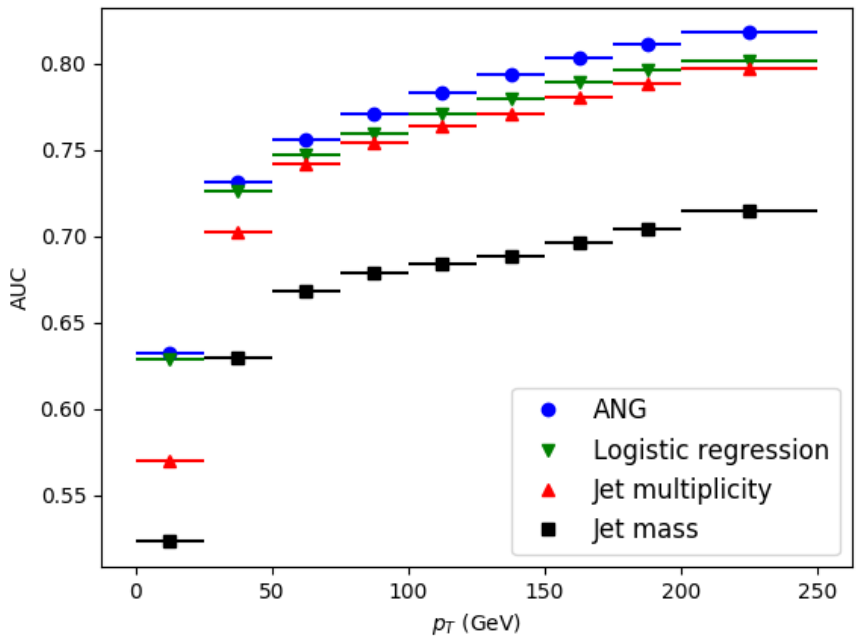
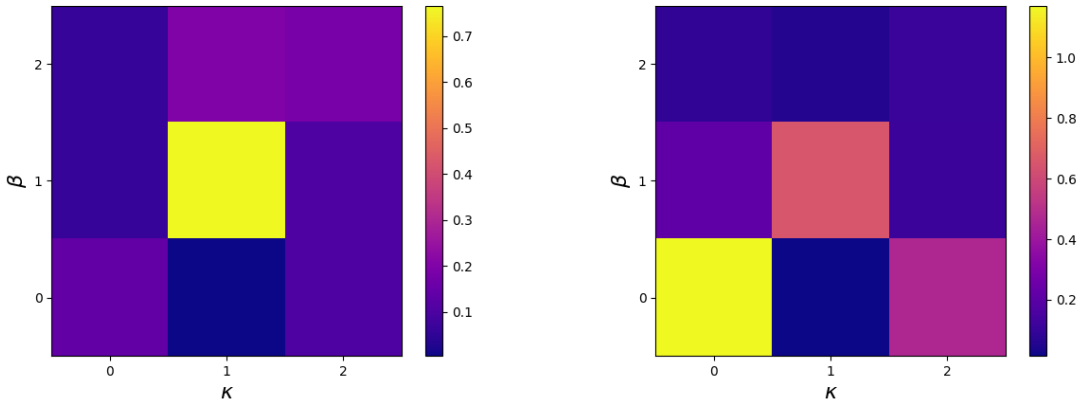
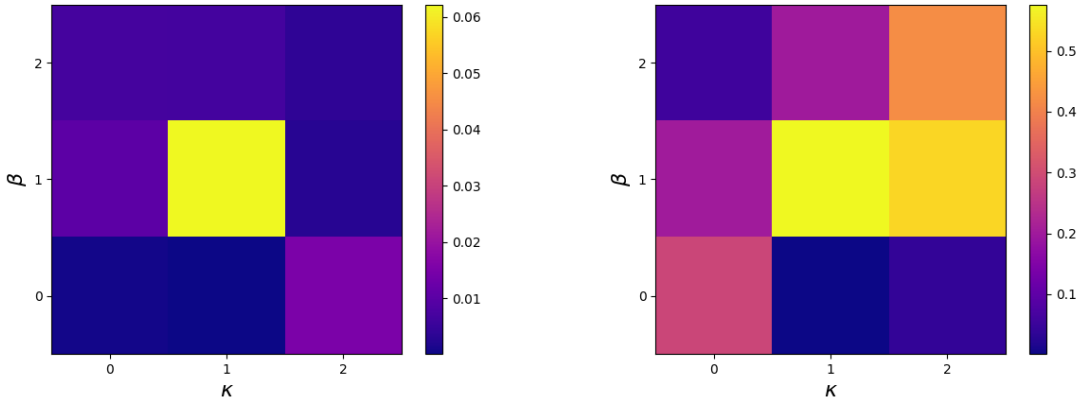


FIGURE 5.2: p_T dependence of AUC metrics for various convolutional neural network models and the ANG model.



(A) $p_T \in [0 - 25\,\text{GeV}]$ (B) $p_T \in [200 - 250\,\text{GeV}]$

FIGURE 5.3: Logistic regression weights assigned to generalized angularities (κ, β) for jets in the lowest and the highest p_T bins.

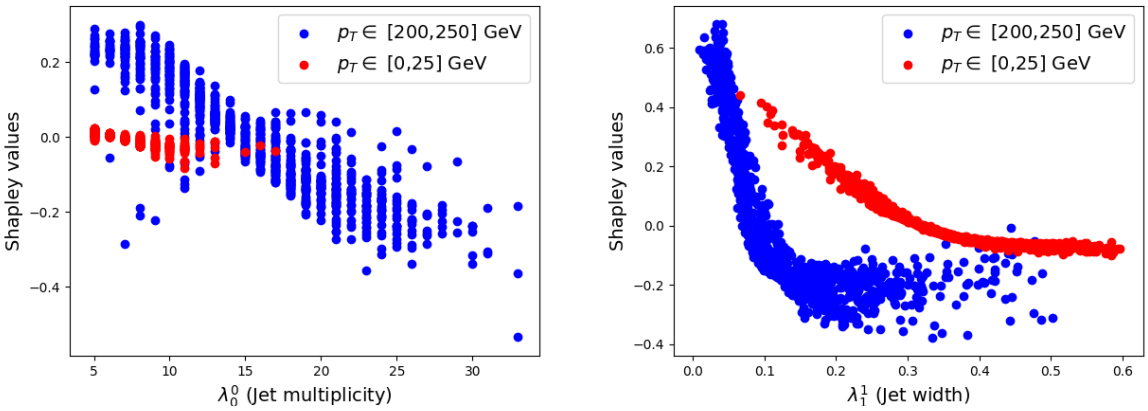


(A) $p_T \in [0 - 25\,\text{GeV}]$ (B) $p_T \in [200 - 250\,\text{GeV}]$

FIGURE 5.4: PFI values of generalized angularities (κ, β) used in the ANG model for jets in the lowest and the highest p_T bins.

specific feature against the feature value. In Fig. 5.5 we can see such scatter plots for jet multiplicity and jet width variables according to ANG models in the lowest and the highest p_T bins. The Shapley values are evaluated on 1 000 data instances. Clearly, we can see a higher contribution from the jet width variable than the jet multiplicity variable, whose contribution in the lowest p_T bin is almost negligible. The jet width shows a significant contribution in the lowest p_T bin. A continuous trend is found for both variables, such that their contributions to the prediction become more and more

negative as their values increase. This means that higher values of these variables imply that a jet is more probable to be a gluon jet. Alternatively, the jets are more likely to be quark jets if the jet width and the jet multiplicity are low in value. We can safely conclude that the jet width variable is the most important for the discriminating power of the ANG model.



(A) Jet multiplicity (B) Jet width

FIGURE 5.5: Shapley values for jet multiplicity and jet width.

5.1.2 Convolutional models

In this section various convolutional neural networks, namely the CNN, zCNN and zCNN-ID models, are evaluated. Fig. 5.6 shows the p_T dependence of the AUC metric for the mentioned models in comparison with the ANG model. The CNN model which uses (E, p) channels of jet images has the worst performance of the three. The other two models have a similar performance but PFN-ID exhibits slightly better performance across the entire range of jet p_T-s. These results give us two conclusions. The standalone fraction of jet total transverse momentum z proves to be a very suitable variable. An additional conclusion is that information of the particle identity is important and can further improve the performance of a model. In comparison with the ANG model, the CNN model exhibits similar performance. zCNN and zCNN-ID perform better than the ANG across the entire p_T range with greater differences being bigger at high p_T.

Similarly to the ANG model, we performed a feature importance analysis for jet images. Different pixels in the $(\Delta\phi, \Delta y)$ grid are considered to be the features of the

jet images. This way, a PFI value of a specific pixel is obtained. In Fig. 5.7 PFI values of different pixels are shown for jets in the lowest and the highest p_T bins. One must be careful when drawing strong conclusions from the PFI in high-dimensional data, like jet images, since there can be a lot of correlated pixels and omitting one important pixel may not alter the PFI score. However, one can gain some insight on what is happening. In this case, one can observe that the model is much more sensitive to the pixels in the close proximity to the center, i.e. to the jet axis. For jets with the lowest p_T, the sensitive area is much larger than the sensitive area around the jet axis of the model trained on high p_T jets. This is in an agreement with the observed jet width importance in the ANG model where jet width is more important relative to other generalized angularities at low p_T. At higher p_T, jets seem to be more collimated and jet width is not as important relative to other observables. In that case, the model needs to find additional information to successfully distinguish between the two categories.

As is the case with the ANG model, we calculated the Shapley values for the pixels

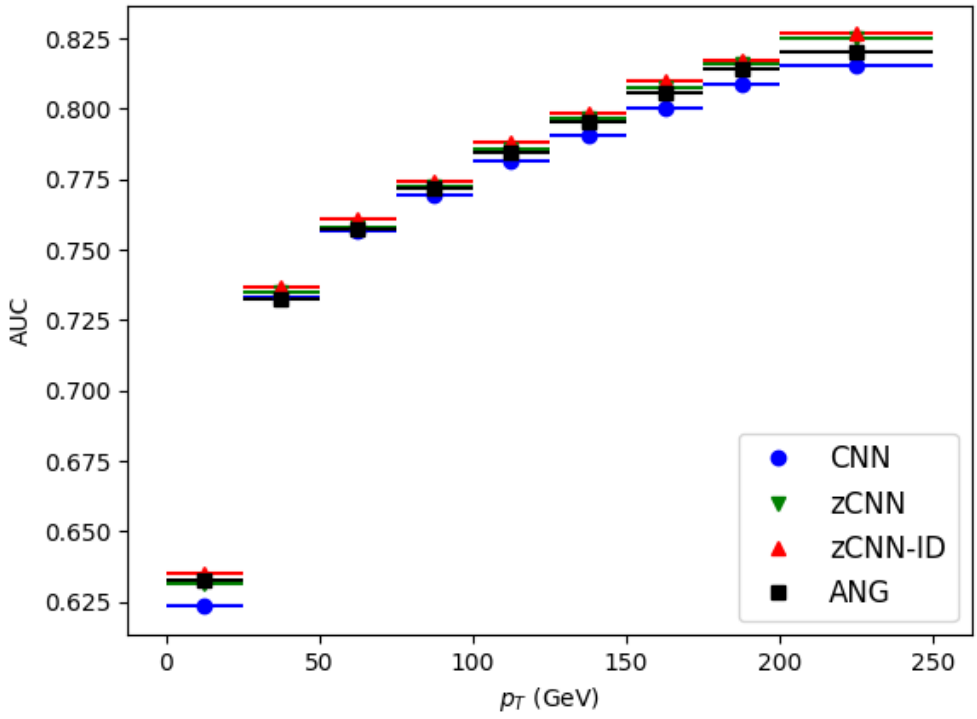
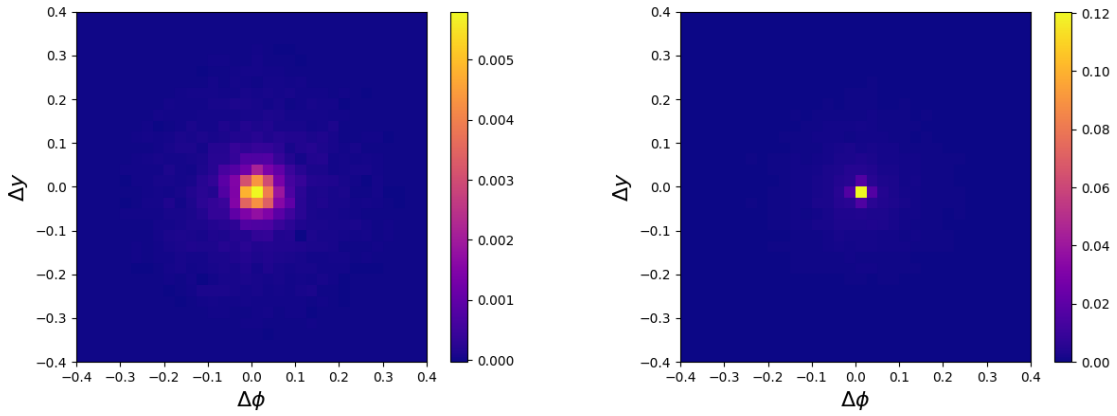


FIGURE 5.6: The p_T dependence of AUC metrics for various convolutional neural network models.



(A) $p_T \in [0 - 25\ \text{GeV}]$ (B) $p_T \in [200 - 250\ \text{GeV}]$

FIGURE 5.7: PFI values of different pixels in jet images used by the zCNN model for jets in the lowest and the highest p_T bins.

in the jet image. The results for all the pixels are presented together in 5.8. In the highest p_T bin there are two distinct groups of points. The lower group corresponds to the most central group of pixels, a 2×2 square in the center of the image. This result shows that if there is a particle with $z > 0.6$, a jet is more likely to be a quark jet. For lower values of z we can see a small negative contribution and the model is trying to find additional information in the wider areas. The higher of the two groups corresponds to the all remaining pixels. The Shapley values of all the pixels, according to zCNN trained on the jets from the lowest p_T bin, follow a similar distribution. This could be explained by the fact that jets which do not contain a particle with a high momentum fraction are more similar to low-energy jets. These results are also in agreement with the conclusion that jet width is an important discriminative variable.

5.1.3 ParticleFlow models

In this section EFN, PFN, PFN-ID and pPFN models are evaluated. The p_T dependence of the AUC metric for these models is shown in Fig. 5.9. EFN has proven to be worst of the four models. From this, one can conclude that some information whether a jet is quark-initiated or gluon-initiated is contained in non-infrared safe variables. So, if one wants to design a good discriminative model one must include non-infrared safe variables in jet representations. pPFN performs slightly better than EFN, but is significantly worse than PFN and PFN-ID. This shows us that $(z, \Delta y, \Delta\phi, (m))$ is a superior representation of the particle's kinematic properties than its four-vector (E, p_x, p_y, p_z),

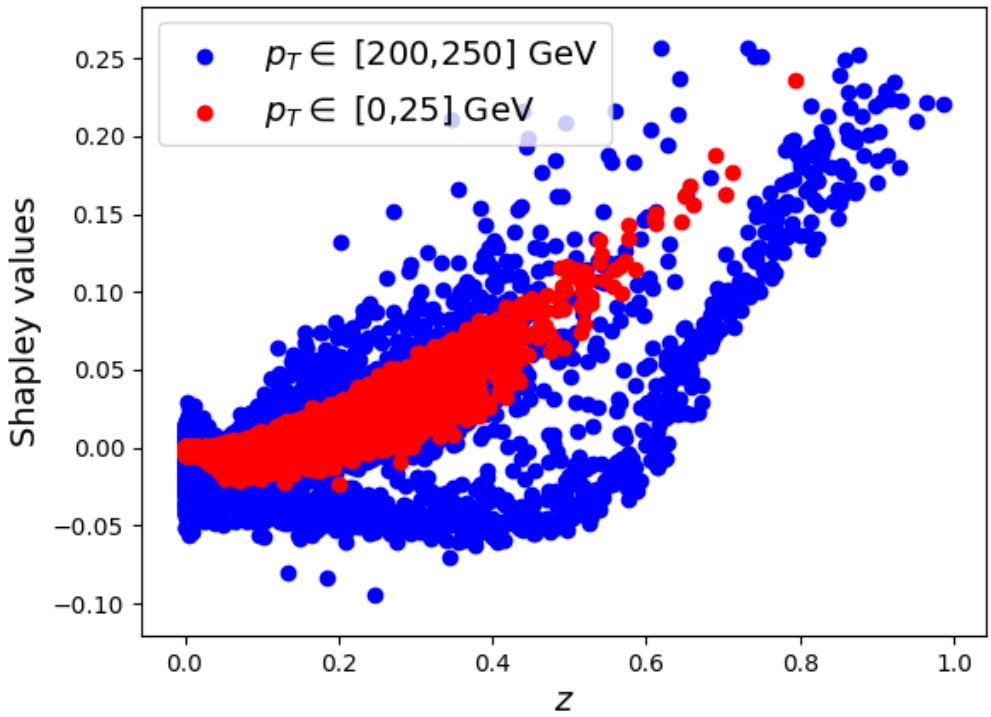


FIGURE 5.8: Shapley values for the pixels in z-jet images in dependence on the z value.

which can also be seen in the previous section with convolutional models. Finally, best discriminative power is achieved by the PFN-ID model. This confirms that the particle identity is very important in the predictive power. Furthermore, PFN-ID shows the best performance among all trained and used models consistently across the entirety of the p_T range. Fig. 5.10 shows the p_T dependence of the AUC metric of the PFN-ID in comparison with the zCNN-ID and the ANG models. The Shapley values for the features used by the PFN-ID are also calculated.

The scatter plots of Shapley values versus feature values for variables z, Δy, $\Delta\phi$ and m are presented in Figures 5.11 and 5.12. These values are calculated and presented together for all the particles in the jet. We were able to do this since the PFN-ID model is invariant with respect to ordering and it treats each particle independently. We can observe more positive contributions for particles with higher z and more negative contributions for particles with lower z. This agrees with the theoretical expectation, which states that gluon jets contain softer particles relative to quark jets. The importance of z increases almost linearly with its value for both p_T bins. This similarity can indicate a

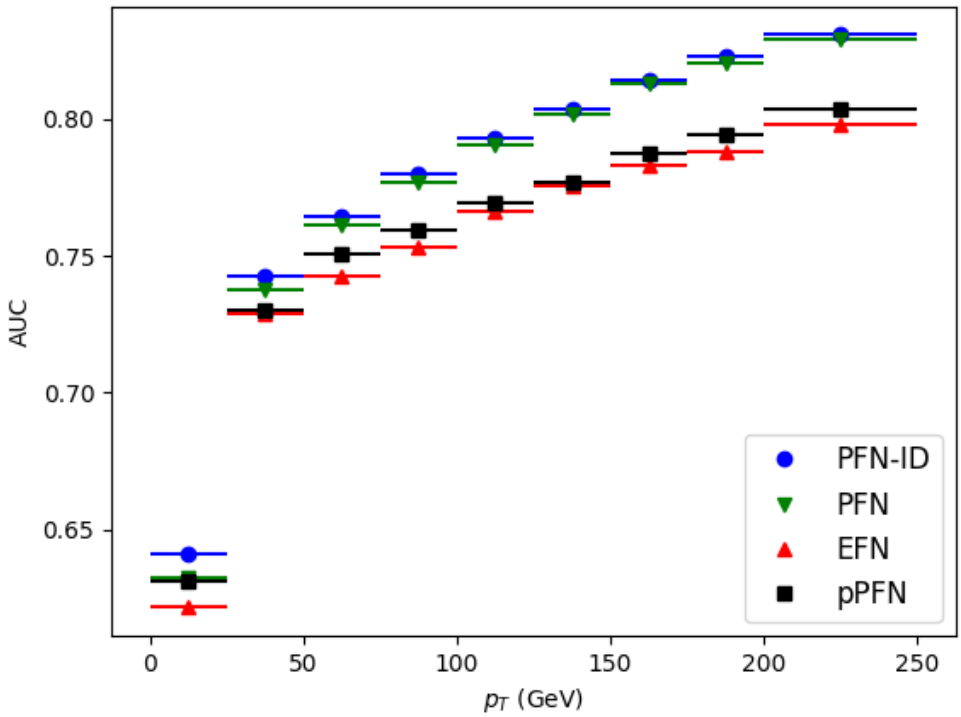


FIGURE 5.9: The p_T dependence of AUC metrics for ParticleFlow neural network models.

scaling symmetry of the QCD. The mean absolute values of the Shapley values for the masses of particles are higher for protons and heavier charged particles. This effect is more predominant in the lowest p_T bin where the presence of a heavier hadron almost certainly means that a jet is a quark jet. However, in the highest p_T bin, the particle mass does not have such a strong contribution as in the lowest p_T bin, but it is still impactful according to the mean absolute of the Shapley values. Finally, the Shapley values for Δy and $\Delta\phi$ confirm once again that jet width is crucial in the discrimination of quark and gluon jets, even for jets with low p_T. Fig. 5.12 clearly indicates that the model sees more collimated jets as quark jets and wider jets as gluon jets. It is interesting to observe that the contributions of these variables to the final prediction are more clear for jets in the lowest p_T bin. Still, it has to be noted that as with the PFI metric, this analysis is blind to the importance of a particular combination of variables on the model and only describes a particular feature unilaterally.

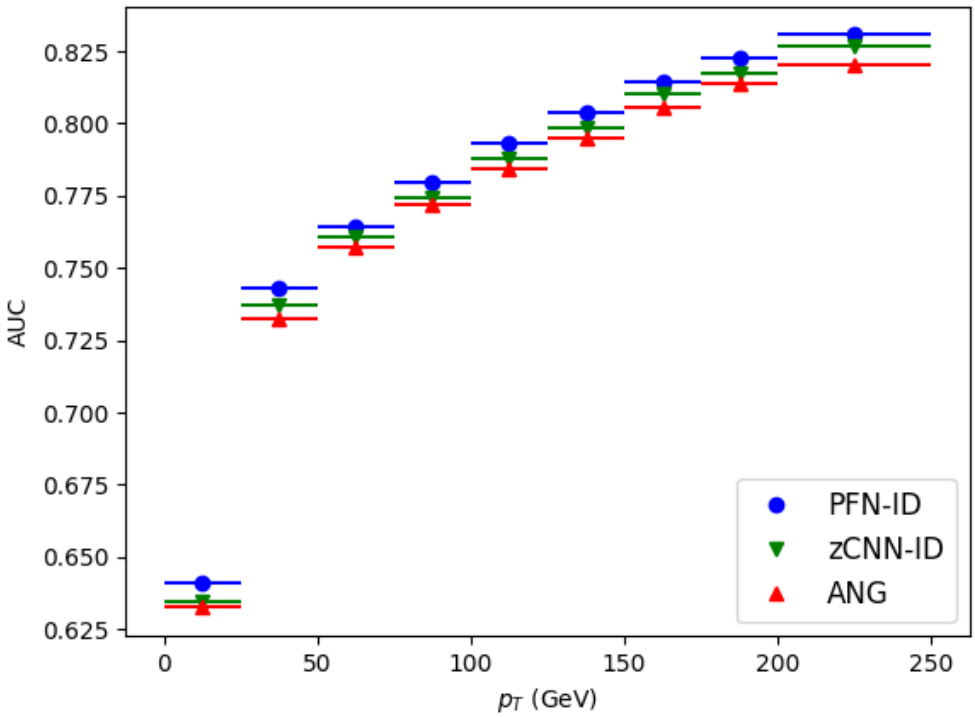
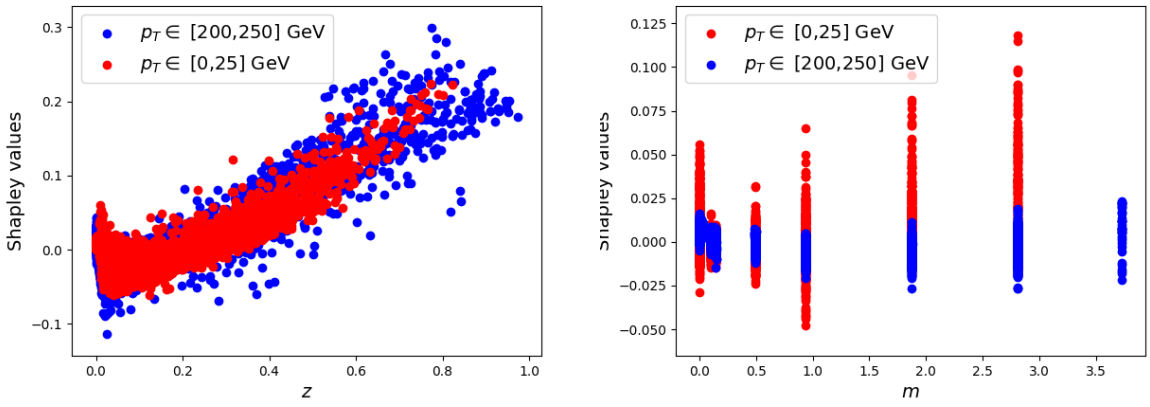
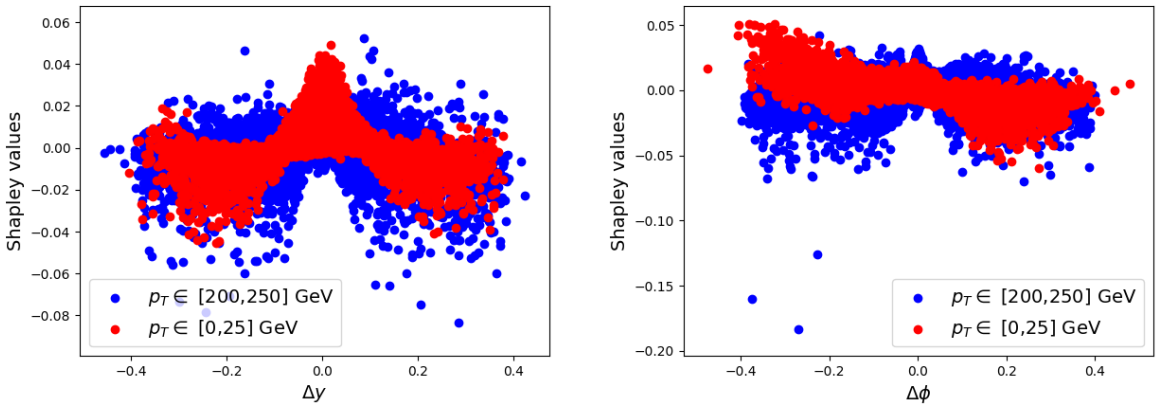


FIGURE 5.10: The comparison of AUC values for PFN-ID, zCNN-ID and ANG models.



(A) z (B) m

FIGURE 5.11: Shapley values for z and m variables against their values.



(A) Δy (B) $\Delta\phi$

FIGURE 5.12: Shapley values for Δy and $\Delta\phi$ variables against their values.

5.1.4 Model performance summary

A full summary of the AUC values obtained by each model for a particular p_T bin is shown in Table 5.1. Uncertainties of the AUC values are approximately in the range 0.001-0.002. The uncertainties are estimated from several training sessions of a particular model. The first conclusion that can be drawn is the universality of the model performance, meaning that the best model will perform the best across the entire p_T range. In our case that would be PFN-ID. It must be noted that PFN-ID is a family of models with the same architecture but different weights since they are separately trained over the samples from different p_T bins. Therefore, one should choose a distinct PFN-ID model for real data application.

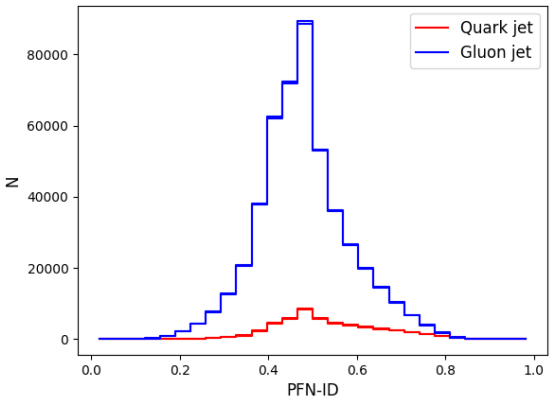
5.2 Quark-gluon separation

The general purpose of a tagger is to remove as much background as possible while keeping a sufficient number of signal events. In our case, signal events are quark-initiated jets and background events are gluon initiated jets. Such a tagger would be useful in analysis where pure or almost pure samples of quark jets are needed. In this section outputs of our models are analyzed on the test dataset and the possible purity of the samples is discussed. We used the best of our models, the PFN-ID. Figures 5.13 and 5.14 show histograms of the PFN-ID output values evaluated on the test dataset. This is done for each p_T bin separately. This way, one can visualize how well the

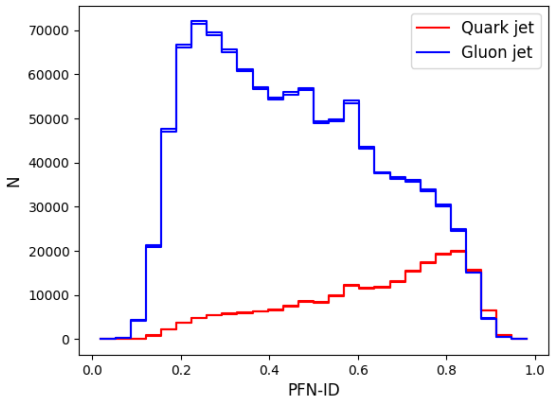
model separates quark and gluon jet samples. In many cases one wants to maximize the purity of the signal events. In this case we want to find the decision threshold, i.e., when the purity of quark jets sample is maximized for a sufficient quark efficiency. The purity can be defined as the ratio of true positives in total number of positives:

$$P = \frac{TP}{TP + FP}\,. \tag{5.1}$$

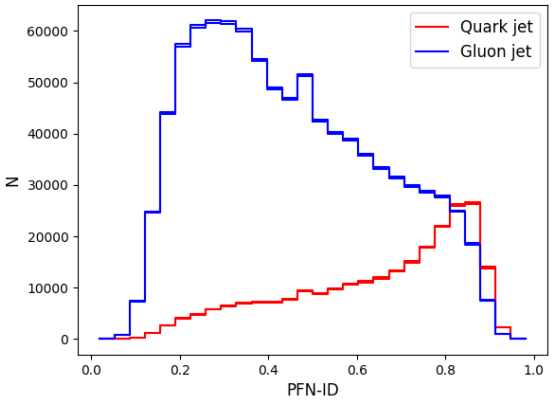
This way, after the application of the PFN-ID model and usage of a certain decision threshold t, the purity of the quark sample can be obtained. The threshold t can be chosen according to an analysis one wants to do. Some analyses want to keep a large number of quark jets regardless of the purity and some want pure quark jet samples.



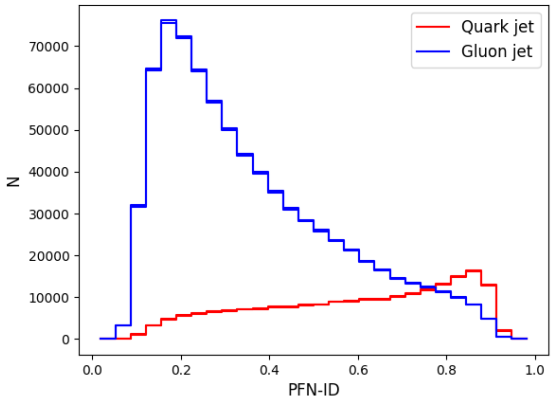
(A) $p_T \in$ [0-25 GeV]



(B) $p_T \in$ [25-75 GeV]

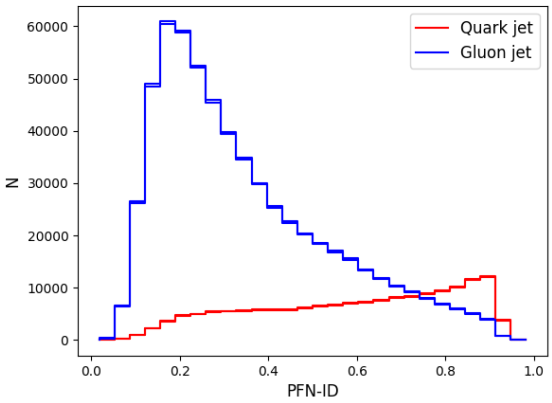


(C) $p_T \in$ [50-75 GeV]

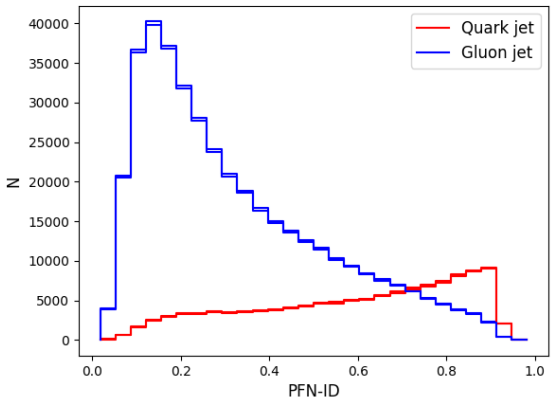


(D) $p_T \in$ [75-100 GeV]

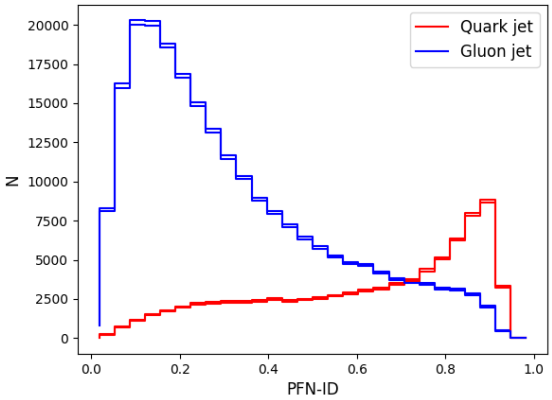
FIGURE 5.13: Histograms of the PFN-ID predictions for different p_T bins (0-100 GeV).



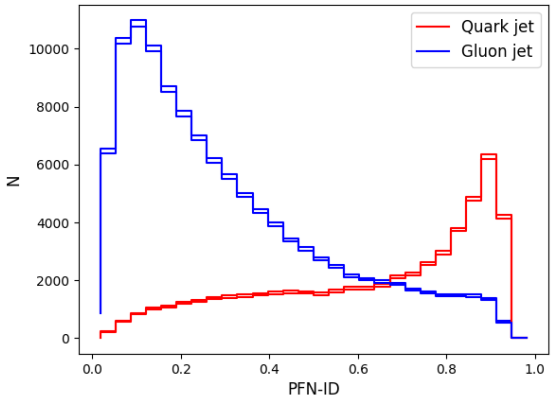
(A) $p_T \in$ [100-125 GeV]



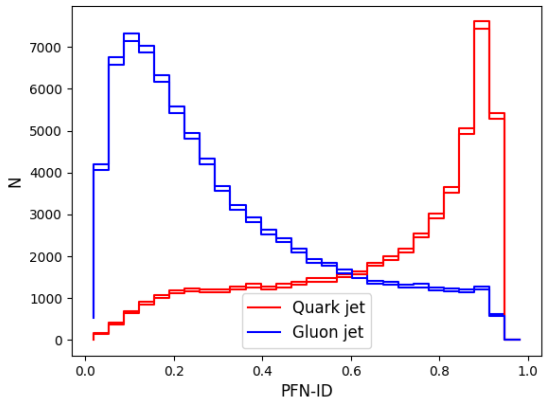
(B) $p_T \in$ [125-150 GeV]



(C) $p_T \in$ [150-175 GeV]



(D) $p_T \in$ [175-200 GeV]



(E) $p_T \in$ [200-250 GeV]

FIGURE 5.14: Histograms of PFN-ID predictions for different p_T bins (200-250 GeV).

The purities of the quark jet samples at the quark jet efficiency of 20% for different p_T bins are shown in Fig 5.15. The uncertainties are estimated by the propagation of the square root error through 5.1. We can notice increasing purities with increasing p_T, as expected.

At quark efficiency of 20% we see more than 50% pure quark samples for every p_T bin, except for the two lowest ones. An application of the PFN-ID to jets from the highest p_T bin can produce very pure quark jet samples with the purity of approximately 90%.

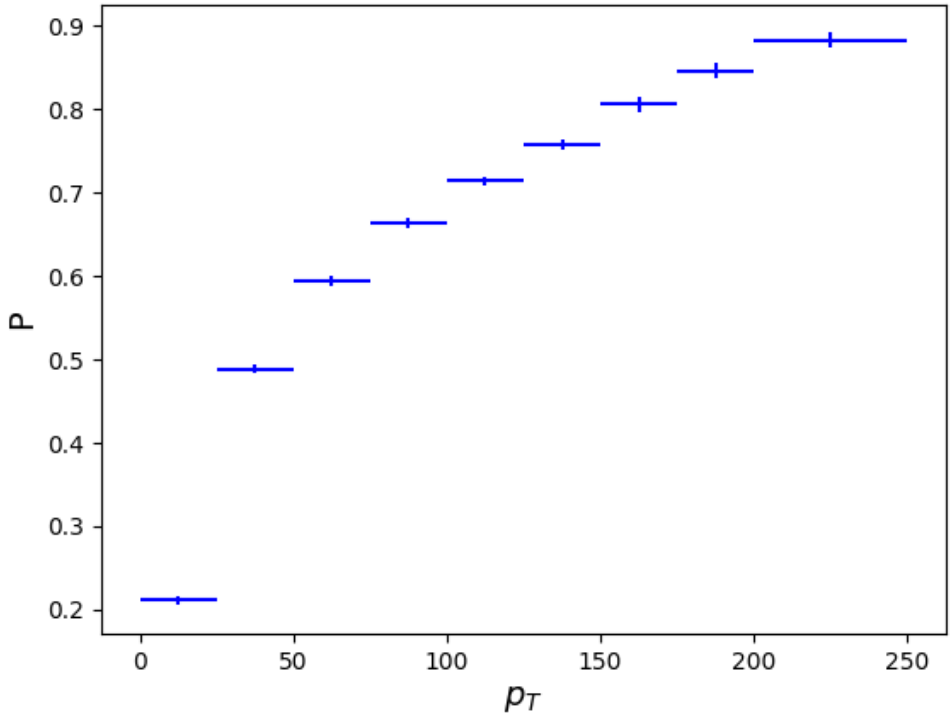


FIGURE 5.15: The p_T dependence of the produced quark jet purity by the PFN-ID model metrics at a quark efficiency of 20%.

5.3 Performance of the 2NN algorithm

In this section we present our findings after applying the $2NN$ algorithm on 500 000 jets created using the particle generator described in 4.2.2. During each iteration, the classifier is optimized using 50 000 randomly picked jets from the „real" dataset, and 50 000 jets generated using the neural network f. To optimize the neural network f, we use 50 000 jets as well. The algorithm performed 800 iterations in total. In each step, a single epoch is used when training the classifier and the neural network f in order to prevent overfitting on small subsamples, which would slow down the algorithm. After the final iteration, we obtain the calculated probability densities, which can be then used to generate samples of jets. First, we show the energy spectrum of the particles in the final state in jets generated by the calculated probabilities in Fig. 5.16. This spectrum is directly compared to the energy spectrum of the particles taken from jets belonging to the „real" dataset and shown on Figure 5.16.

The plotted spectra are obtained using 10 000 randomly selected jets from each dataset. The error bars in the histogram are smaller than the marker size and are hence not visible. A resemblance between the two spectra is notable, especially at higher energies. This points to the fact that the calculated probabilities are approximately correct, so we can use them to generate samples of jets that resemble „real" jets. To further examine the calculated probability densities we need to reconstruct the hidden resonances which are not found in the final state. For this purpose, the calculated probability densities for mother particle masses of $M = 25.0$, $M = 18.1$, $M = 14.2$ and $M = 1.9$ are analyzed and compared to the real probability densities in the following subsections. These masses are chosen since they match the masses of the hidden resonances, as was introduced earlier in table 4.2.

5.3.1 Mother particle with mass M = 25.0

The calculated $2D$-probability density $p(m_1, m_2 \mid M)$ is shown in Figure 5.17, compared to the real probability density. A simple look reveals that 3 possible decays of particle of mass $M = 25.0$ are recognized by the algorithm. After dividing the probability space as in panel (c) in Figure 5.17 with lines $m_2 > 16.0$ and $m_2 < 10.0$, we calculate the mean and the variance of the data on each of the subspaces. As a result, we obtain $(m_1, m_2) = (18.1 \pm 0.5, 6.1 \pm 0.5)$ for $m_2 > 16.0$, $(m_1, m_2) = (14.0 \pm 0.7, 8.4 \pm 0.7)$ for $16.0 \leq m_2 > 10.0$ and $(m1, m2) = (4.8 \pm 0.2, 4.6 \pm 0.2)$ for $m_2 \leq 10.0$. These values agree with the masses of the resonances expected as the products of decays of the particle with mass

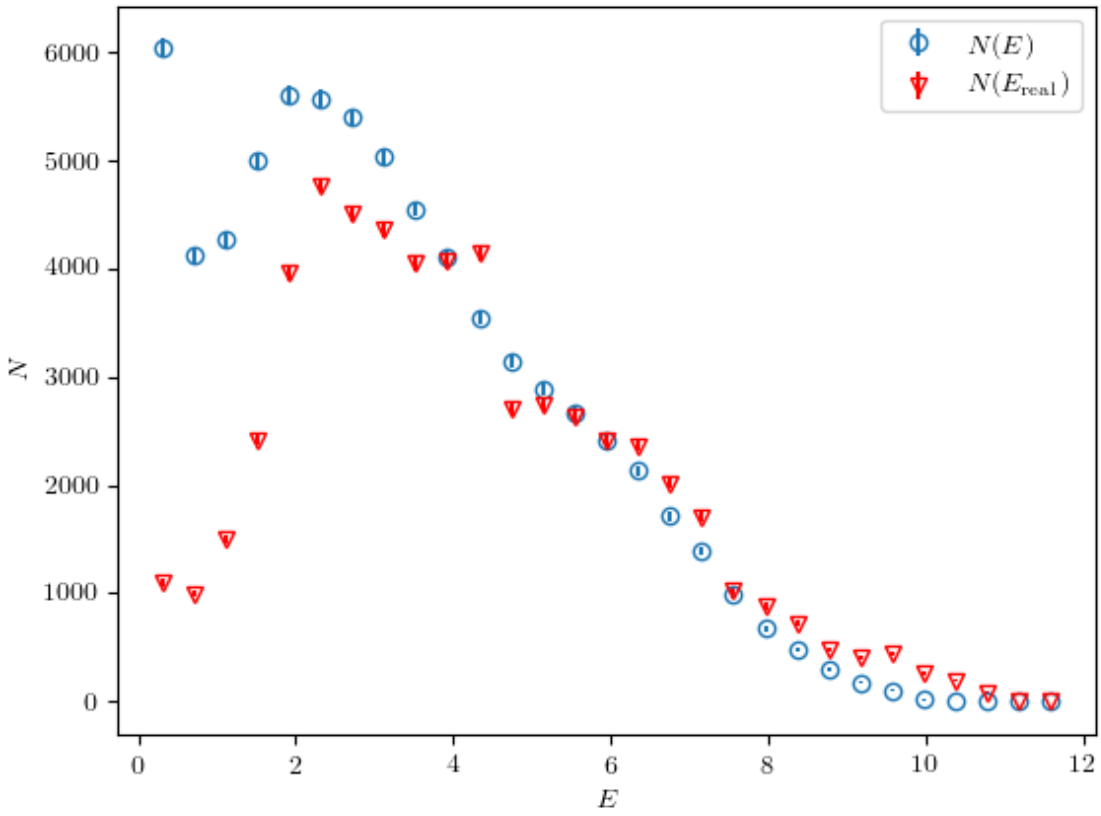
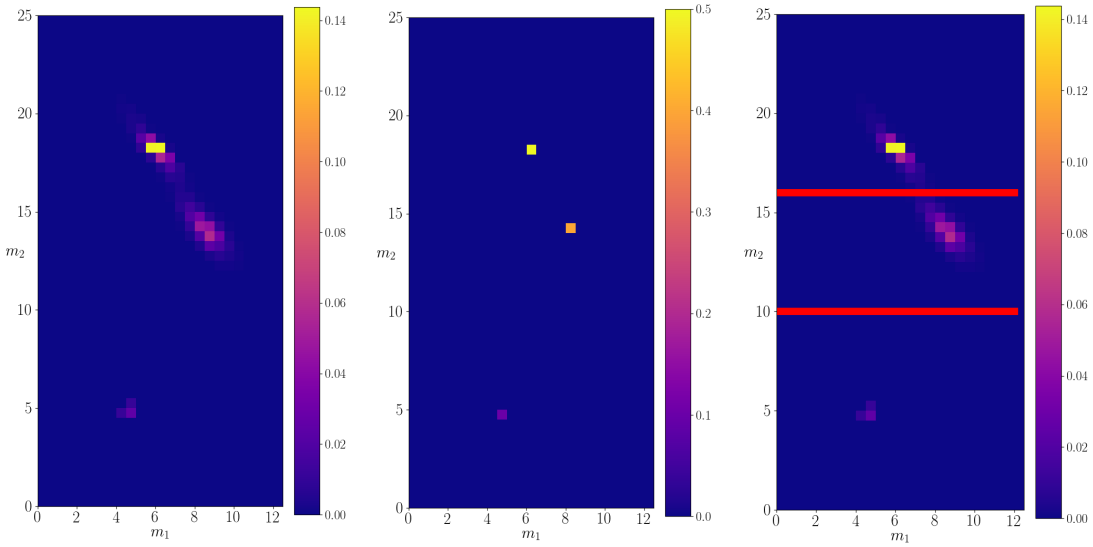


FIGURE 5.16: The energy spectrum of the particles in the final state in jets generated by the calculated probabilities, compared to the energy spectrum of particles taken from jets belonging to the „real" dataset.

$M = 25.0$. The calculated small variances indicate that the algorithm is very precise. The total decay probabilities for each of the subspaces are equal to $p_1 = 0.48$, $p_2 = 0.47$, $p_3 = 0.05$, which approximately agree with the probabilities of decay channels of the particle with mass $M = 25.0$, as defined in table 4.2.

These results show that we can safely assume that the $2NN$ algorithm successfully recognizes the decay modes of the particle that initiates a jet. In this specific case, all three possible decays were clearly found. To quantify the difference between the calculated probability density and the real probability density, we use the KL-divergence.

Figure 5.18 shows the dependence of the KL-divergence on the iteration index of the $2NN$ algorithm. At first, we observe a steep decrease in the value of the divergence. Large variations in the divergence value are observed later. This is an indicator that the approximate probability density is found relatively quickly - after a few hundred



(A) (B) (C)

FIGURE 5.17: The calculated probability density for a decaying particle of mass $M = 25.0$. (**a**) The left panel shows the density evaluated on the entire discretized probability space. (**b**) The probability density of „real" data. (**c**) A division of the probability space into three subspaces, in order to isolate particular decays.

iterations. As the algorithm decreases the width of the peaks found in the probability distribution, the KL-divergence becomes very sensitive to small variations in the location of these peaks and can therefore vary by a large relative amount.

5.3.2 Mother particle with mass M = 18.1

A similar analysis as for the mother particle with mass M = 25.0 is performed for the particle with mass $M = 18.1$. The calculated probability density is shown in Figure 5.19 compared to the expected probability density. In this case, only one decay is allowed, so a division into probability subspaces is not necessary, as was for the case when M = 25.0. The calculated mean and the variance of the shown probability density are $(m_1, m_2) = (5.9 \pm 0.4, 8.2 \pm 0.6)$. In this case, just as in the former, the calculated values closely agree with the only possible decay, in which the mother particle decays into two particles of masses 6.1 and 8.4. Also, just as in the previous subsection, the obtained result is very precise. Therefore, the algorithm can successfully find hidden resonances, as well as recognize the decay channels, without ever seeing them in the final state in the „real" dataset.

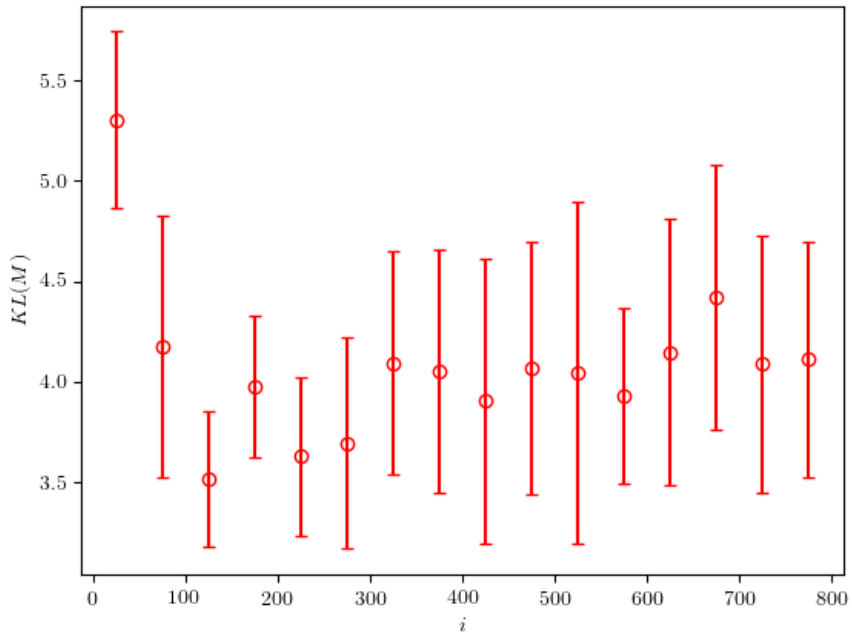


FIGURE 5.18: The KL-divergence between the calculated and the real probability densities, evaluated in the case of particle of mass $M = 25.0$. The presented results are averaged over 50-iteration intervals. The error bars represent the standard deviation calculated on the same intervals.

The calculated KL-divergence in the case of particle with mass $M = 18.1$ decreases over time in a very smooth manner, as can be seen in Figure 5.20. We believe this could be due to the simpler expected probability density, which the algorithm manages to find very quickly.

5.3.3 Mother particle with mass M = 14.2

Figure 5.21 shows the $2D$-probability density for the decaying particle of mass $M = 14.2$. In this case, we can identify 3 possible decay channels, which are not as clearly separated as the channels in the previous subsections. Similar to the case of decaying particle of mass $M = 25.0$, we divided the probability space into 3 subspaces, each of which covers one of the possible decays. In this case, the three subspaces cover areas where $m_2 \leq 4.0$, $4.0 < m_2 \leq 5.5$ and $m_2 > 5.5$. The calculated mean values of the probability density on each of the subspaces are $(m_1, m2) = (2.4 \pm 0.5, 2.9 \pm 0.7)$, $(m_1, m_2) = (2.7 \pm 0.7, 4.3 \pm 0.3)$ and $(m_1, m_2) = (4.4 \pm 0.4, 6.2 \pm 0.3)$, respectively. The

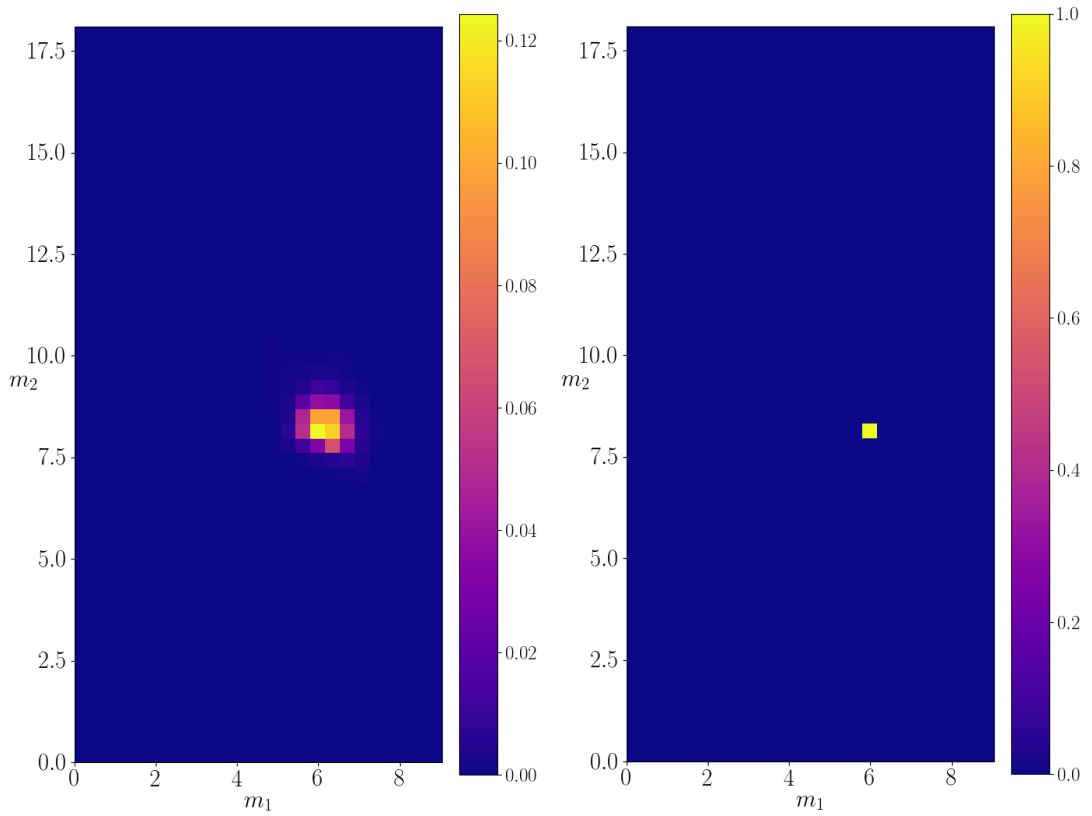


FIGURE 5.19: The calculated probability density for a decaying particle of mass $M = 18.1$. (**a**) The calculated density evaluated on the entire discretized probability space. (**b**) The probability density of „real" data.

allowed decays of a mother particle with mass $M = 14.2$ in the „real" data are into channels with masses $(1.9, 1.9)$, $(1.9, 4.4)$ and $(4.4, 6.2)$, which agree with the calculated results. However, in this case the calculations show higher variance, especially for decays where one of the products is a particle with mass 1.9. The total probabilities of decay in each of the subspaces are 0.89, 0.05 and 0.06, respectively. The relative probabilities of decay channels into particles with masses (4.4, 6.1) and (1.9, 4.4) are approximately the same as expected. However, the algorithm predicts more decays in the channel (1.9,1.9) than expected. The KL-divergence shows a steady decrease with occasional spikes, as shown on Figure 5.22.

5.3.4 Mother particle with mass $M = 1.9$

The last probability density we analyze is the probability density for the mother particle with mass $M = 1.9$. Figure 5.23 shows the calculated probability density. It can

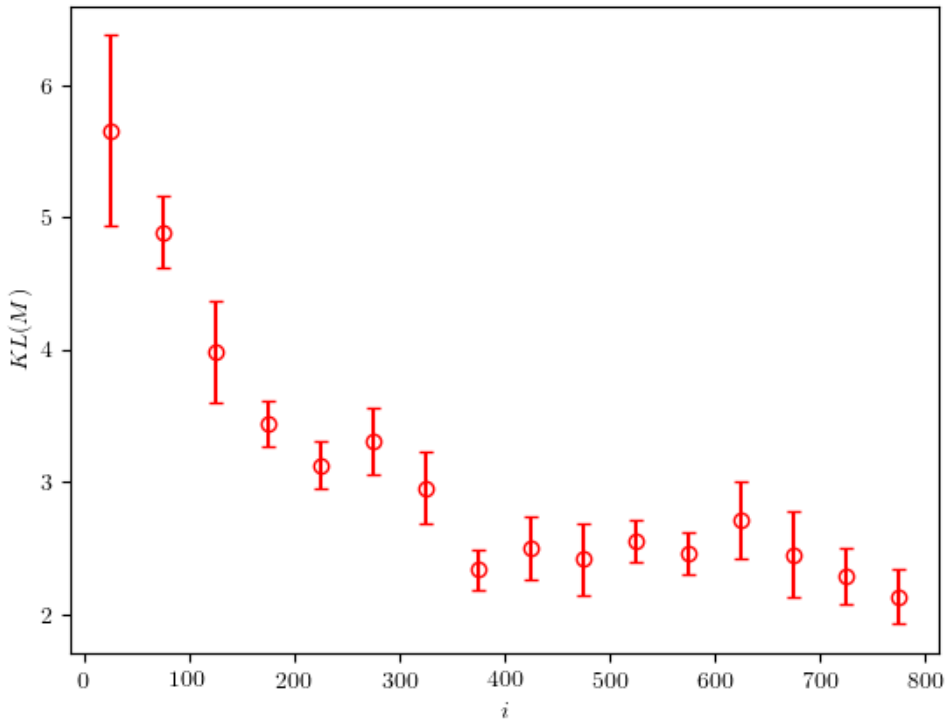
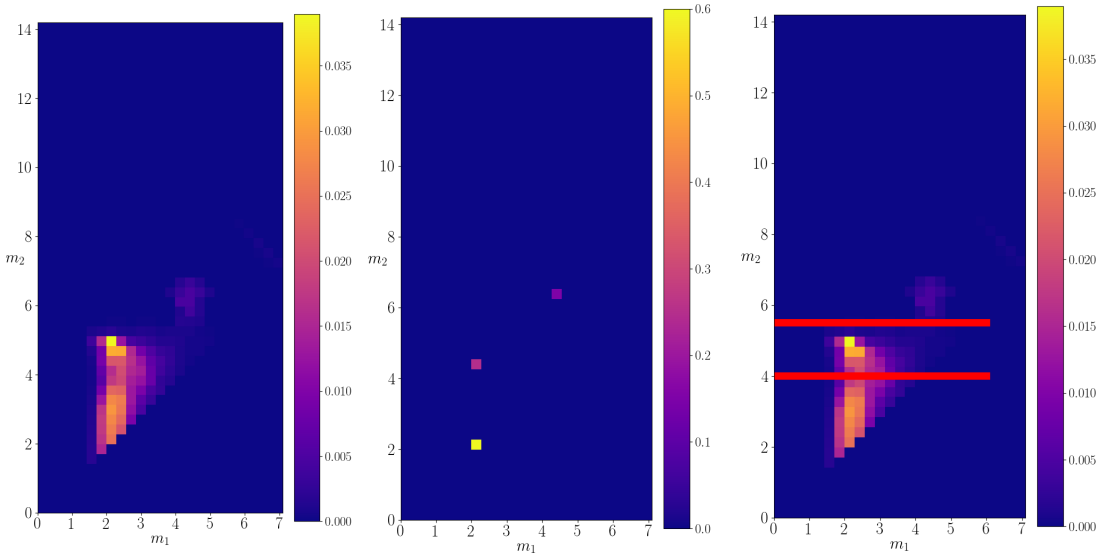


FIGURE 5.20: The KL-divergence between the calculated and the real probability densities, evaluated in the case of particle of mass $M = 18.1$. The presented results are averaged over 50-iteration intervals. The error bars represent the standard deviation calculated on the same intervals.

be seen that one of the decay modes present in the „real" data, namely when the particle decays in the $(0.1, 0.1)$ channel, is not recognized by the algorithm, but the decay mode when the particle decays in the $(0.1, 1.3)$ channel is visible. If we isolate the given decay as shown in the right panel of Figure 5.23, we get a mean value of $(m_1, m_2) = (0.14 \pm 0.09, 1.27 \pm 0.09)$, which agrees with the expected decay. We also observe significant decay probabilities along the line $m_1 + m_2 = 1.9$. The decays that correspond to the points on this line in effect create particles with zero momentum in the rest frame of the mother particle. In the lab frame this corresponds to the daughter particles flying off in the same direction as the mother particle. Since they reach the detector in the same time, they are registered as one particle of total mass $M = 1.9$. Thus, we can conclude that the probabilities on this line have to add up to the total probability of the mother particle not decaying. The calculated probabilities in the case of no decay and in the case when decaying into particles with masses $(0.1, 1.3)$ are 0.71 and



(A) (B) (C)

FIGURE 5.21: The calculated probability density for a decaying particle of mass $M = 14.2$. (**a**) The left panel shows the density evaluated on the entire discretized probability space. (**b**) The probability density of „real" data. (**c**) A division of the probability space into three subspaces, in order to isolate particular decays.

0.29, respectively. We note that relative probabilities are not correct, but 2 of the 3 decay modes are still recognized by the algorithm. The KL-divergence in this case can't produce reasonable results, simply because of multiple points in the (m_1, m_2) phase space which produce the same decay and is therefore omitted from the analysis. We summarize the obtained results for different masses in Table 5.2.

5.3.5 The accuracy of the classifier

The accuracy of the classifier is defined as the fraction of correctly „guessed" samples on a given dataset. The criterion used for guessing is checking whether the output of the classifier, C_{NN}, is greater than 0.5. The accuracy can indirectly indicate how distinguishable are two given datasets. In our algorithm, after starting from a test probability density, we approach the real probability density with increasing iteration number, so we can expect that the two jet datasets, the „real" and the „test" dataset, are less and less distinguishable over time. In Figure 5.24 we show the accuracy of the classifier in dependence on the iteration number.

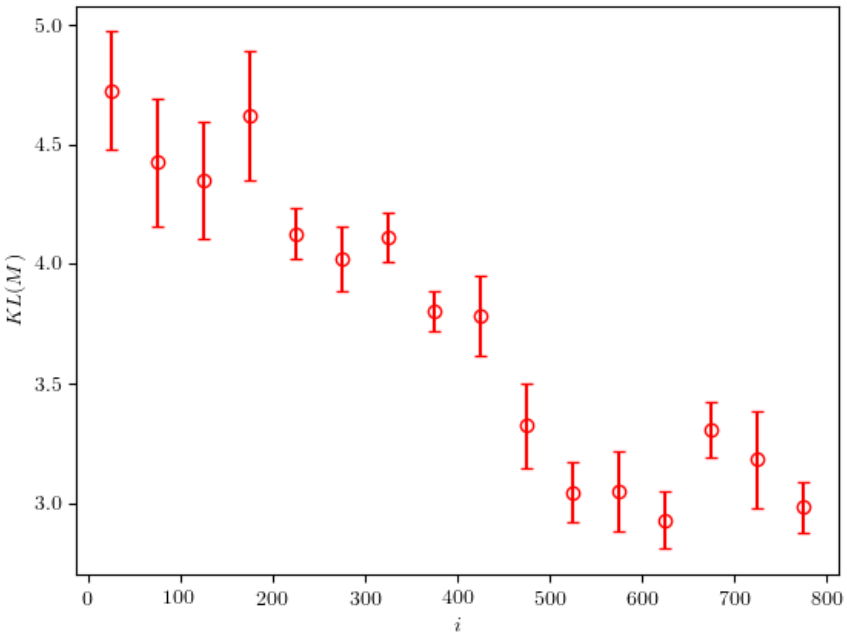
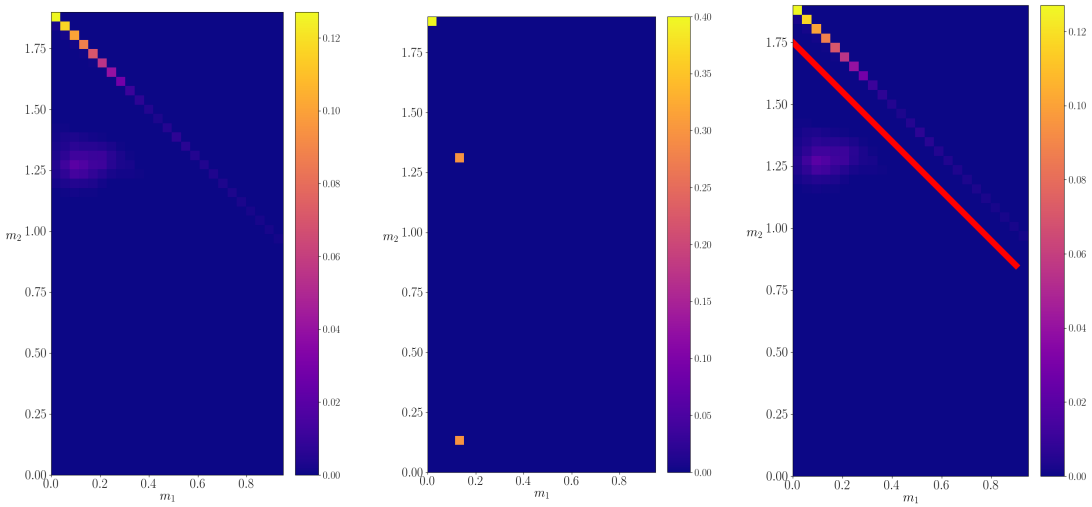


FIGURE 5.22: The KL-divergence between the calculated and the real probability density evaluated for a particle of mass $M = 14.2$. The results are averaged over intervals of 50 iterations. The error bars represent the standard deviations on the same interval.

After an initially high value, the accuracy decreases with growing iteration number, which demonstrates that the test dataset becomes more and more similar to the real dataset. Ideally, the datasets are no longer distinguishable by a given classifier if the evaluated accuracy reaches 0.5. Therefore, we can use the evaluated accuracy of the classifier as a criterion for stopping the algorithm. Other measures can also be used as the stopping criterion such as the loss value of the classifier or the area under ROC curve of the classifier. In this work, the algorithm is stopped after the accuracy reaches a value of 0.65, because we didn't see any significant decrease in the accuracy once it reached this value. An accuracy value of 0.65 clearly shows that the classifier is capable of further discriminating between the two datasets. This is explained by the fact that the neural network f and its hyperparameters are not fully optimized. For the algorithm to perform better, we need to optimize the neural network f and possibly improve the architecture for the selected task.



(A) (B) (C)

FIGURE 5.23: The calculated probability density for a decaying particle of mass $M = 1.9$. (**a**) The left panel shows the density evaluated on the entire discretized probability space. (**b**) The probability density of „real" data. (**c**) A division of the probability space into three subspaces in order to isolate particular decays.

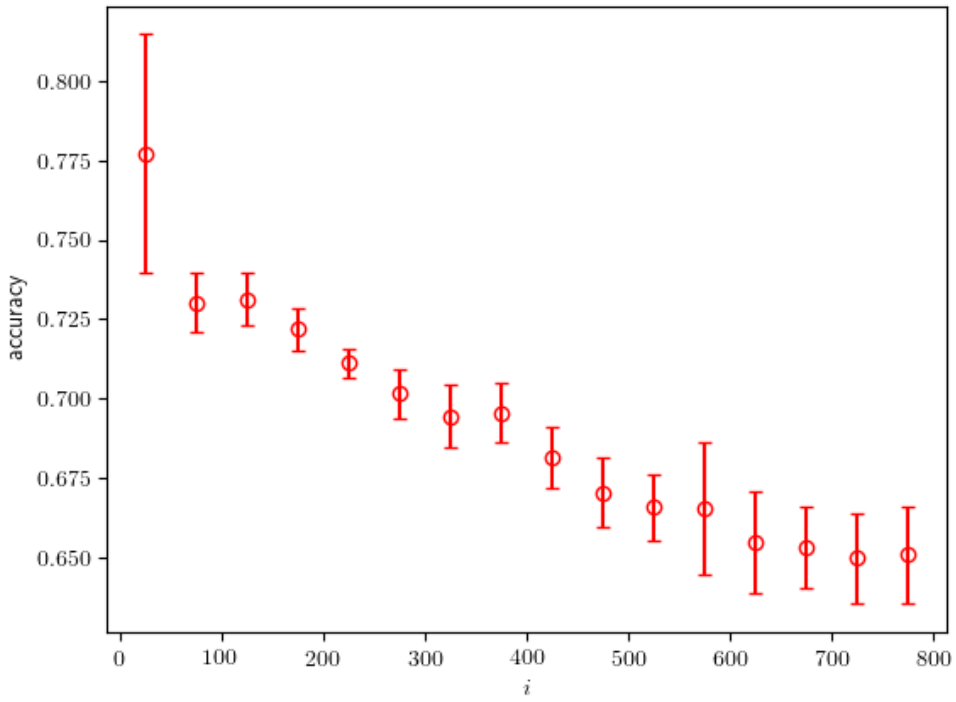


FIGURE 5.24: The calculated accuracy of the classifier in dependence on the iteration number.

TABLE 5.1: A summary of AUC values for different models for each p_T bin.

p_T	LR	ANG	CNN	zCNN	zCNN-ID
0-25 GeV	0.630	0.632	0.623	0.631	0.635
25-50 GeV	0.727	0.732	0.733	0.734	0.737
50-75 GeV	0.748	0.757	0.757	0.758	0.761
75-100 GeV	0.759	0.772	0.769	0.772	0.774
100-125 GeV	0.771	0.784	0.782	0.786	0.788
125-150 GeV	0.780	0.795	0.791	0.796	0.798
150-175 GeV	0.790	0.805	0.800	0.808	0.810
175-200 GeV	0.797	0.814	808	0.816	0.817
200-250 GeV	0.802	820	0.816	0.825	0.827
p_T	EFN	pPFN	PFN	PFN-ID	
0-25 GeV	0.622	0.631	0.632	0.641	
25-50 GeV	0.729	0.730	0.737	0.743	
50-75 GeV	0.743	0.751	0.761	0.764	
75-100 GeV	0.753	0.759	0.777	0.780	
100-125 GeV	0.766	0.770	0.790	0.793	
125-150 GeV	0.775	0.777	0.802	0.804	
150-175 GeV	0.783	0.788	0.813	0.814	
175-200 GeV	0.788	0.794	820	0.823	
200-250 GeV	0.798	804	0.829	0.831	

particle mass	25		25		25	
decay masses	18.1	6.1	14.2	8.4	4.4	4.4
reconstructed masses	18.1 ± 0.5	6.1 ± 0.5	14.0 ± 0.7	8.4 ± 0.7	4.8 ± 0.2	4.6 ± 0.2
channel probability	0.5		0.4		0.1	
reconstructed probability	0.48		0.47		0.05	
particle mass	18.1		14.2		14.2	
decay masses	8.4	6.1	6.1	4.4	4.4	1.9
reconstructed masses	8.2 ± 0.6	5.9 ± 0.4	6.2 ± 0.3	4.4 ± 0.4	4.3 ± 0.3	2.7 ± 0.7
channel probability	1		0.15		0.25	
reconstructed probability	1		0.05		0.06	
particle mass	14.2		1.9		1.9	
decay masses	1.9	1.9	1.3	0.1	no decay	
reconstructed masses	2.4 ± 0.5	2.9 ± 0.7	1.27 ± 0.09	0.14 ± 0.09	no decay	
channel probability	0.6		0.3		0.4	
reconstructed probability	0.89		0.29		0.71	
particle mass	14.2		1.9		18.1	
decay masses	1.9	1.9	1.3	0.1	no decay	
reconstructed masses	2.4 ± 0.5	2.9 ± 0.7	1.27 ± 0.09	0.14 ± 0.09	no decay	
channel probability	0.6		0.3		0.4	
reconstructed probability	0.89		0.29		0.71	

TABLE 5.2: The characteristics of the reconstructed decay channels for decaying particles with masses $M = 25$, 18.1, 14.2 and 1.9. The decay channel for the particle with mass 1.9 which was not reconstructed is not given here.

Chapter 6

Conclusion

This research presents an introduction to machine learning models and techniques used to discriminate between quark-initiated and gluon-initiated jets for applications on data gathered by the ALICE experiment at CERN. This is done by using a large amount of Monte Carlo simulated data to train and validate the proposed models. A comprehensive analysis of the models and the predictions is performed. The model that shows the best performance is PFN-ID, a specially designed neural network which uses particle level kinematics along with the information of particle flavour. PFN-ID is universally the best model since it is superior to the every other model across the whole range of observed jet total momenta.

Several conclusions can be drawn from the analysis. The particle-level variables should first be preprocessed in order to minimize the amount of symmetry in data. Models that use raw four-momenta showed inferior performance to the models that use fractions of total transverse momenta, rapidity and azimuthal angle relative to the jet axis. Another conclusion is that a particle-level flavour is very important in building discriminators between quark and gluon jets. Each model that uses this information performs better than its analogue which doesn't use it. This may be very convenient for applications on data collected by the ALICE detector since that detector exhibits very good particle identification capabilities. Models that use global jet variables, such as generalized angularities, perform slightly worse than models that use particle-level information (including particle identity). This is to be expected since at the particle-level, the data contain much more information which the models with sufficient complexity can exploit. However, such models can be computationally demanding and difficult to train.

The discriminative power of each of the trained models increases with the total jet p_T, regardless of the architecture and data representation. This can imply that samples

of quark and gluon jets are mutually more inseparable at low p_T. The radiation patterns inside the parton shower may be too similar to retrieve the initial parton flavour with satisfying accuracy. The origin of jets with high p_T (200 GeV) can be deduced very well, so that we are able to achieve high purity samples of quark jets. We note that the achieved purity continuously increases with increasing jet p_T.

Feature importance analysis shows us which variables are significant in a model's decision process. The most important global variables are jet width and jet multiplicity, both being special cases from the family of generalized angularities. Jet width is dominantly important for models trained on jets with low p_T. The importance of other variables increases with increasing jet p_T. The feature importance performed on jet images show an area around the jet axis which is the most sensitive to model decisions. This effective area decreases with increasing jet p_T.

Furthermore, we developed a method for calculating underlying probability distributions in particle decays, using only data that can be collected in a real-world physical system. First, we generate an artificial physical system based on a part of the QCD fragmentation process. Next, we present the core part of the method: the $2NN$ algorithm, which we describe in detail. The algorithm performs very well when tested on the developed physical system. It accurately predicts most of the hidden resonant particles, as well as their decay channels, which can occur in the evolution of jets. The energy spectra of the particles in the final state can also be accurately reproduced.

Although tested only on the developed artificial physical system, we believe that the method is general enough to be applicable to real-world physical systems, such as collisions of high-energy particles, with a few possible modifications. For example, we hope that, in the future, this method can prove helpful in measuring the fragmentation functions of quarks and gluons. Additionally, one could employ such a method in the search for supersymmetric particles of unknown masses, or in measuring the branching ratios of known decays.

The $2NN$ algorithm does not specify the exact architecture of used the neural networks, nor the representation of the data used. In fact, the classifier does not need to be a neural network at all – it can be any machine learning technique which maximizes likelihood. Although the algorithm has a Generative Adversarial Network (GAN)-like structure, it converges readily and does not show issues that are usually associated with GANs, such as mode collapse or vanishing gradients. The downside of the presented algorithm are high computational requirements. Continuous probability distributions, which we expect to occur in nature, are approximated by discrete

probability distributions. In quest for higher precision and a better description of reality, one always aims to increase the resolution of discrete steps, but this carries a high computational cost. We note that in our case, the used neural networks are not fully optimized, which slows down the convergence of the algorithm. In conclusion, in order to cut down computational costs, a more thorough analysis of convergence is needed to achieve better performance.

In future work we hope to make the method even more general and thus even more applicable to real-world physical systems. Currently, the method approximates a part of the QCD decay tree, covering $1 \to 2$ decays. Even though we prove that some of the decay characteristics can be recovered from the final state without prior knowledge of the underlying processes, the method still doesn't lend itself for use on general QCD data. To remedy that, we want to introduce angle dependent probability distributions, which can be retrieved from detector data. We would also like to investigate the possibility of including other decay modes, such as $1 \to 3$ type decays. Finally, we plan to include other processes that appear naturally in QCD, such as $2 \to 2$ and $2 \to 3$ type interactions.

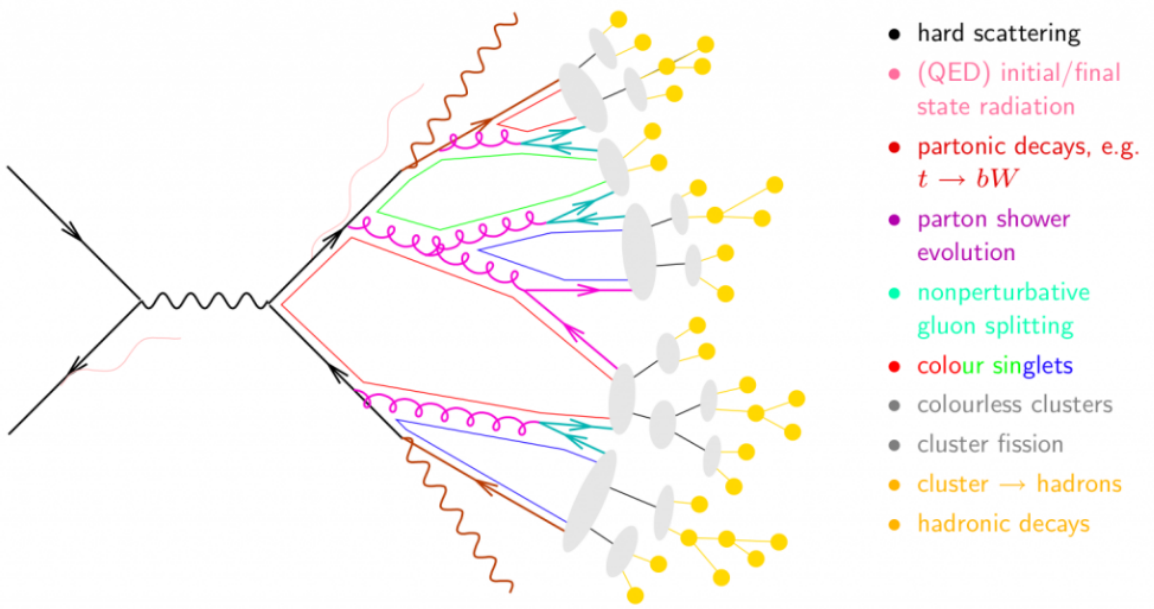
Poglavlje 7

Hrvatski prošireni sažetak

7.1 Uvod

Sudaranje čestica pri visokim energijama daje nam eksperimentalni uvid u fiziku elementarnih čestica. U Europskom centru za nuklearna istraživanja (CERN) nalazi se najveći postojeći sudarivač koji se naziva Veliki Hadronski Sudarivač (LHC). Trenutna dostupna energija sudara u sustavu centra mase na LHC-u je čak 13 TeV po nukleonu. Jedan od najčešćih objekata koji se pojavljuje kao posljedica sudara su tzv. mlazovi. To su usmjerene skupine čestica, uglavnom hadrona, detektirane u eksperimentima. Mlazovi su eksperimentalne manifestacije kvarkova i gluona koji su nastali u procesima s visokim prijenosom energije, takozvanim „tvrdim" procesima. Kvarkovi i gluoni često se zajedničkim imenom nazivaju partoni. Tijekom procesa fragmentacije početni partoni emitiraju dodatne partone koji kaskadno emitiraju još novih partona. Pojedine partone nije moguće detektirati izolirane od ostalih partona zbog temeljnih svojstava kvantne kromodinamike kao što su zatočenje boje i asimptotska sloboda te se procesom hadronizacije grupiraju u bojne singlete koji se nazivaju hadroni. Najčešće detektirani hadroni su pioni, kaoni i protoni. Pioni i kaoni spadaju u kategoriju mezona, bojnih singleta koji se sastoje od dva kvarka. Protoni i neutroni se za razliku od njih sastoje od tri kvarka te spadaju u kategoriju bariona.

Postoje razni algoritmi za rekonstrukciju mlazova na eksperimentima. Najčešće korišteni algoritmi su k_T, Cambridge/Aachen (C/A) i anti-k_T algoritmi. Oni spadaju u skupinu rekominacijskih sekvencionalnih algoritama u kojima se parovi čestica iterativno rekombiniraju zbrajanjem njihovih četveromomenata. Par čestica koji će se rekombinirati se odabire prema najmanjoj udaljenosti među njima. Navedeni algoritmi se razlikuju samo prema načinu definiranja udaljenosti među česticama. Cambridge/Aachen i k_T algoritmi se često koriste pri analizama unutarnje strukture mlaza,



SLIKA 7.1: Shematski prikaz evolucije mlaza [16].

dok se anti-k_T algoritam koristi u slučajevima gdje je bitno točno identificirati mlazove zbog njegove neosjetljivosti na razne neželjene pozadinske događaje i efekte koji se pojavljuju u praksi. Nakon rekonstrukcije mlazova, njihova globalna kinematička svojstva mogu se identificirati s kinematičkim svojstvima partona koji je inicirao taj mlaz. Dok se kinematičke varijable kao što su energija, rapiditet ili azimutalni kut relativno lako povežu s istovjetnim kinematičkim varijablama početnog partona, informacija o njegovoj vrsti nije trivijalno dostupna. Posjedovanje te informacije bi uvelike pomoglo u raznim analizama u eksperimentalnoj fizici elementarnih čestica. Varijable kao što su multiplicitet mlaza ili ukupna masa mlaza posjeduju moć razlikovanja kvarkovskih i gluonskih mlazova do neke mjere. Međutim, za bolje raspoznavanje i identifikaciju okusa početnog partona potrebne su naprednije metode. Zbog velikih količina dostupnih podataka i visoke dimenzionalnosti u opisima mlazova, tehnike strojnog učenja su sve popularnije u rješavanju ovog problema.

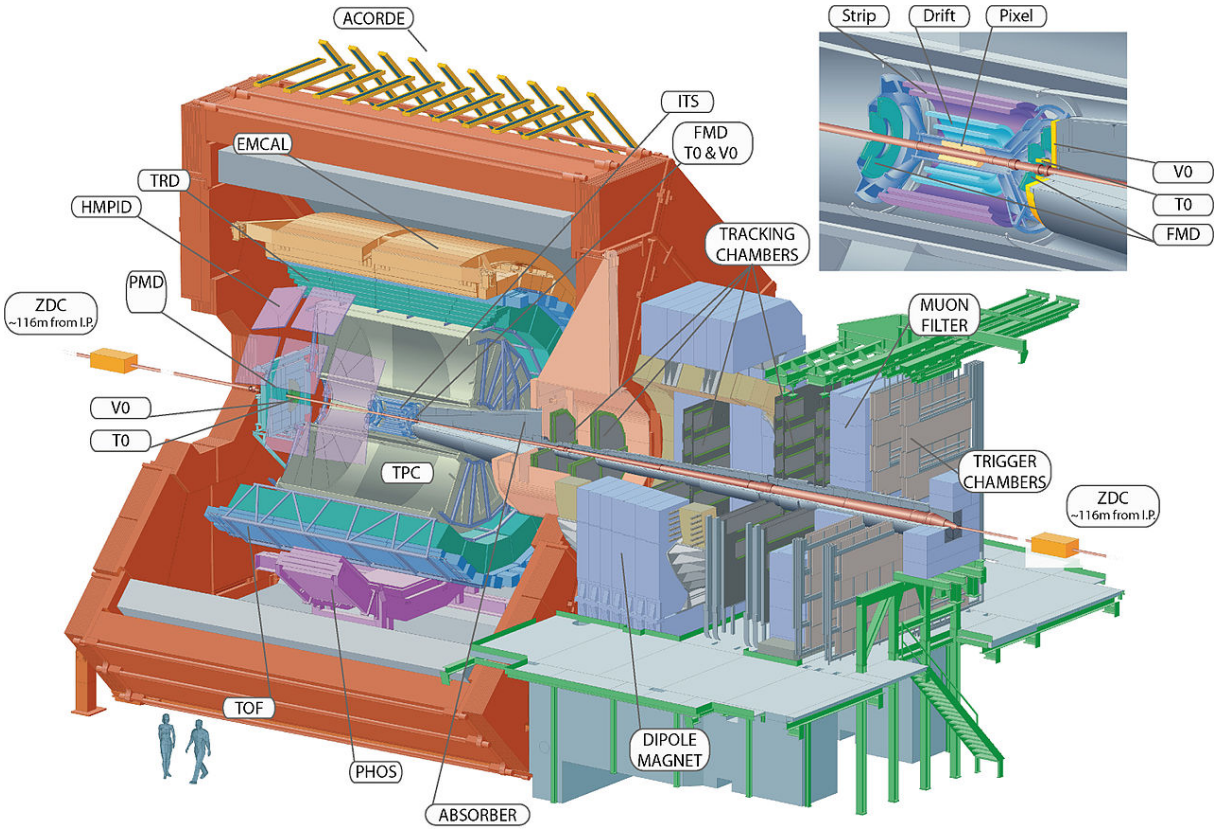
Cilj ovoga istraživanja je razviti metodu koristeći tehnike strojnog učenja koja bi razlikovala kvarkovske i gluonske mlazove u podacima prikupljenima na detektoru ALICE. Također, dodatni cilj je razviti metodu koja bi mogla pomoći u određivanju fragmentacijskih funkcija za kvarkove i gluone, konkretno metodu koja bi određivala vjerojatnosti emitiranja partona tijekom evolucije mlaza.

U sljedećim poglavljima opisani su ukratko LHC i detektor ALICE. Također, predstavljen je pregled najčešćih tehnika strojnog učenja. U metodologiji su opisane sve korištene metode uz opis razvijenih algoritama. Nadalje, prezentirani su dobiveni rezultati, kao i evaluacija razvijenih metoda i algoritama. Konačno, diskusija dobivenih rezultata nalazi se u zaključku.

7.2 LHC i detektor ALICE

LHC je trenutno najveći svjetski ubrzivač čestica. Sagrađen je od strane Europske organizacije za nuklearna istraživanja (CERN) s namjerom testiranja raznih predviđanja u fizici elementarnih čestica, primarno postojanje Higgsovog bozona. LHC je posljednji dodatak CERN-ovom ubrzivačkom kompleksu. Sa stabilnim je radom počeo u ožujku 2010. godine. Najvažnije otkriće na LHC-u je potvrda postojanja Higgsovog bozona s masom od otprilike $125\,\mathrm{GeV}/c^2$. LHC se nalazi na francusko-švicarskoj granici blizu Ženeve. Ima opseg 27 km te se nalazi u prosjeku na dubini od 100 m ispod zemlje.

Prije ulaska na LHC protoni se prvo ubrzavaju kroz Linac4, a nakon toga se postepeno ubrzavaju kroz redom Proton Synchrotron Booster (PSB), Proton Synchrotron (PS) i Super Proton Synchrotron (SPS). Tijekom svakog koraka energija protona se povećava dok na kraju protoni u LHC-u ne dostignu maksimalnu energiju od 6.5 TeV po snopu. Veliki dipolni supravodljivi magneti se koriste za održavanje čestica na njihovoj kružnoj putanji. Kvadrupolni i ostali višepolni magneti se koriste za fokusiranje snopova s obzirom da pozitivno nabijeni protoni imaju tendenciju udaljavanja jedan od drugoga. Ubrzavanje se događa unutar 16 radiofrekventnih šupljina. Postoje četiri velika ekperimenta na LHC-u, od kojih su najveći ATLAS i CMS. Radi se o velikim višenamjenskim detektorima koji se bave istraživanjima u fizici elemetarnih čestica. Primjer takvih istraživanja je potraga za česticama koje bi mogle tvoriti tamnu materiju. ALICE eksperiment je specijaliziran za fiziku sudara teških iona, uz poseban naglasak na istraživanje kvarkovsko-gluonske plazme, stanja tvari za koje se pretpostavlja da je vladalo u ranom svemiru. LHCb istražuje male razlike između materije i antimaterije analiziranjem lijepih kvarkova. Također, na LHC-u se nalaze tri dodatna manja eksperimenta. TOTEM dijeli iterakcijsku točku s CMS-om i bavi se mjerenjima ukupnog udarnog presjeka i elastičnim raspršenjima. MoEDAL dijeli interakcijsku točku s LHCb-om i bavi se potragom za magnetskim monopolima i drugim visokoionizirajućim česticama. Konačno, LHCf dijeli interakcijsku točku s ATLAS-om te se



SLIKA 7.2: Detektor ALICE.

bavi mjerenjima koje pomažu što boljoj simulaciji kozmičkih zraka u laboratorijskim uvjetima.

Detektor ALICE je specijaliziran je za analizu sudara teških iona. U takvim sudarima nastaje kvarkovsko-gluonska plazma (QGP), juha slabo interagirajućih kvarkova i gluona. Analiza kvarkovsko-gluonske plazme je izrazito važna za razumijevanje kvantne kromodinamike te uvjeta koji su potencijalno postojali u ranim razdobljima svemira. Detektor ALICE je opremljen sustavima koji istovremeno mogu pratiti putanje tisuća čestica nastalih u sudarima teških iona. Također, ALICE posjeduje veliku moć identifikacije vrsta čestica s obzirom na druge detektore. Dvadeset manjih podsustava surađuju zajedno kako bi se to omogućilo. Njih okružuje veliki L3 elektromagnet koji proizvodi polje od 0.5 T s ciljem zakretanja nabijenih čestica, što omogućuje mjerenje potrebnih kinematičkih svojstava. Najvažnija komponenta detektora ALICE je TPC (Time Projection Chamber) koji koristi radni obujam od čak 90 m^3 ispunjen plinskom mješavinom kako bi uspješno izmjerio putanje velikog broja čestica. Visoko-energetska

čestica ionizira plin prolaskom kroz njega. Novonastali slobodni elektroni zbog narinutog električnog polja putuju prema ravnini za očitavanje. Radijalna komponenta položaja čestice se odredi prema mjestu na kojem elektroni udare o ravninu. Longitudinalna komponenta se odredi mjerenjem vremena putovanja elektrona pošto je driftna brzina elektrona u plinskoj mješavini poznata. Informacija o vrsti čestice može se dobiti mjerenjem specifičnog gubitka energije prolaskom kroz plin koristeći Bethe-Blochovu formulu.

Od ostalih sustava izdvajamo Inner Tracking System (ITS) koji je također krucijalan za praćenje putanja čestica te mjerenje točnog položaja sudara. ITS se sastoji od 3 vrste silicijskih detektora koji su redom, radijalno prema van, Silicon Pixel Detector (SPD), Silicon Drift Detector (SDD) te Silicon Strip Detector (SSD).

7.3 Strojno učenje

Strojno učenje je područje istraživanja u računalnoj znanosti koje za cilj ima razviti metode i modele koji bi mogli riješiti neki zadatak bez da im se eskplicitno kaže što i kako napraviti. Model bi trebao iz prikupljenih podataka naučiti koja su svojstva podataka najvažnija te iz njih uspjeti predvidjeti buduća ponašanja. Za razliku od čistih optimizacijskih algoritama, strojno učenje želi razviti tehnike koje mogu uspješno generalizirati naučena svojstva i značajke na neke nove, dosad neviđene primjere. Tehnike strojnog učenja mogu biti vrlo uspješne u problemima gdje ljudsko znanje nije potpuno (znanost), ili je potrebno znanje prisutno, ali teško objašnjivo (prepoznavanje govora ili računalni vid).

Postoje tri osnovna pristupa u strojnom učenju: nadzirano učenje, nenadzirano učenje i učenje s potkrepljenjem. Nadzirano učenje pretpostavlja da imamo primjere za koji su poznati željeni rezultati. Cilj ovog pristupa je naučiti općenito pravilo koje mapira primjere i željene rezultate te naučeno pravilo primijeniti na nekim novim podacima za koje rezultat nije poznat. Problemi koji se susreću u nadziranom učenju mogu se ugrubo podijeliti u dvije katogorije, klasifikacijske i regresijske probleme. U nenadziranom učenju željeni rezultati nisu poznati. Cilj ovakvog učenja je pronaći skrivene uzorke i korelacije u podacima. Konačno, u učenju s potkrepljenjem, algoritam međudjeluje s okolišem u kojem želi postići neki definiran cilj. U ovom pristupu okoliš daje algoritmu potrebnu povratnu informaciju.

Strojno učenje je područje koje se vrlo brzo razvija. Svakim danom se predlaže sve više novih tehnika i algoritama. Najjednostavniji primjer algoritma u strojnom učenju

je linearna regresija. Od ostalih najćešće korištenih algoritama izdvajamo logističku regresiju, Support Vector Machine (SVM), K-Means grupiranje, kNN, stabla odluke i neuralne mreže.

Neuralne mreže se temelje na međusobno povezanim objektima koji se nazivaju neuroni. Oni ugrubo modeliraju neurone u ljudskom mozgu. Matematički opis neurona možemo vidjeti u sljedećoj jednadžbi:

$$y = f\left(\sum_i w_i x_i + b\right). \tag{7.1}$$

Neuron prima neki ulazni vektor te izračunava težinsku sumu komponenti tog vektora. Vrijednosti težina se prilagođavaju tijekom učenja. Izlazna vrijednost neurona je neka nelinearna aktivacijska funkcija primijenjena na izračunatu sumu. Primjeri najčešćih aktivacijskih funkcija su ReLU i *sigmoid*. Ovako opisani neuroni mogu se međusobno spajati na način da izlazna vrijednost jednog neurona bude ulazna vrijednost u drugi neuron. Na taj se način može konstruirati proizvoljno komplicirana mreža povezanih neurona. Takvi objekti se nazivaju neuralnim mrežama.

Prema načinu međusobnog povezivanja neurona razlikujemo razne arhitekture neuralnih mreža. Najopćenitija arhitektura neuralne mreže je takozvana gusto povezana neuralna mreža. U takvoj mreži su svi neuroni u određenom sloju povezani sa svim ostalim neuronima. Iako posjeduju kapacitet za rješavanje najkompleksnijih problema, za korištenje gusto povezane mreže u praksi često nema dovoljno podataka te teško postiže konvergenciju. U svrhu stabilnijeg i bržeg učenja predložene su razne arhitekture neuralnih mreža. Konvolucijska neuralna mreža se pokazala kao jedna od najuspješnijih arhitektura, posebice u području računalnog vida. U konvolucijskim neuralnim mrežama neuroni su ograničeni mali broj lokalnih veza te se uz to težinske vrijednosti dijele među neuronima. Na taj način se se značajno smanjuje ukupni broj slobodnih parametara.

Proces učenja neuralne mreže i optimiziranje vrijednosti njezinih parametara se vrši korištenjem algoritma povratne propagacije u kojem se minimizira vrijednost neke *loss* funkcije na dostupnim primjerima. Za regresijske probleme se najčešće koristi srednje kvadratno odstupanje, dok se za klasifikacijske probleme nejčešće koristi srednja unakrsna entropija.

Uz primjene u nadziranom učenju, neuralne mreže se često koriste u nenadziranom učenju. Primjerice, *autoencoder* je neuralna mreža koja želi aproksimirati funkciju

identiteta. Cilj takve mreže je naći najvažnije značajke koje su potrebne za rekonstrukciju ulaznih podataka. Također, neuralne mreže se koriste i u generativnim modelima kao što su Variational autoencoder (VAE) ili Generative Adversarial Network (GAN) u kojima se želi iz nekog nasumičnog šuma generirati primjere koji prate gustoću vjerojatnosti dostupnih podataka te ih na taj način imitiraju.

7.4 Metodologija

Korištena metodologija se može podijeliti na dva dijela. U prvom dijelu je opisana metoda razvijenja diskriminante kvarkovskih i gluonskih mlazova. U drugom dijelu dan je kratak opis 2NN algoritma predstavljenog u [59].

7.4.1 Diskriminacija kvarkovskih i gluonskih mlazova

Za potrebe diskriminacije kvarkovskih i gluonskih mlazova simuliran je veliki broj događaja pomoću Pythia8 Monte Carlo generatora. Za modeliranje efekata ALICE detektora korišten je paket GEANT4. Algoritam anti-k_T s parametrom $R = 0.4$ je iskorišten u svrhu identificiranja mlazova. Podaci su podijeljeni prema intervalima ukupnog transverzalnog impulsa mlaza. Tablica 7.1 daje pregled generiranih podataka. U simuliranim događajima dostupna je informacija o vrsti početnog partona mlaza. Razvijeni modeli će se optimizirati pomoću ovih informacija. Nakon učenja i testiranja modeli se mogu iskoristiti na stvarnim podacima u kojima ta informacija nije dostupna. Podaci su spremljeni kao lista četveroimpulsa svih čestica koje konstituiraju mlaz.

Reprezentacije podataka

Korištene su tri različite reprezentacije podataka. Lista kinematičkih varijablih pojedinih čestica, generalizirane uglatosti (generalized angularities) i slike mlazova. Liste kinematičkih varijabli su predstavljene tenzorom veličine (50×4). Prva os ide po različitim česticama unutar mlaza, a druga os po različitim kinematičkim varijablama. Korištena su dva seta po četiri kinematičke varijable. Prvi set sastoji se od četveroimpulsa. Drugi set sastoji se od frakcije ukupnog transverzalnog momenta, translatiranog rapiditeta, translatiranog azimutalnog kuta i mase ćestice. Navedene varijable su dobivene na sljedeći način:

TABLICA 7.1: Broj mlazova po p_T intervalima.

p_T	N	N_q	N_g	N_{train}	N_{test}
0-25 GeV	5352459	509416 (9.5 %)	4843043 (90.5%)	4817213	1338114
25-50 GeV	5059608	872273 (17.2%)	4187335 (82.8%)	3794706	1264902
50-75 GeV	4774522	1011871 (21.2%)	3762651 (78.8%)	3580891	1193630
75-100 GeV	3951685	840789 (21.3%)	3110896 (78.7%)	2963763	987921
100-125 GeV	3051618	661499 (21.7%)	2390119 (78.3%)	2288713	762904
125-150 GeV	2080107	487988 (23.5%)	1592119 (76.5%)	1560080	520026
150-175 GeV	1246585	335412 (26.9%)	911173 (73.1%)	934938	311646
175-200 GeV	681821	217856 (32.0%)	463965 (68.0%)	511365	170455
200-250 GeV	536163	211876 (39.5%)	324287 (60.5%)	402122	134040

$$z = \frac{p_T}{\sum_{i \in jet} p_{T,i}}, \tag{7.2}$$

$$\Delta y = y - y_{jet} = \frac{1}{2} \ln \frac{E + p_z}{E - p_z} - \frac{1}{2} \ln \frac{E_{jet} + p_{z,jet}}{E_{jet} - p_{z,jet}}, \tag{7.3}$$

$$\Delta \phi = \phi - \phi_{jet} = \arccos \frac{p_x p_{x,jet} + p_y p_{y,jet}}{p_T p_{T,jet}}, \tag{7.4}$$

$$m = \sqrt{E^2 - p_x^2 - p_y^2 - p_z^2}. \tag{7.5}$$

Drugi način reprezentacije podataka su globalne varijable koje opisuju mlazove nazvane generalizirane uglatosti te su definiraju se na sljedeći način:

$$\lambda_\beta^\kappa = \sum_{i \in jet} z_i^\kappa \theta_i^\beta, \tag{7.6}$$

pri čemu je $\theta = \sqrt{\Delta y^2 + \Delta \phi^2}/R$ te indeks i predstavlja redni broj čestice u mlazu. U ovom istraživanju se koriste sve varijable s kombinacijom indeksa $\kappa = 0, 1, 2$ i $\beta = 0, 1, 2$. Zadnji način na koji su podaci predstavljeni su slike mlazova. Takva reprezentacija koristi tenzore veličine $(32 \times 32 \times N)$, gdje prve dvije osi redom predstavljaju diskretizirani $(\Delta \phi, \Delta y)$ prostor, dok zadnja os predstavlja broj korištenih kanala.

Za vrijednosti piksela u pojedinim kanalima korištene su energije čestica E, iznos ukupnog impulsa čestice p, frakcija ukupnog impulsa mlaza z, frakcija frakcija ukupnog impulsa mlaza ako čestica ima masu manju od 0.3 GeV/c^2, z_π, frakcija ukupnog impulsa mlaza ako čestica ima masu u rasponu od 0.3 GeV/c^2 do 0.6 GeV/c^2, z_κ te frakcija ukupnog impulsa mlaza ako čestica ima masu veću od 0.6 GeV/c^2, z_p. Tri različite vrste slika mlazova su konstruirane. Prva se sastoji od E i p kanala, a druga se sastoji od slika samo sa z kanalom. Posljednja ima kombinaciju od 3 kanala, (z_π, z_κ, z_p).

Modeli neuralnih mreža

Za svrhu diskriminacije kvarkovskih i gluonskih mlazova i njihove međusobne usporedbe korišteno je ukupno 9 različitih modela. Na generaliziranim uglatostima korištene su logistička regresija i ANG model. ANG model je gusto povezana neuralna mreža koja prima vektor veličine 9 i kao izlaznu vrijednost daje realan broj između 0 i 1. ANG model sadrži 3 skrivena sloja sa 100 neurona koji su aktivirani ReLU funkcijom.

Modeli koji koriste liste kinematičkih varijabli su EFN, PFN, PFN-ID te pPFN. Arhitekture ovih modela su preuzete iz [56]. pPFN koristi četveroimpulse kao kinematičke varijable. EFN i PFN koriste $(z, \Delta y, \Delta\phi)$, dok PFN-ID uz navedene varijable koristi i mase pojedinih čestica.

Slike mlazova se koriste kao ulazni tenzori konvolucijskih neuralnih mreža CNN, zCNN i zCNN-ID. CNN koristi slike mlazova s (E, p) kanalima. zCNN koristi slike sa z-kanalom dok zCNN-ID uzima (z_π, z_κ, z_p) slike mlazova. Arhitektura svake od navedenih mreža sastoji se od 3 konvolucijska sloja uparena s MaxPooling slojevima. Broj filtera u konvolucijskim slojevima je redom 32, 54, i 128 te koriste veličinu filtera 3×3. Nakon toga slijede tri skrivena gusto povezana sloja aktivirana ReLU funckijom s veličinom od, redom, 100, 50 i 10 neurona. Zadnji sloj se sastoji od jednog neurona aktiviranog funkcijom *sigmoid*.

Modeli su trenirani korištenjem algoritma AdaM. Tijekom treniranja podaci su balansirani tako da u svakoj epohi postoji jednaki broj mlazova označenih kao kvarkovski i mlazova označenih kao gluonski. Za usporedbu i evaluaciju modela korištena je površina ispod ROC krivulje (AUC).

7.4.2 2NN algoritam

2NN algoritam se temelji na Neyman-Pearsonovoj lemi koja konstatira da je optimalni diskriminator između dviju slučajnih varijabli njihov *likelihood* omjer. To nam

omogućuje da, u slučaju da smo opremljeni optimalnim klasifikatorom i uz pretpostavku da je gustoća vjerojatnosti jedne od te dvije slučajne varijable poznata, izračunamo gustoću vjerojatnosti po kojoj se ponaša preostala slučajna varijabla.

Uzmimo npr. da imamo uzorak vektora $\mathbf{x}$ koji prate neku nepoznatu višedimenzionalnu gustoću vjerojatnosti $p_{\text{real}}(\mathbf{x})$ koju nam je cilj odrediti. Također, imamo i uzorak vektora $\mathbf{x}$ koji prate poznatu gustoću vjerojatnosti $p_{\text{test}}(\mathbf{x})$. Ako je kao klasifikator između tih dvaju uzoraka korištena neuralna mreža koja daje izlaznu vrijednost $C_{NN}(\mathbf{x})$ trenirana tako da se minimizira *binary crossentropy loss*, tad se gustoća vjerojatnosti $p_{\text{real}}(\mathbf{x})$ može odrediti na sljedeći način:

$$p_{\text{real}}(\mathbf{x}) = \frac{C_{NN}(\mathbf{x})}{1 - C_{NN}(\mathbf{x})} p_{\text{test}}(\mathbf{x}) \,. \tag{7.7}$$

Napomenimo da je gornja jednadžba egzaktna samo u slučaju da je neuralna mreža optimalan klasifikator. Međutim, u praksi to nije slučaj te izračunata gustoća vjerojatnosti neće odgovarati željenoj gustoći vjerojatnosti. Iako ta nova gustoća vjerojatnosti nije željena, ona je u prosjeku bliža $p_{\text{real}}(\mathbf{x})$ nego što je to $p_{\text{test}}(\mathbf{x})$ bila. Ova spoznaja omogućava iterativno korištenje jednadžbe 7.7 tako što se u svakom idućem koraku $p_{\text{test}}(\mathbf{x})$ zamjenjuje s gustoćom vjerojatnosti dobivenom iz prošle iteracije.

Za korištenje 2NN algoritma mlaz se modelira kao kolekcija uzastopnih neovisnih $1 \rightarrow 2$ raspada. Svaki raspad je ograničen očuvanjima četveroimpulsa. Svaki raspad ima točno 4 stupnja slobode. Za varijable koje opisuju raspad su odabrane mase produkata raspada i kutovi u sustavu centra mase. Ako pretpostavimo da su kutovi uniformno raspoređeni, vjerojatnost pojavljivanja određenog mlaza $\mathbf{x}$ je:

$$p\left(\mathbf{x}\right) = \prod_i p(m_1^i, m_2^i|M) p_\theta(\theta^i) p_\phi(\phi^i), \tag{7.8}$$

pri čemu je M masa čestice koja se raspada. Mase m_1 i m_2 označavaju mase novonastalih produkata. Vjerojatnost $p(m_1^i, m_2^i|M)$ označava vjerojatnost da će se čestica mase M raspasti u dvije čestice masa m_1 i m_2. Algoritam 2NN je konstruiran da iz uzorka konačnih stanja mlazova rekonstuira tražene vjerojatnosti prema sljedećoj generalizaciji jednadžbe 7.7:

$$\sum_i \ln p(m_1^i, m_2^i|M) = \ln C_{NN}(\vec{x}) - \ln(1 - C_{NN}(\vec{x})) + \sum_i \ln q(m_1^i, m_2^i|M), \tag{7.9}$$

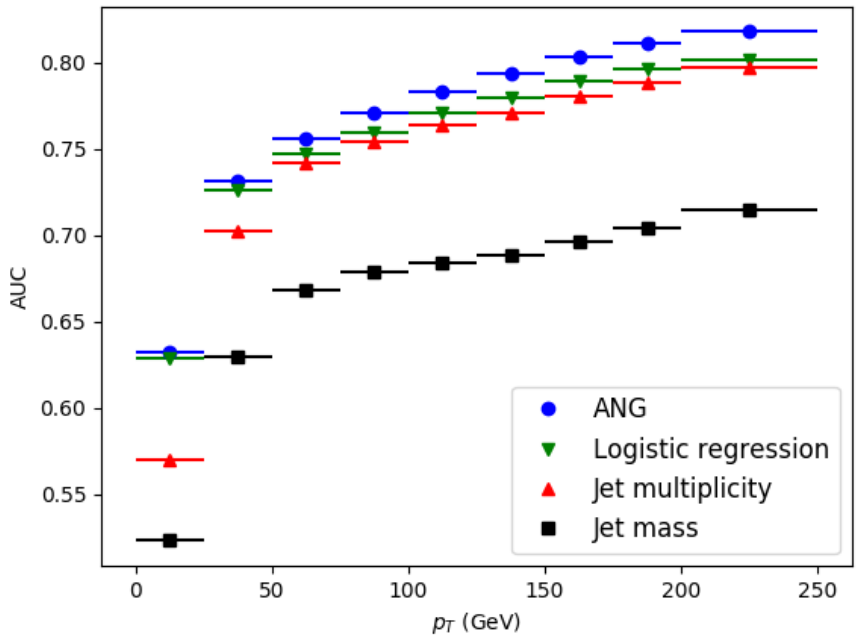
pri čemu je $q(m_1^i, m_2^i|M)$ neka poznata pretpostavljena vjerojatnost raspada, $\mathbf{x}$ predstavlja konačna stanja mlazova generiranih prema pretpostavljenim vjerojatnostima i $C_{NN}(\mathbf{x})$ izlazna vrijednost neuralne mreže koja je trenirana da razlikuje stvarne dostupne podatke od mlazova generiranih prema vjerojatnosi q. Kako je u pitanju visokodimenzionalni linearni sustav jednadžbi, vjerojatnosti $p(m_1^i, m_2^i|M)$ su modelirane neuralnom mrežom f koja za ulazne vrijednosti prima trojku realnih brojeva (m_1, m_2, M). Cilj 2NN algoritma je optimizirati neuralnu mrežu f kako bi što bolje opisivala stvarne gustoće vjerojatnosti. Daljnji zahtjevi za neuralnu mrežu f kao i način njene optimizacije su opisani u [59]. Također, predstavljen je jednostavni fizikalni sustav na kojem je testiran 2NN algoritam.

7.5 Rezultati

U ovom poglavlju predstavljeni su rezultati modela razvijenih za svrhu razlikovanja kvarkovskih i gluonskih mlazova. Uz to, predstavljen je i kratak pregled rezultata testiranja 2NN algoritma na jednostavnom fizikalnom sustavu.

7.5.1 Model za kvarkovsko-gluonsku diskriminaciju

Modeli su evaluirani koristeći ROC krivulje. Kao kvantitativna skalarna vrijednost koje opisuje performase modele korištena je vrijednost površine ispod ROC krivulje (AUC). Što je veća AUC vrijednost to model može bolje raspoznati razliku među kvarkovskim i gluonskim mlazovima. Radi bolje preglednosti, rezultati modela opisanih u prošlom poglavlju su podijeljeni u tri skupine. Na slici 7.3 prikazane su AUC vrijednosti ANG modela i logističke regresije provedene na generaliziranim uglatostima za različite intervale transverzalnog impulsa mlaza. Uspoređene su s AUC vrijednostima za diskriminatore koji su temeljeni na multiplicitetu mlaza i masi mlaza. Iz priloženog može se primjetiti kako ANG model pokazuje najbolje performanse na svim p_T intervalima. ANG model i logistička regresija puno bolje razlikuju vrstu mlaza nego jednostavni rez na varijablu mase mlaza. Također, navedeni modeli su značajno bolji od reza na multiplicitet na niskim vrijednostima transverzalnog impulsa. Iz ovoga se može zaključiti da je multiplicitet kao diskriminativna varijabla pogodnija za korištenje na mlazovima visokog ukupnog transverzalnog impulsa. Na slici 7.4 vidimo relativne važnosti različitih generaliziranih uglatosti na predviđanja ANG modela. Može se primijetiti kako je najvažnija generalizirana uglatost λ_1^1 daleko najvažnija za mlazove niskog ukupnog



SLIKA 7.3: AUC vrijednosti ANG modela i logističke regresije na generaliziranim uglatostima za različite p_T intervale.

transverzalnog impulsa. Ta varijabla se može interpretirati kao širina mlaza. Ovaj rezultat se slaže s teoretskim spoznajama koje konstatiraju da gluoni emitiraju partone u širim kutovima u odnosu na kvarkovske mlazove. Širina mlaza se pokazala kao najvažnija i na mlazovima visokog ukupnog transverzalnog impulsa, međutim i ostale varijable također pokazuju značajnu važnost. AUC vrijednosti za različite konvolucijske modele su prikazane na slici 7.5. Kao najbolji konvolucijski model se pokazao zCNN-ID koji koristi (z_π, z_κ, z_p)-kanalne slike mlazova. Konvolucijski modeli pokazuju unapređenje u odnosu na ANG model. Takvo ponašanje je očekivano s obzirom da slike mlazova sadrže više informacija u odnosu na samo nekolicinu globalnih varijabli koje koristi ANG model. Nedostatak takvih modela je velika količina potrebnih podataka i relativno dugo vrijeme potrebno za konvergenciju modela. Dodatni zaključak koji se može donijeti je taj da informacija o vrsti čestice koja konstituira mlaz može unaprijediti performanse modela.

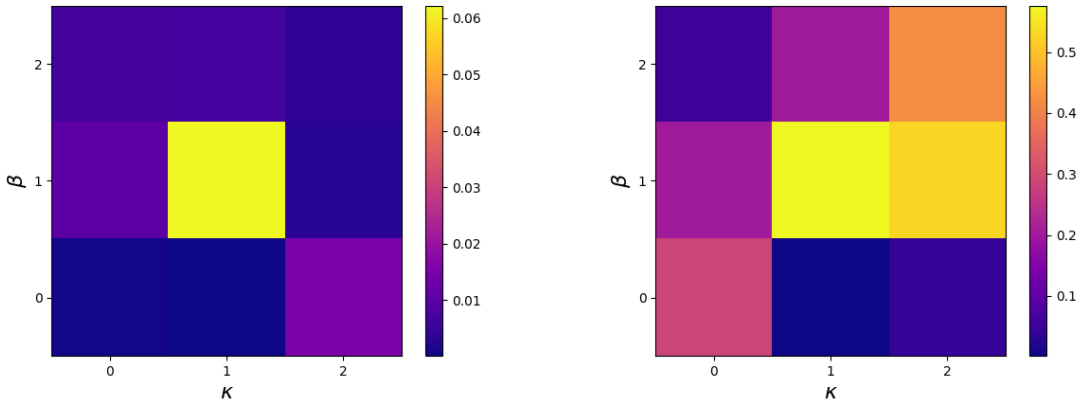
Konačno, slika 7.6 predstavlja AUC vrijednosti za EFN, PFN, pPFN i PFN-ID modele. Može se jasno vidjeti kako se modeli PFN i PFN-ID svojim performansama izdvajaju od EFN i pPFN modela. EFN model je konstruiran tako da je neosjetljiv na infracrvene divergencije koje se pojavljuju u kvantnoj kromodinamici. Ostali modeli nisu

infracrveno sigurni. Priloženi rezultati potvrđuju da infracrveno nesigurne varijable posjeduju neku informaciju bitnu za razlikovanje vrsta mlazova. PFN-ID pokazuje najbolje performanse u usporedbi sa svim predstavljenim modelima. Još jednom je potvrđeno da je informacija o vrsti čestica u mlazu važna za razlikovanje kvarkovskih i gluonskih mlazova. PFN-ID se pokazao univerzalno najboljim u svim p_T-intervalima. U tablici 7.2 je prikazan pregled AUC vrijednosti za sve modele izvrednjene u svim p_T intervalima. Nepouzdanosti AUC vrijednosti su procijenjene ponovljenim treniranjem modela te se njihove vrijednosti kreću u rasponu od 0.001 i 0.002.

Nadalje, provedena je analiza koliku čistoću kvarkovskog uzorka možemo postići pomoću PFN-ID modela u slučaju njegovog korištenja na pravim ALICE podacima. Ako želimo zadržati barem 20% kvarkovskih mlazova najveća čistoća uzorka ovisi o p_T intervalu koji želimo gledati. Vrijednost čistoće se tada kreće od otprilike 90% u intervalu transverzalnog impulsa [200-250 GeV] do malo manje od 50% za interval od [25-50 GeV]. U intervalu ukupnog transverzalnog impulsa [0-25 GeV] je nemoguće izolirati uzorak kvarkovskih mlazova korisne čistoće pomoću PFN-ID modela.

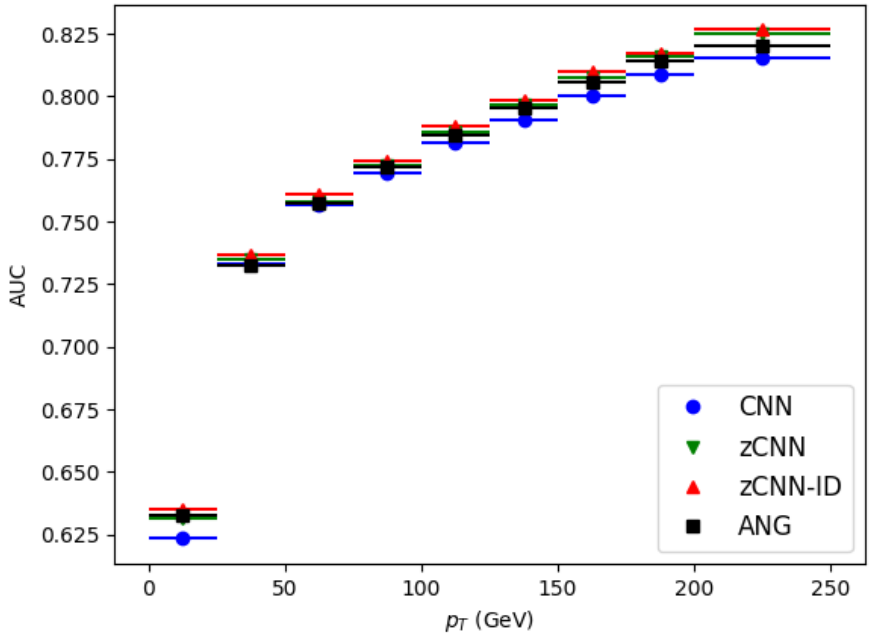
7.5.2 Performanse 2NN algoritma

2NN algoritam je testiran na jednostavnom fizikalnom sustavu koji je detaljno opisan u [59]. U tom fizikalnom sustavu mlazovi nastaju uzastopnim kaskadnim raspadima počevši od čestice mase $M = 25.0$. Svi mogući kanali raspada u tom sustavu zajedno s



(A) $p_T \in [0 - 25\,\text{GeV}]$ (B) $p_T \in [200 - 250\,\text{GeV}]$

SLIKA 7.4: Relativne važnosti generaliziranih uglatosti (κ, β) korištenih ANG modelom za mlazove u najnižem i najvišem p_T intervalu.



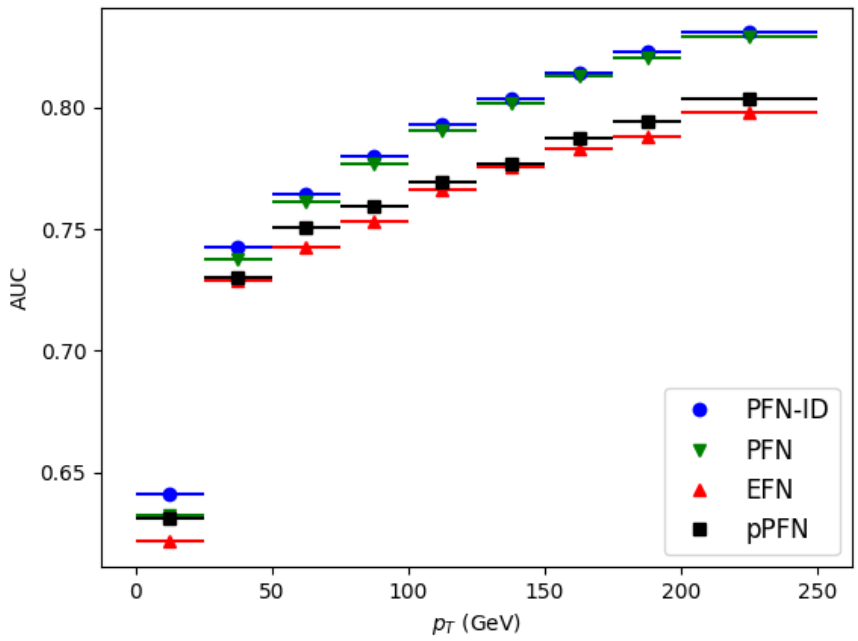
SLIKA 7.5: p_T ovisnost AUC vrijednosti za različite konvolucijske modele.

pripadnim vjerojatnostima su prikazani u tablici 7.4. Zadatak 2NN algoritma je pronaći kanale raspada i pripadne vjerojatnosti tih raspada samo iz konačnih čestica u mlazovima. Iako su vjerojatnosti raspada u fizikalnom sustavu diskretne i svaka čestica se može raspasti u najviše 3 kanala, ta informacija nije *a priori* poznata. 2NN algoritam je konstruiran tako da pronađe bilo koju moguću dvodimenzionalnu vjerojatnosnu distribuciju raspada $p(m_1, m_2 \mid M)$ te je ograničen samo kinematičkim zakonima očuvanja i pretpostavkom da su raspadi međusobno neovisni. Tablica 7.3 prikazuje rekonstruirane kanale raspada 2NN algoritmom. Možemo vidjeti kako se rekonstruirani kanali raspada vrlo dobro poklapaju s kanalima raspada u fizikalnom sustavu. Identificirane su čak i čestice koje se ne pojavljuju u konačnom stanju. Bolja rekonstukcija se može primijetiti za čestice većih masa u odnosu na čestice manjih masa. Detaljniji pregled dobivenih rezultata i evaluacija 2NN algoritma se može pronaći u [59].

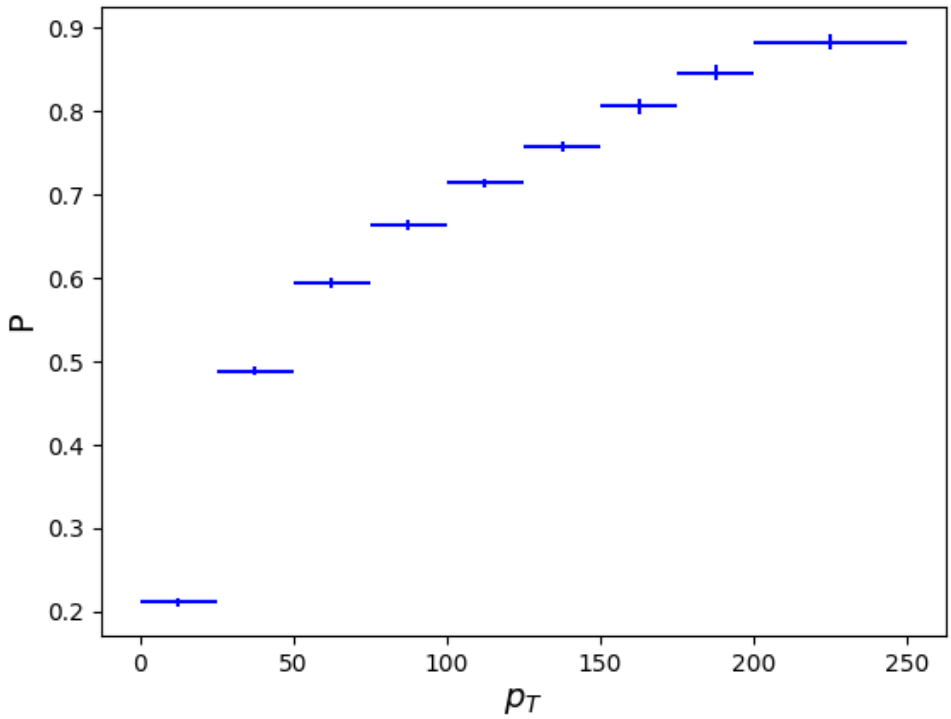
7.6 Zaključak

Ovo istraživanje predstavlja modele strojnog učenja i tehnike koje se koriste za razlikovanje mlazova nastalih od kvarkova ili gluona, a primjenjeni su za korištenje na podacima prikupljenim eksperimentom ALICE u CERN-u. To se postiže korištenjem velike količine Monte Carlo simuliranih podataka za obuku i validaciju predloženih modela. Izvršena je opsežna analiza modela i predviđanja. Model koji pokazuje najbolje performanse je PFN-ID, posebno dizajnirana neuralna mreža koja koristi kinematičke veličine čestica zajedno s informacijama o njihovom okusu. PFN-ID je univerzalno najbolji model budući da je superiorniji od svakog drugog modela u cijelom rasponu promatranih ukupnih momenta mlazova.

Iz analize se može izvući nekoliko zaključaka. Varijable na razini čestica treba prvo prethodno obraditi kako bi se smanjila količina simetrije u podacima. Modeli koji koriste neobrađene četverovektore količina gibanja četiri pokazuju se gorim od modela koji koriste udjele ukupnih transverzalnih momenta, brzine i azimutalne kutove u odnosu na os mlaza. Drugi zaključak je da je okus na razini čestica vrlo važan za



SLIKA 7.6: p_T ovisnost AUC vrijednosti za EFN, PFN, pPFN i PFN-ID model.



SLIKA 7.7: Ovisnost čistoće uzorka kvarkovskih mlazova u ovisnosti o p_T intervalu za efikasnost od 20 %

diskriminantu koja razlučuje kvarkovske i gluonskie mlazove. Svaki model koji koristi ove podatke ima bolju izvedbu od svog analognog modela koji ih ne koristi. Ovo je vrlo korisno za primjenu na podacima koje prikuplja ALICE detektor, budući da taj detektor ima vrlo dobru sposobnost identifikacije čestica. Modeli koji koriste globalne varijable mlaza pokazuju nešto goru izvedbu od modela koji koriste informacije na razini čestica (uključujući identitet čestica). To je za očekivati budući da na razini čestica, podaci sadrže mnogo više informacija koje dovoljno složeni modeli mogu iskoristiti. Međutim, takvi modeli mogu biti računalno zahtjevni i teški za obuku.

Diskriminacijska moć svakog od treniranih modela raste s ukupnim transverzalnim momentom mlaza p_T, bez obzira na arhitekturu i prikaz podataka. To znači da su uzorci kvarkovskih i gluonskih mlazova međusobno slabije odvojivi pri niskim p_T. Analiza važnosti značajki pokazuje nam koje su varijable značajne u procesu odlučivanja modela. Najvažnije globalne varijable su širina mlaza i broj čestica u mlazu. Širina mlaza je dominantno važna za modele trenirane na mlazovima s niskim p_T. Važnost ostalih varijabli raste s povećanjem p_T mlazova. Važnosti značajki izvedenih

TABLICA 7.2: Pregled AUC vrijednosti za različite modele u svim p_T intervalima.

p_T	LR	ANG	CNN	zCNN	zCNN-ID
0-25 GeV	0.630	0.632	0.623	0.631	0.635
25-50 GeV	0.727	0.732	0.733	0.734	0.737
50-75 GeV	0.748	0.757	0.757	0.758	0.761
75-100 GeV	0.759	0.772	0.769	0.772	0.774
100-125 GeV	0.771	0.784	0.782	0.786	0.788
125-150 GeV	0.780	0.795	0.791	0.796	0.798
150-175 GeV	0.790	0.805	0.800	0.808	0.810
175-200 GeV	0.797	0.814	808	0.816	0.817
200-250 GeV	0.802	820	0.816	0.825	0.827
p_T	EFN	pPFN	PFN	PFN-ID	
0-25 GeV	0.622	0.631	0.632	0.641	
25-50 GeV	0.729	0.730	0.737	0.743	
50-75 GeV	0.743	0.751	0.761	0.764	
75-100 GeV	0.753	0.759	0.777	0.780	
100-125 GeV	0.766	0.770	0.790	0.793	
125-150 GeV	0.775	0.777	0.802	0.804	
150-175 GeV	0.783	0.788	0.813	0.814	
175-200 GeV	0.788	0.794	820	0.823	
200-250 GeV	0.798	804	0.829	0.831	

na slikama mlazova pokazuje područje oko osi mlaza koje je najosjetljivije na odluke modela. Ova efektivna površina se smanjuje s povećanjem mlaza p_T.

Nadalje, razvili smo metodu za izračunavanje temeljnih distribucija vjerojatnosti raspada čestica, koristeći samo podatke koji se mogu prikupiti u detektoru. Prvo, generiramo umjetni fizikalni sustav temeljen na dijelu QCD procesa fragmentacije. Zatim predstavljamo ključni dio metode: algoritam $2NN$. Algoritam točno predviđa

TABLICA 7.3: Dozvoljeni diskretni raspadi s pripadnim vjerojatnostiima u modelu fizikalnog sustava.

Particle	A	B	C	D	E
Mass	0.1	0.6	1.3	1.9	4.4
p/channel	1 A	0.7 B 0.3 A+A	1 C	0.4 D 0.3 A+A 0.3 A+C	0.6 C+C 0.4 E
Particle	F	G	H	I	J
Mass	6.1	8.4	14.2	18.1	25
p/channel	0.5 A+A 0.5 B+C	0.9 B+B 0.1 A+F	0.6 D+D 0.25 D+E 0.15 E+F	1 F+G	0.5 F+I 0.4 G+H 0.1 E+E

particle mass	25		25		25	
decay masses	18.1	6.1	14.2	8.4	4.4	4.4
reconstructed masses	18.1 ± 0.5	6.1 ± 0.5	14.0 ± 0.7	8.4 ± 0.7	4.8 ± 0.2	4.6 ± 0.2
channel probability	0.5		0.4		0.1	
reconstructed probability	0.48		0.47		0.05	
particle mass	18.1		14.2		14.2	
decay masses	8.4	6.1	6.1	4.4	4.4	1.9
reconstructed masses	8.2 ± 0.6	5.9 ± 0.4	6.2 ± 0.3	4.4 ± 0.4	4.3 ± 0.3	2.7 ± 0.7
channel probability	1		0.15		0.25	
reconstructed probability	1		0.05		0.06	
particle mass	14.2		1.9		1.9	
decay masses	1.9	1.9	1.3	0.1	no decay	
reconstructed masses	2.4 ± 0.5	2.9 ± 0.7	1.27 ± 0.09	0.14 ± 0.09	no decay	
channel probability	0.6		0.3		0.4	
reconstructed probability	0.89		0.29		0.71	
particle mass	14.2		1.9		18.1	
decay masses	1.9	1.9	1.3	0.1	no decay	
reconstructed masses	2.4 ± 0.5	2.9 ± 0.7	1.27 ± 0.09	0.14 ± 0.09	no decay	
channel probability	0.6		0.3		0.4	
reconstructed probability	0.89		0.29		0.71	

TABLICA 7.4: Značajke rekonstruiranih kanala raspada za čestice masa $M = 25$, 18.1, 14.2 and 1.9.

većinu skrivenih rezonantnih čestica, kao i njihove kanale raspada, koji se mogu pojaviti u evoluciji mlazova. Energetski spektri čestica u konačnom stanju također se mogu dosta točno reproducirati. Iako je testiran samo na spomenutom sustavu, vjerujemo da je algoritam dovoljno općenit kako bi se primijenio na stvarne fizikalne sustave, kao što su sudari visokoenergetskih čestica. Nadamo se da će se u budućnosti ova metoda pokazati korisnom u mjerenju fragmentacijskih funkcija kvarkova i gluona. Dodatno,

takva bi se metoda mogla upotrijebiti u potrazi za supersimetričnim česticama nepoznatih masa ili u mjerenju omjera grananja poznatih raspada.

Bibliography

[1] W.E.Lamb, "Fine structure of the Hydrogen Atom. III," *Physical Review*, vol. 85, 1952.

[2] E.Fermi, "Tentativo di una teoria dei raggi β," *La Ricerca Scientifica*, vol. 2, 1933.

[3] C. Wu, E. Ambler, R. Hayward, D. Hoppes, and R. Hudson, "Experimental test of parity conservation in beta decay," *Physical Review*, vol. 105, 1957.

[4] S. Glashow, "The renormalizability of vector meson interactions.," *Nucl. Phys.*, vol. 10, 1959.

[5] J. A. Salam, "Weak and electromagnetic interactions," *Nuovo Cimento*, vol. 11, 1959.

[6] S. Weinberg, "A model of leptons," *Physical Review Letters*, vol. 19, 1967.

[7] P. Higgs, "Broken symmetries and the masses of gauge bosons," *Physical Review Letters*, vol. 13, 1964.

[8] ATLAS Collaboration, "Measurements of Higgs boson production and couplings in the four-lepton channel in pp collisions at center-of-mass energies of 7 and 8 TeV with the ATLAS detector," *Physical Review D.*, vol. 91, 2015.

[9] (), [Online]. Available: https://en.wikipedia.org/wiki/File:Standard_Model_of_Elementary_Particles.svg.

[10] D. J. E. Callaway and A. Rahman, "Microcanonical ensemble formulation of lattice gauge theory," *Physical Review Letters*, vol. 49, 1982.

[11] (), [Online]. Available: https://www.hep.phy.cam.ac.uk/~chpotter/particleandnuclearphy Lecture_07_QCD.pdf.

[12] D. Gross and F. Wilczek, "Ultraviolet behavior of non-abelian gauge theories," *Physical Review Letters*, vol. 30, 1973.

[13] K. Adcox and et.al., "Formation of dense partonic matter in relativistic nucleus–nucleus collisions at rhic: Experimental evaluation by the phenix collaboration," *Nuclear Physics A*, vol. 757, 2005.

[14] B. Andersson, G. Gustafson, G. Ingelman, and T. Sjöstrand, "Parton fragmentation and string dynamics," *Physics Reports*, vol. 97, 1983.

[15] G. Altarelli and G. Parisi, "Asymptotic freedom in parton language"," *Nuclear Physics B*, vol. 126, 1977.

[16] (), [Online]. Available: https://ilmonteux.github.io/2018/10/15/jet-tagging-cnn.html.

[17] R. Atkin, "Review of jet reconstruction algorithms," *J. Phys.: Conf. Ser.*, vol. 645, 2015.

[18] CDF Collaboration, "Charged jet evolution and the underlying event in proton-antiproton collisions at 1.8 TeV," *Physical Review D*, vol. 65, 2002.

[19] M. Cacciari, G. P. Salam, and G. Soyez, "The anti-kt jet clustering algorithm," *JHEP*, vol. 063, Apr. 2018.

[20] J. M. Butterworth, A. R. Davison, M. Rubin, and G. P. Salam, "Jet substructure as a new Higgs-search channel at the Large Hadron Collider," *Physical Review D*, vol. 100, 2008.

[21] M. Dasgupta, A. Fregoso, S. Marzani, and A. Powling, "Jet substructure with analytical methods," *arXiv:1307.0013 [hep-ph]*, 2013.

[22] A. J. Larkoski, J. Thaler, and W. J. Waalewijn, "Gaining (mutual) information about quark/gluon discrimination," *JHEP*, vol. 11, 2014.

[23] J. Thaler and K. V. Tilburg, "Identifying boosted objects with n-subjettiness," *JHEP*, vol. 03, 2011.

[24] ATLAS Collaboration, "Measurement of the lund jet plane using charged particles in 13 tev proton-proton collisions with the atlas detector," *Phys. Rev. Lett.*, vol. 222002, 2020.

[25] F. A. Dreyer, G. P. Salam, and G. Soyez, "The lund jet plane," *JHEP*, vol. 064, Dec. 2018.

[26] A. Lifson, G. P. Salam, and G. Soyez, "Calculating the primary lund jet plane density," *JHEP*, vol. 064, Oct. 2020.

[27] ATLAS Collaboration, "Discrimination of light quark and gluon jets in pp collisions at $\sqrt{s}$ = 8 TeV with the atlas detector," *ATLAS-CONF-2016-034*, 2016.

[28] ——, "Measurement of the charged-particle multiplicity inside jets from $\sqrt{s}$ = 8 TeV *pp* collisions with the ATLAS detector," *Eur. Phys. J. C*, vol. 76, 2016.

[29] CMS Collaboration, "Performance of quark/gluon discrimination in 13 TeV data," *CMS-DP2016-070*, 2016.

[30] (), [Online]. Available: https://atlas.cern/about.

[31] (), [Online]. Available: https://home.cern/science/accelerators/accelerator-complex.

[32] B. J. Holzer, "Lattice design in high-energy particle accelerators," *Proceedings of the CAS-CERN Accelerator School*, 2014.

[33] LHCb Collaboration, "Identification of beauty and charm quark jets at lhcb," *Journal of Instrumentation*, vol. 10, 2015.

[34] L.G.Gagnon, "Searching for supersymmetry using deep learning with the atlas detector," Ph.D. dissertation, Montreal University, 2020.

[35] I. Jolliffe and J. Cadima, "Principal component analysis: A review and recent developments," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 374, 2016.

[36] R. Bellman, "A markovian decision process," *Journal of Mathematics and Mechanics*, vol. 6, 1957.

[37] D. Silver and *et.al.*, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," *arXiv:1712.01815*, 2017.

[38] (), [Online]. Available: https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm.

[39] (), [Online]. Available: http://help.pyramidanalytics.com/Content/Root/MainClient/apps/Model/Model\%20Pro/Data\%20Flow/ML/Images/DecisionTrees-MachineLearining-01.png.

[40] C. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, 1948.

[41] L. Breiman, "Arcing the edge," *Technical Report 486, Statistics Department, University of California, Berkeley.*, 1997.

[42] T. K. Ho, "Random decision forests," *Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal*, 1995.

[43] (), [Online]. Available: https://towardsdatascience.com/the-concept-of-artificial-neurons-perceptrons-in-neural-networks-fab22249cbfc.

[44] J. B. D. P. Kingma, "Adam: A method for stochastic optimization," *arXiv:1412.6980*, 2014.

[45] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. 61, pp. 2121–2159, 2011. [Online]. Available: http://jmlr.org/papers/v12/duchi11a.html.

[46] T.Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural Networks for Machine Learning*, no. 4, pp. 26–31, 2012. [Online]. Available: http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.

[47] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, 2014.

[48] (), [Online]. Available: https://en.wikipedia.org/wiki/Convolutional_neural_network#/media/File:Typical_cnn.png.

[49] (), [Online]. Available: https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf.

[50] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv:1312.6114*, 2013.

[51] I. Higgins, L. Matthey, A. Pal, *et al.*, "beta-VAE: Learning basic visual concepts with a constrained variational framework," *ICLR 2017*, 2017.

[52] I. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, "Generative adversarial nets," *Proceedings of the International Conference on Neural Information Processing Systems (NIPS 2014)*, 2672–2680, 2014.

[53] (), [Online]. Available: https://medium.datadriveninvestor.com/artificial-intelligence-gans-can-create-fake-celebrity-faces-44fe80d419f7.

[54] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," *International Conference on Machine Learning PMLR*, 214–223, 2017.

[55] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskevera, and P. Abbeel, "InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets," *Proceedings of the International Conference on Neural Information Processing Systems (NIPS 2016)*, 2016.

[56] P.T.Komiske, E.M.Metodiev, and J.Thaler, "Energy flow networks: deep sets for particle jets," *Journal of High Energy Physics*, vol. 1, 2019.

[57] L. Shapley, "A value for n-person games," *Contributions to the Theory of Games II, Princeton University Press, Princeton*, pp. 307–317, 2022.

[58] B. Rozemberczki, L. Watson, P. Bayer, *et al.*, "The Shapley value in machine learning," *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, 2022.

[59] M. Jerčić, I. Jerčić, and N. Poljak, "Introduction and analysis of a method for the investigation of qcd-like tree data," *Entropy*, vol. 24, 2022.

[60] M. Jerčić and N. Poljak, "Exploring the possibility of a recovery of physics process properties from a neural network model," *Entropy*, vol. 22, 2020.

[61] J. Neyman and E. S. Pearson, "On the problem of the most efficient tests of statistical hypotheses.," *Phil. Trans. R. Soc. Lond. A.*, vol. 231, 1933.

[62] R. L. Streit, "A neural network for optimum neyman-pearson classification.," *IJCNN International Joint Conference on Neural Networks*, 1990.

[63] X. Tong, Y. Feng, and J. Li, "Neyman-pearson classification algorithms and np receiver operating characteristics," *Science Advances*, 2018.

Supervisor

Nikola Poljak is an associate professor at the physics department, Faculty of science, University of Zagreb. He received his PhD from the University of Zagreb in 2010 for his research in spin physics at the Brookhaven National Laboratory. Later, he worked as a postdoctoral researcher at the NIKHEF Institute in Amsterdam, Ruđer Boskovic Institute in Zagreb and at the Faculty of science at the University of Zagreb. His research interests concentrate on experimental particle physics and, more recently, the use of machine learning methods in physics. He is also interested in various problems in educational and general physics. In the course of his research, he has coauthored 445 papers with over 40 000 citations.

Curriculum vitae

Marko Jerčić was born on February 27, 1993 in Split. He attended Josip Pupačić Elementary School in Omiš and high school (III. Gimnazija) in Split, which he graduated in 2011. After that, he enrolled in physics studies at the Physics Department of the Faculty of Science at the University of Zagreb. He graduated with magna cum laude on the master's thesis "Reproduction and study of the ridge phenomenon in two-particle correlations on the ALICE detector" under the guidance of izv.prof.dr.sc. Nikola Poljak. He began his work experience in 2016 as an assistant at the Department of Physics on the courses General Physics and Computer Science, and enrolled in a postgraduate course in Physics majoring in Elementary Particle Physics. He is a member of the international collaborations ALICE and RD51. He is the winner of the Dean's Award in 2021. He is also the winner of two "Brdo" awards, which are awarded to the best lecturers as chosen by students. Until now, he has published 252 scientific papers as an author or co-author.