Deep Learning for Identification of Short-Lived Particles in the CBM Experiment at FAIR

Master Thesis

for attaining the Master of Science degree
in Computer Science

submitted by

Robin Lakos

Institute of Computer Science
Johann Wolfgang Goethe-University
Frankfurt am Main, Germany

Supervisor: Prof. Dr. Ivan Kisel

05.04.2023

Erklärung zur Abschlussarbeit

gemäß § 34, Abs. 16 der Ordnung für den Masterstudiengang Informatik vom 17. Juni 2019

Hiermit erkläre ich

Lakos, Robin
(Nachname, Vorname)

Die vorliegende Arbeit habe ich selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel verfasst.

Ebenso bestätige ich, dass diese Arbeit nicht, auch nicht auszugsweise, für eine andere Prüfung oder Studienleistung verwendet wurde.

Zudem versichere ich, dass die von mir eingereichten schriftlichen gebundenen Versionen meiner Masterarbeit mit der eingereichten elektronischen Version meiner Masterarbeit übereinstimmen.

Frankfurt am Main, den 05.04.2023

Unterschrift des Studierenden

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Prof. Dr. Ivan Kisel, for his invaluable guidance, feedback and support throughout the present thesis and beyond. I am also grateful for the members of his work group, who have supported me at all times. Lastly, special thanks to my family and friends. During my studies, their belief in me has kept my motivation high.

Abstract

The future heavy-ion experiment Compressed Baryonic Matter (CBM) at the Facility for Antiproton and Ion Research (FAIR) will provide scientists around the globe an opportunity to investigate rare short-lived particles and the properties of matter under extreme conditions. CBM's research program will complement existing heavy-ion experiments by studies of phase transitions at high baryonic densities and, furthermore, with a collision rate of up to 10 MHz, CBM offers unprecedented capabilities in the research of rare short-lived particles. The event selection in CBM requires a full event reconstruction that is accomplished by the First Level Event Selection (FLES) package.

One of the centerpieces of FLES is the Kalman Filter Particle Finder (KFPF), an algorithm package for online reconstruction and selection of short-lived particles. KFPF includes a method for reconstruction of particle decays, where possible mother particles compete with each other. The competition initiates a rejection of falsely assumed mother particles and, hence, reduces the background. This, in turn, directly increases the quality of physics analysis. In this work, neural networks are used as an attempt to improve the method's performance.

In recent years, the application of machine learning algorithms and neural networks in particular has become popular in several operational areas. A high performance neural network package called Artificial Neural Networks for First Level Event Selection (ANN4FLES) is developed to be integrated in FLES. In the present thesis, ANN4FLES is integrated in FLES for the first time to solve a classification task within the package.

The K_s-mesons and Λ-hyperons are neutral particles that consist of strange quarks. Since theoretical predictions suggest that enhanced strangeness production might be an indicator for deconfined matter, K_s and Λ are investigated in particular. Both particles are present abundantly and offer therefore reliable information about the collision's properties.

Two deep learning approaches are introduced: One based on the mass and PDG-mass values of the candidates, the other using mass and transverse momentum instead. It will be shown that the difference between both ANN4FLES approaches is negligible. During training- and testing-phase, both networks classify candidates with an accuracy of more than 98% on the validation set. The pre-trained neural network is then included in KFPF to show its potential in the event reconstruction chain. Therefore, not only the raw classification performance is analyzed, but also a signal-background analysis using the performance tools provided by the KFPF. For K_s and Λ, the neural networks offer slightly better results than the existing method of KFPF. This suggests that neural networks might be helpful to improve physics analysis by solving the competition task. However, other particles are not considered in the present work. Therefore, it is suggested to extend the approach to a multiple particle classification.

Kurzfassung

Das zukünftige Schwerionenexperiment Compressed Baryonic Matter (CBM) an der Facility for Antiproton and Ion Research (FAIR) wird Wissenschaftlern aus vielen Ländern der Welt die Möglichkeit bieten, seltene kurzlebige Teilchen sowie die Eigenschaften von Materie unter extremen Bedingungen zu untersuchen. Das CBM-Forschungsprogramm wird die bestehenden Schwerionenexperimente durch Untersuchungen von Phasenübergängen bei hohen Baryonendichten ergänzen. Darüber hinaus bietet CBM mit einer Kollisionsrate von bis zu 10 MHz noch nie dagewesene Möglichkeiten zur Erforschung seltener kurzlebiger Teilchen. Die Ereignisauswahl in CBM erfordert eine vollständige Ereignisrekonstruktion, die mithilfe des First Level Event Selection (FLES) Pakets durchgeführt wird.

Eines der Herzstücke des FLES Pakets ist der Kalman Filter Particle Finder (KFPF), ein Algorithmenpaket für die online Rekonstruktion und Auswahl von kurzlebigen Teilchen. KFPF beinhält eine Methode zur Rekonstruktion von Zerfallsprozessen, in der mögliche Mutterpartikel-Kandidaten miteinander konkurrieren. Dieser Wettkampf zwischen Kandidaten initiiert eine Ablehnung von fälschlicherweise rekonstruierten Mutterpartikeln und reduziert so Störsignale (engl. *Background*). Dadurch hat die Methode einen direkten Einfluss auf die Qualität der Ergebnisse, die für die physikalische Analyse notwenig sind. Als ein Verbesserungsversuch des bereits implementierten Wettkampfs, werden in der vorliegenden Thesis neuronale Netze eingesetzt.

In den letzten Jahren hat die Anwendung von neuronalen Netzen in verschiedenen Einsatzbereichen an Beliebtheit dazu gewonnen. Daher wurde ein leistungsstarkes Paket für neuronale Netze namens Artificial Neural Networks for First Level Event Selection (ANN4FLES) entwickelt, das in der vorliegenden Arbeit zum ersten Mal in FLES integriert wird, um eine Klassifizierungsaufgabe innerhalb des Pakets zu bewältigen.

Die beiden neutralen Partikel K_s-Meson und Λ-Hyperon beinhalten beide Strange-Quarks. Da theoretische Vorhersagen darauf hindeuten, dass eine erhöhte Strangeness-Produktion ein Anzeichen für entfesselte Materie (engl. deconfined matter) sein könnte, beinhalten beide Partikel wertvolle Informationen. Zudem kommen beide Partikel in den Kollisionen zahlreich vor und bieten somit eine zuverlässliche Informationsquelle über die Eigenschaften der Kollision.

In dieser Arbeit werden zwei Deep-Learning Ansätze vorgestellt. Der erste Ansatz basiert auf den Masse-Werten und den Soll-Masse-Werten (PDG-Masse) der konkurrierenden Kandidaten, während der zweite Ansatz auf den Masse-Werten und dem Transversalimplus der jeweiligen Kandidaten basiert. Es wird gezeigt, dass beide ANN4FLES-Ansätze mit einer Klassifizierungsrate von über 98% auf den Testdaten hervorragende Klassifizierungsergebnisse erzielen. Die vor-trainierten neuronalen Netze werden dann in den KFPF implementiert, um deren Leistung in der vollständigen Rekonstruktionskette zu messen. Mithilfe der von KFPF bereitgestellten Werkzeuge zur Leistungsmessung wird eine Signal-Background-Analyse durchgeführt. Da die neuronalen Netze auch hier gute Ergebnisse erzielen, wird für zukünftige Forschung empfohlen, Ansätze mit neuronalen Netzen weiter zu verfolgen, insbesondere mit der Implementierung einer Klassifikation von mehreren Partikeln.

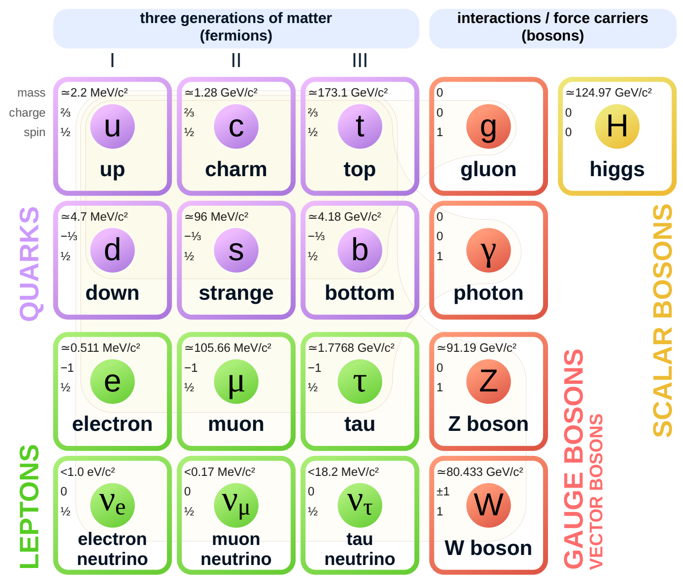
Contents

Chapter 1

Introduction

For several centuries, humanity assumed that all matter is made up of *atoms* - very small and indivisible. Back in 1911, after his famous gold-foil experiment, Rutherford supposed that atoms consists of a tiny high concentrated positively charged atomic nucleus, whereas the rest of the atom is spread out in a relatively large volume around the core [1]. Later, further experiments gained evidence that supported the theory of an atom's core-shell structure, which is generally acknowledged until today. However, at the latest in the early 1960s, physicists proposed their first theories about the existence of *quarks* as fundamental particles that make up protons, neutrons and other particles [2, 3].

Nowadays, the idea of quarks is considered as a breakthrough in the understanding of the subatomic world. Since the 1970s, experimental evidence confirmed not only the existence of quarks, but also of other fundamental particle classes, such as leptons, gauge bosons and scalar bosons. The developed Standard Model (see Figure 1.1) summarizes the current knowledge in particle physics, including six *flavors* of discovered quarks and their corresponding anti-particles, three types of charged leptons along with their associated neutrinos. Furthermore, there are other particles that interact with quarks and leptons in various ways, such as gluons, that are strong force-carrying particles responsible to bind quarks and, thus, help to form hadrons like protons and neutrons.



three generations of matter (fermions)
interactions / force carriers (bosons)
I
II
III
mass
charge
spin
≈2.2 MeV/c² ⅔ ½ u up
≈1.28 GeV/c² ⅔ ½ c charm
≈173.1 GeV/c² ⅔ ½ t top
0 0 1 g gluon
≈124.97 GeV/c² 0 0 H higgs
≈4.7 MeV/c² −⅓ ½ d down
≈96 MeV/c² −⅓ ½ s strange
≈4.18 GeV/c² −⅓ ½ b bottom
0 0 1 γ photon
≈0.511 MeV/c² −1 ½ e electron
≈105.66 MeV/c² −1 ½ μ muon
≈1.7768 GeV/c² −1 ½ τ tau
≈91.19 GeV/c² 0 1 Z Z boson
<1.0 eV/c² 0 ½ νe electron neutrino
<0.17 MeV/c² 0 ½ νμ muon neutrino
<18.2 MeV/c² 0 ½ ντ tau neutrino
≈80.433 GeV/c² ±1 1 W W boson
QUARKS
LEPTONS
GAUGE BOSONS
VECTOR BOSONS
SCALAR BOSONS

Figure 1.1: The Standard Model of particle physics, including the six flavors of quarks, three types of leptons with their respective neutrinos and particles responsible for interactions. [4]

For example, the already mentioned proton consists of two up quarks and one down quark, resulting in a positive electric charge of $2/3 + 2/3 - 1/3 = 1$ whereas neutrons consist of a single up quark and two down quarks, resulting in a neutral electric charge of $2/3 - 1/3 - 1/3 = 0$. These particles are grouped by their amount and type of quarks: three quarks build a baryon, three anti-quarks build an anti-baryon, whereas a quark plus an anti-quark build a meson. Today, about 120 types of baryons and 140 types of mesons are known [5] and the properties of these particles are studied in particle physics experiments around the world for several reasons. Firstly, a better understanding of matter and the forces that govern it. For instance, where does the strong nuclear force originates that binds quarks and gluons, and why is it stronger than other fundamental forces such as gravity and electromagnetism [6]? Secondly, these experiments provide an opportunity to test existing theories in physics like the studies of assumed phase transitions of matter in the Quantum Chromodynamics phase diagram. And thirdly, the properties of particles might be important for new technologies: for example, applications in medicine. There are already experiments for heavy-ion cancer therapy with confident results [7]. However, one of the main reasons is the discovery of new physics within the Standard Model and beyond, that lead to a better understanding of our universe.

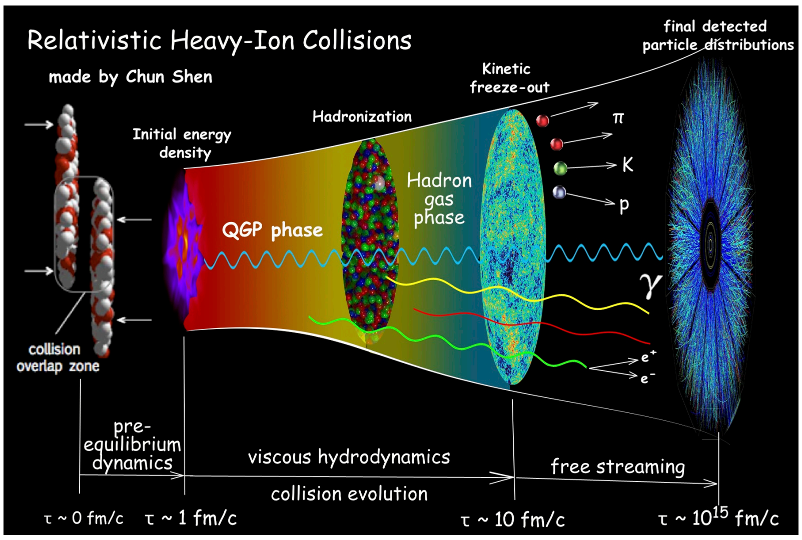


Figure 1.2: Relativistic heavy-ion collisions as a tool to understand the structure and behavior of matter. Starting from the left, particles collide within the collision overlap zone, whereas particles that do not take part in the collision are called spectators. Under extreme conditions, colliding particles transform into Quark Gluon Plasma (QGP), where deconfinement allows quarks to move freely and form to hadrons (Hadronization). In the kinetic freeze-out, the hadron gas cools down further below the point where new hadrons can be formed. The existing hadrons fly into the detector setup and the collision can be reconstructed. [8]

1.1 Heavy-Ion Experiments

The future heavy-ion experiment Compressed Baryonic Matter (CBM) at the Facility for Antiproton and Ion Research (FAIR) will provide scientists the ability to study super-dense nuclear matter under various extreme conditions, producing enormous high baryonic densities [9]. Whereas other experiments such as STAR (BNL) and

ALICE (CERN) investigate the properties of matter under extreme temperatures as they existed in the Big Bang [10], CBM complements the heavy ion research program by focusing on the state of matter as it occurs in the center of neutron stars before they collapse into black holes.

From another perspective, the research area of CBM is a different region in the Quantum-Chromodynamics (QCD) phase diagram, compared to other heavy-ion physics experiments. The diagram visualizes states of matter and phase transitions of nuclei under extreme conditions of temperature and density (see Figure 1.3), that are studied by researchers worldwide. Further, it illustrates the investigation area of the critical point, where the conditions are created for the so-called deconfinement: the phase transition where quarks and gluons become Quark-Gluon-Plasma (QGP) and break their strong connectivity.

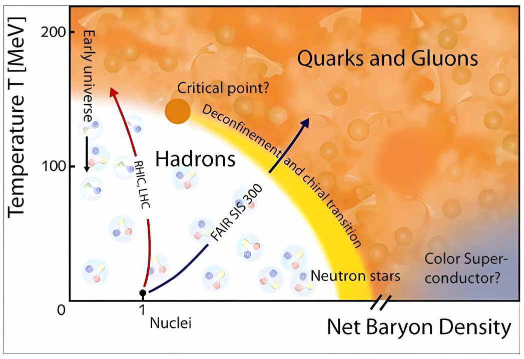


Figure 1.3: Quantum-Chromodynamics (QCD) phase diagram: temperature on y-axis, density on x-axis. The dark blue arrow indicates the research area of CBM (FAIR), the red arrow shows the region of STAR (RHIC) and ALICE (LHC). The areas in blue and orange are studied in particle physics experiments around the globe. [11]

In general, there are two types of heavy-ion physics experiments: fixed-target and collider experiments. For example, STAR[1] and ALICE are collider experiments. In these experiments, beams of heavy-ions are accelerated from opposite directions up to relativistic speeds to collide within the detector environment. Their collision creates enormous temperatures in the main collision point, the so-called *primary vertex*: Colliding ions burst into particles that are measured by the detectors that are build barrel-shaped around the intended collision point. A fixed-target experiment, such as CBM, accelerates beams of particles targeting a stationary target, e.g. a block of gold. In these experiments, the detectors are usually build around and behind the intended target, such that particles can be measured after the collision.

Some of the created particles are common and already well-investigated, whereas others are rare. The investigation of rare short-lived particles is a major goal of the CBM experiment, but rareness leads to two further challenges. First of all, the experiment has to be repeated thousands of times to create rare events where particles of interest are created. Statistical evidence is crucial when errors in measurements, calculation inaccuracy and mistakenly assumed hypotheses can not be eliminated with certainty. Secondly, there is a huge amount of energy required to run these experiments [13]. Beside the particle accelerator itself, a major factor is the energy consumption by high performance computers that are used for reconstruction and analysis algorithms during the beam-time. Furthermore, in CBM, a full event reconstruction is required to be performed in real-time and therefore, efficient and fast algorithms are essential for event analysis [14]. In the present thesis, neural network

[1]Within the scope of the RHIC Beam Energy Scan program, STAR also has a fixed-target program for investigation of lower energies. [12]

based approaches are further investigated to increase the reconstruction efficiency for the experiment.

1.2 Reconstruction of Events

The study of particles created in the collisions, so called *events*, requires to process large amounts of detector responses of up to 1TB/s [15]. On the one hand, the collision rates are chosen as high as possible to find rare particles efficiently. On the other hand, data streams created by these high interaction rates can not be fully-stored by modern computers. Therefore, usually, first analysis stages are used to select events of interests, so called trigger stages. In CBM, a major goal is the investigation of rare short-lived particles. Since these particles tend to decay before reaching any detector they could interact with, there is no simple criteria to implement such triggers in CBM. The experiment requires a full event reconstruction for the decision of storing events of interest. The full event reconstruction processing requires fast algorithms for online-reconstruction. The First Level Event Selection (FLES) package is used for handling CBM's high interaction rates of up to 10 MHz, including a reconstruction of particle decays. After that, a decision can be made if an event is of interest and should be stored or rejected.

However, in recent years the application of machine learning algorithms in particle physics experiments has become popular and neural networks, for example, were applied in several stages of the experiments [16, 17, 18]. For applications in CBM's FLES package: ANN4FLES, a fast, modular and independent neural network package is developed [19, p. 161]. In this work, deep neural networks provided by ANN4FLES are applied to the reconstruction of particle decays of K_s-mesons and Λ-hyperons. Whereas ANN4FLES is still in development, the neural networks were tested and compared on multiple well-known data sets to allow confidence in the network's functionality.

The Kalman Filter Particle Finder (KFPF), an important package inside FLES, is used for the reconstruction of particles and their decays online [20]. After the particle trajectories (tracks) were reconstructed and extrapolated by other packages of FLES, KFPF tries to find intersections between tracks to find possible points of decay - so called *secondary vertices*. Either due to the extreme conditions, particle instability, particle interaction with other particles or interactions between particles and detectors, some particles decay (see Figure 1.4). This might happen right after the collision or at and between detectors. The KFPF package provides reconstruction capabilities for more than 150 decays, covering all signals relevant for the CBM experiment [21].

In this work, the performance of neural networks in this context is investigated. When particles decay, they are referred to as mother particle, that splits into multiple daughter particles. In KFPF, a competition approach already exists. The existing method compares two mother particle candidates, using their mass distance to their respective known mass distribution peak, to find the best fitting mother particles. Particles that lose the competition are rejected. Here, the challenge is that some daughter particles can be raised by different types of mother particles. For example,

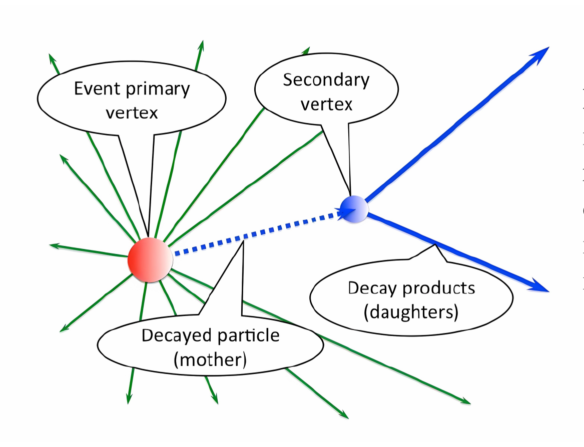


Figure 1.4: Starting from the primary vertex, particles fly in many directions. Several of them tend to decay at the so called secondary vertices into daughter particles. The decaying particle is called mother particle. [22]

the decay of $\Lambda \rightarrow \mathrm{p}\pi^-$ gives raise to π^- as well as the decay of $K_s \rightarrow \pi^+\pi^-$. In turn, for a full event reconstruction a hypothesis has to be made, if π^- was raised by the decay of Λ or K_s.

Previous work suggests, that neural networks can offer comparable results to the existing approach, investigating the decays of Λ and K_s [23]. The neural network was trained on 10'000 simulated events using generated data of the PHSD model, classifying Λ or K_s based on the mass values of the mother candidates and their respective PDG mass — the mass particles should have. However, it also proposes to study the classification performance based on the candidate's masses and transverse momentum, which is also studied and introduced in the present thesis.

Chapter 2

The CBM Experiment at FAIR

The Facility for Antiproton and Ion Research (FAIR) is an international particle accelerator facility being build in Darmstadt, Germany. It extends the research areas and technical possibilities of the already existing Gesellschaft für Schwerionenforschung (GSI) laboratory [24]. FAIR provides scientists from all over the world the opportunity to analyze the structure of matter to gain insights in the evolution of our universe. The synchrotrons SIS100 and SIS300 will form the centerpiece at FAIR, providing acceleration of particles up to relativistic speeds.

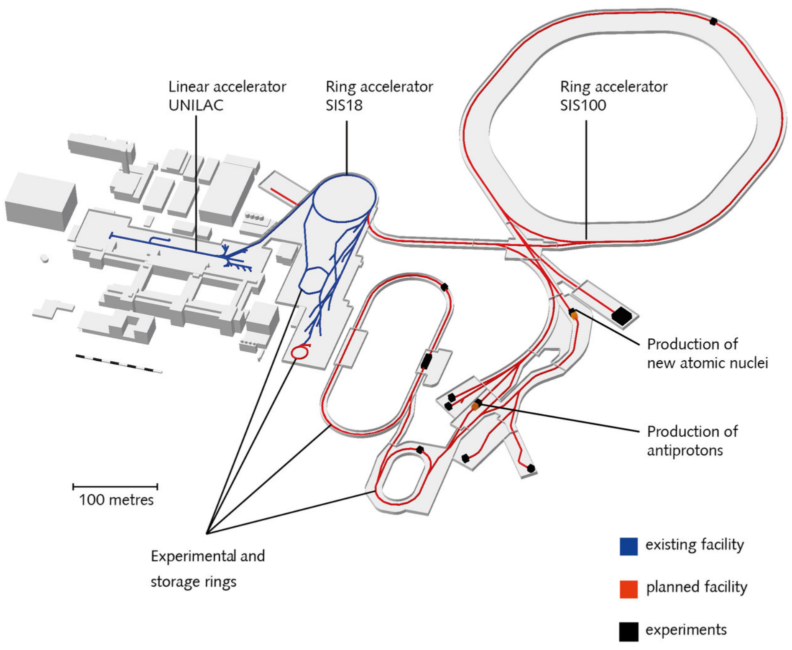

Linear accelerator UNILAC
Ring accelerator SIS18
Ring accelerator SIS100
Production of new atomic nuclei
Production of antiprotons
100 metres
Experimental and storage rings
■ existing facility
■ planned facility
■ experiments

Figure 2.1: Overview of experimental complexes Gesellschaft für Schwerionenforschung (GSI) in blue and Facility for Antiproton and Ion Research (FAIR) in red. [25]

The synchrotrons SIS100 and SIS300 are designed to create heavy-ion and proton beams with an energy range of 2-44 AGeV [26]. The charge of ions and protons is utilized to accelerate the particles with magnets, allowing significant fractions of the speed of light. At these speeds, relativistic effects such as time dilation and length contraction become relevant as a direct consequence of the special theory of relativity [27]. Therefore, for instance, particles that collide at relativistic (or ultra-relativistic) speeds, will have a longer lifetime before decaying, which has to be considered in physics analysis. The accelerators will be used to generate high frequency collisions of up to 10 MHz for the analysis of rare probes, e.g. for the CBM experiment.

2.1 Compressed Baryonic Matter (CBM) Experiment

Compressed Baryonic Matter (CBM) at FAIR will be a heavy-ion experiment used to (1) explore the region of high baryonic densities and moderate temperatures in the Quantum-Chromodynamics phase diagram and (2) search for rare short-lived particles. CBM is a fixed-target experiment and thus has a detector setup lined up behind and around the stationary target (typically several 100μm thin [28]) to measure particles produced in the collisions. As shown in Figure 2.2, the planned detector setup for CBM consists of up to seven detectors, each measuring different types of particles or used for specific measurements as, for instance, momentum and energy of particles.

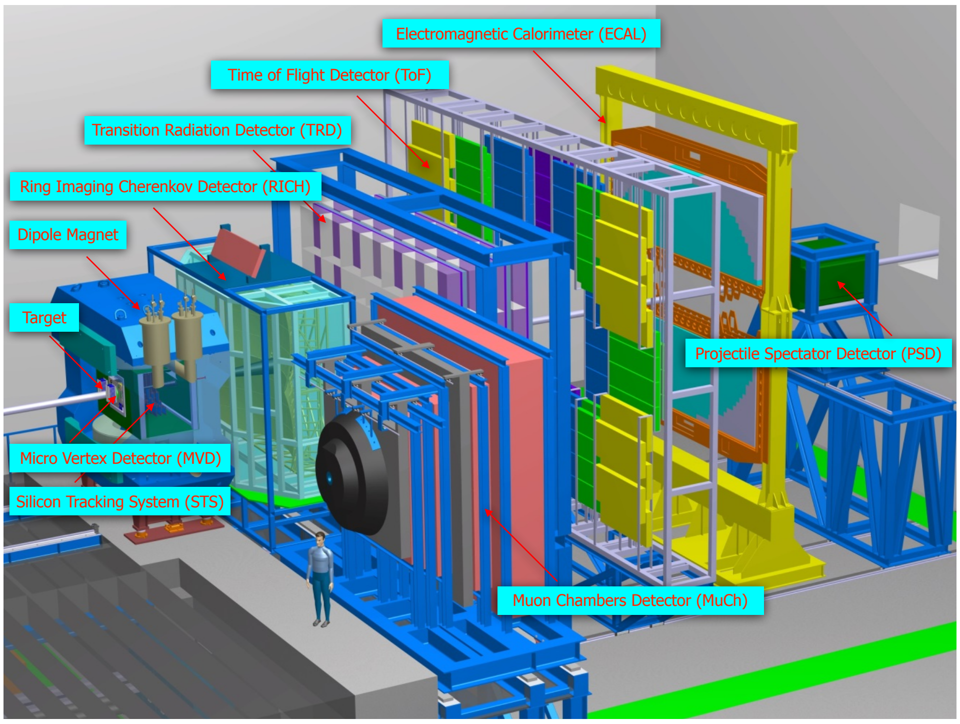

Electromagnetic Calorimeter (ECAL)
Time of Flight Detector (ToF)
Transition Radiation Detector (TRD)
Ring Imaging Cherenkov Detector (RICH)
Dipole Magnet
Target
Projectile Spectator Detector (PSD)
Micro Vertex Detector (MVD)
Silicon Tracking System (STS)
Muon Chambers Detector (MuCh)

Figure 2.2: Planned detector setup of the future CBM experiment at FAIR: particle beam direction from left to right. RICH, TRD and ECAL are removed in case of muon setup, whereas RICH is replaced by MuCh. [21]

A particle can only be recognized, if it interacts with a detector or decays into detectable particles. Thus, for example, when a neutral particle hits a detector that is triggered by electrical charge, it will be invisible and has to be measured by other detectors directly or indirectly through reconstruction by other methods, such as the Missing Mass Method [21].

The first detectors are a *Micro Vertex Detector (MVD)* for high resolution tracking of charged particles and a *Silicon Tracking System (STS)* to track particles over a wider range of angles and momenta. Both are build within a magnet. The magnetic field curves charged particles' trajectories, which (1) allows to recognize their electric charge's sign based on the deflection direction and (2) makes it possible to calculate particles' momenta based on their trajectories' radius. Therefore, MVD and STS are able to recognize positions, type of charge and momenta of charged particles close to the primary vertex. [29, 30, p. 625-626]

The *Ring-Imaging CHerenkov (RICH)* detector follows: Charged particles that fly through it send out Cherenkov radiation (light) by means of photons. Depending on the particle's properties, the angle in which these photons leave the particle changes. Thus, the detector measures photon rings with a diameter that allows to infer the particle's properties and therefore helps to identify particles. [31, 30, p. 626-627]

As the third detector in CBM, a *Transition Radiation Detector (TRD)* is planned. The TRD is designed to detect charged particles such as electrons and positrons. It is usually built with thin metal foil layers separated by gas-filled chambers. When charged particles pass the foils, they will interact and emit transition radiation. This in turn can provide information about particles momenta and energy by measuring the amount and energy distribution of the transition radiation within the chambers. [32, 30, p. 627]

The *Time Of Flight (TOF)* detector follows TRD. As the name suggests, the TOF measures the time that particles take to travel within the detector from one point to another. This provides information about the particles' momenta and velocities. The velocity in turn is used to identify different types of particles with similar energies but different masses. For example, it can be used to separate pions from kaons or protons from electrons. [33, 30, p. 628]

The second last detector in CBM is the *Electromagnetic CALorimeter (ECAL)*. It actually destroys charged particles to determine their total energy. These particles decay within the material and produce so called electromagnetic showers causing e^+ / e^- pairs and photons. These shower products in turn interact with other materials that emit light, which is measured and converted into an electrical signal. The particles' energy and the measured light is proportional and therefore allows the calculation of the particle's energy. [34, 30, p. 628]

Finally, the Projectile Spectator Detector (PSD). As it is not unusual for colliding particles to have a slight non-centrality, there exist a part of particles that do not participate in the collision itself and miss the target - so called spectators. The energy and multiplicity of spectators provide information about the nature of an event, and therefore are crucial for the studies at CBM. [35, 30, p. 628]

However, there is another planned detector setup that allows to search for muons. In this case, RICH, TRD and ECAL are removed and a *Muon Chambers (MuCh)* are put in place of RICH [20, p. 14]. Although muons are similar to electrons, they are heavier and can pass through many materials without interaction. Thus, as mentioned before, the non-interaction makes them invisible for many detectors, requiring special detectors. [30, p. 627-628]

Summarized, particles created in the collision are measured by a set of detectors that make use of various physical and chemical reactions. These detector hits can then be used to identify the created particles and to reconstruct parts of the event.

2.2 First Level Event Selection (FLES)

The particles created in collisions of the CBM experiment can be divided into two classes: (1) long-lived particles that are measured directly in a detector setup around the point of collision and (2) short-lived particles that decay - sometimes multiple times - before reaching any detector and which are particularly of scientists' interest. Since the study of rare short-lived particles is a major goal of CBM, high collision rates are required to obtain statistical evidence despite the poor particle yields of rare probes. Therefore, the events are being created with a rate of up to 10 MHz $= 10^7$ collisions per second, which leads to further challenges.

At this rate, modern computers are not able to store the data streams completely. Assuming a theoretical maximal record speed of 7GByte/s for current M.2-NVMe SSDs and 40kByte of data for a gold-gold heavy-ion collision, it is possible to store the data of collisions at a rate of not more than 175kHz. Therefore, obviously not all data will be stored. The data streams created through detector response signals are transmitted into the Data AcQuisition (DAQ) room, where a first data pre-processing is executed (see Figure 2.3). Here, a FLES input node computer cluster prepares the data for its transfer to the Green Cube, the high performance computer center of GSI/FAIR, where the First Level Event Selection is processed [36].

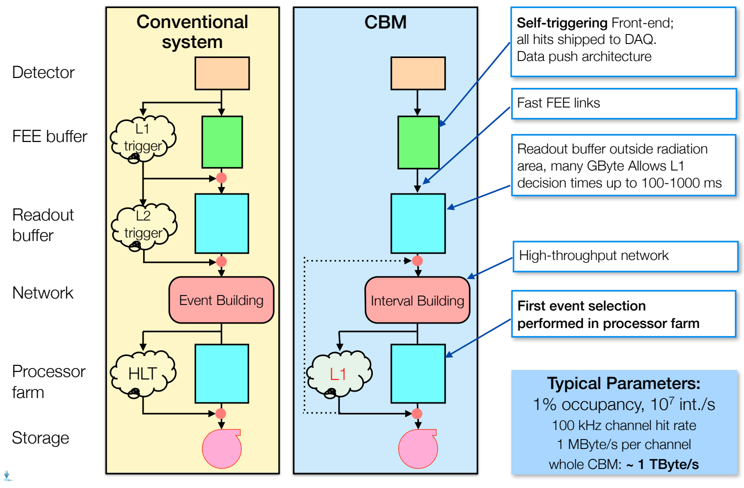


Figure 2.3: Visualization of the Data AcQuisition (DAQ) and event selection at the CBM experiment compared to conventional multi-level trigger stages in other heavy-ion experiments. [21]

In CBM, there is no simple criteria for event selection. As short-lived particles might not be measured at all, a full event reconstruction is required for each collision, including the reconstruction of particle decays to find short-lived particles of interest. An example is the decay-chain of $\bar{\Omega}^+$, that consists of three anti-strange quarks $(\bar{s}\bar{s}\bar{s})$, the anti-particle for Ω^+ that itself consists of three strange quarks (sss). $\bar{\Omega}^+$ is an highly instable particle with a short lifetime of $(0.823 \pm 0.011) \times 10^{-10}$ seconds [37] and is a rare particle as it contains three anti-strange quarks. Theoretical predictions suggest that enhanced strangeness production might be an indicator for deconfined matter [38] and strange quarks are abundantly created in the energy range of CBM [23], making them attractive to study the high density area of the QCD phase diagram.

In the present thesis, K_s mesons and Λ hyperons are investigated in particular, as they both include strange quarks. Since these particles are not rare, they reliably provide information about the collisions properties.

However, whereas K_s and Λ are important for different reasons, rare particles such as $\bar{\Omega}^+$ require the reconstruction of two secondary vertices: Since $\bar{\Omega}^+ \to \bar{\Lambda}K^+ \to \pi^+\bar{\text{p}}K^+$ (see Figure 2.4), first $\bar{\Lambda}$ has to be reconstructed before the secondary vertex of $\bar{\Omega}^+$ can be reconstructed. Thus, the investigation of $\bar{\Omega}^+$ requires a full event reconstruction, such that events including the particle are stored to disk for later analysis.

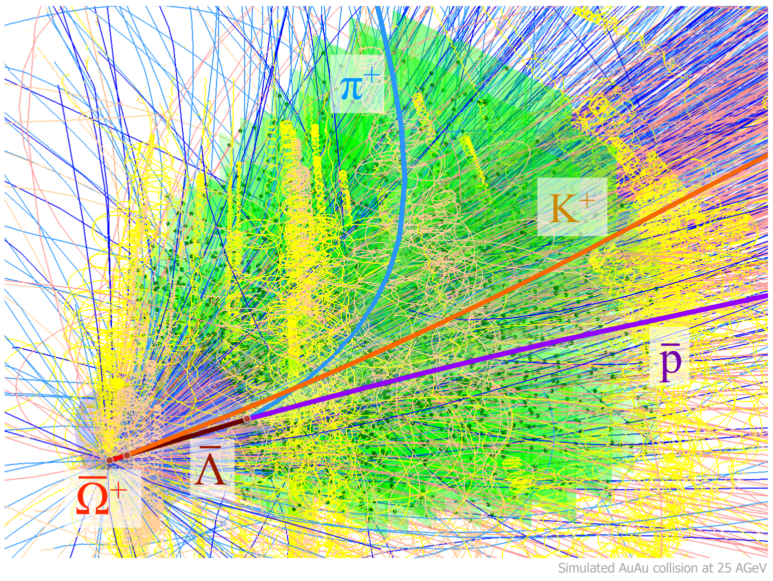


Figure 2.4: Reconstruction of an event. The detector planes are visualized in green, particles are colored depending on their properties and highlighted $\bar{\Omega}^+ \to \bar{\Lambda}K^+ \to \pi^+\bar{\text{p}}K^+$ decay chain. [20]

To accomplish the challenge, a high performance algorithm package, the First Level Event Selection (FLES) [39, 40], is used to select events of interest. This package includes the Cellular Automaton (CA) based Track Finder, a Kalman Filter based Track Fitter, an Event Builder, a Kalman Filter based Particle Finder and a physics analysis, which are all used for event selection (see Figure 2.5). Using FLES, events of interest can be identified and recorded.

The detector setup has specific geometries, such as distances between the detectors or to the collision target. These have to be provided into the FLES package for several calculations, e.g. velocity that requires time and distance. Another input to the package are detector measurements - so called *hits*.

Since the CBM experiment has not started yet, no real experiment data is used for this thesis. Instead, generated simulated data is utilized. In case of CBM, there are two physics models mainly used to generate data: The Parton-Hadron-String Dynamics (PHSD) [41] model and the Ultra-relativistic Quantum Molecular Dynamics (UrQMD) model [42, 30]. The created generated particles are then processed by a transport engine (e.g. GEANT4) to simulate the particles flying into the detector system, including all relevant physics processes such as decays, deflection and in-

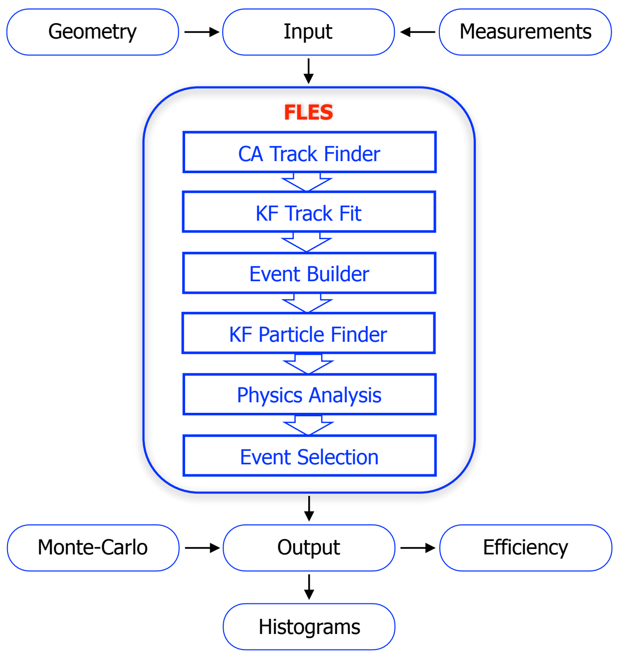


Figure 2.5: Flowchart of the First Level Event Selector (FLES) package that is used in the CBM experiment to evaluate detector response data streams in regard to the interest of physicists. [20]

teraction [28]. At this point, the whole event is created, including all decay chains and trajectories. As a next step, the detector responses are generated, resulting in hits that would also be produced by real data. Hence, hits based on simulated data are used instead of hits produced by real experiment data.

Nevertheless, the quality of simulations in any form is evaluated meticulously to perform as close to real experiment data as possible. This includes the huge amount of data that has to be processed in real-time [36].

In FLES, in a first step, the provided hits are used to determine possible tracks of particles with the 4-dimensional (time and space) CA Track Finder [43]. Here, hits are first connected into segments with all possible matching hits of neighboring detectors. Then, χ^2 cuts are applied on the segments to obtain the most likely track candidates for respective particles.

Then, an on the Kalman Filter method based algorithm calculates the parameters of tracks found by the CA Track Finder. For instance, these parameters include momentum and curvature. This process involves the predicted hit positions based on the track candidates and the actual hit information by the detector and is called *Track Fitting*.

With an interaction rate of 10 MHz, consecutive events will most likely have timely overlapping hits in the detectors [44, 28]. For a high precision measurement and reconstruction, it is mandatory to separate registered particles by their corresponding event. Otherwise, hits of different events lead to wrongly combined tracks. To ensure accurate event-by-event reconstruction, FLES uses an Event Builder after the Kalman Filter based Track Fit.

After the tracks per event are reconstructed, the Kalman Filter Particle Finder package for online selection and reconstruction of short-lived particles is applied to the tracks [20]. This package provides a reconstruction of particle decays to reconstruct short-lived particles that are not registered by any detector. In this work, pre-trained neural networks provided by ANN4FLES are applied within this package. Thus, more details are provided in the following section 2.3.

Up to this point, the full events are reconstructed and depending on the scientists interests, events are selected if specific criteria are satisfied. On simulated event data, reconstructed results are compared to the simulated events, efficiencies are calculated, and FLES provides histograms for physics- and performance-analysis that are used for the present thesis.

2.3 Kalman Filter Particle Finder (KFPF)

The Kalman Filter Particle Finder (KFPF) package included in FLES is developed for online reconstruction and selection of short-lived particles [20]. Based on particle trajectories - so called *tracks* - and their respective particles, the algorithm tries to reconstruct the event, its participating particles and their decays.

In general, KFPF uses many different χ^2-tests to select information of higher quality. Although there is always existing imperfection in measurements (due to finite detector resolutions) and calculations (due to IEEE-754 inaccuracy), certain values could obscure any pattern as they have worse quality than others. Then, χ^2-tests are applied to filter low quality information. In a χ^2-test, measured values are compared to an underlying theoretical model or hypothesis. If the statistic for a certain value is larger than a threshold, it is considered unlikely that the value is consistent with the model or hypothesis.

In KFPF, the cut values (thresholds) are set individually. For example, some cuts are set by statistical evidence, whereas at the same time approaches exist to apply optimized cuts, e.g. by using Boosted Decision Trees [45], another machine learning technique. However, for KFPF, only tracks that pass certain criteria are selected, here by means of χ^2_{fit}. Then, χ^2_{prim} is used to separate primary and secondary tracks. Extrapolated tracks that lead to the *primary vertex* (within error margin), are classified as primary-tracks whereas those that intersect in other space-time coordinates are handled as secondary-tracks (see Figure 2.6).

Particles with trajectories that directly lead to the primary vertex can usually be considered as long-lived particles, as their lifetime is long enough to interact with detector materials to produce a signal. In this case, information of detectors can be used to calculate the particle's properties, for example, mass and energy. In contrast, particles that do not lead to the primary vertex are likely to be decay products that are created in secondary vertices. Here, if a detector close to the primary vertex (such as MVD) is considered, the particles did not live long enough to reach any detector. Although there are differences in several orders of magnitude in regard to their lifetime, these particles can be considered as short-lived particles. However, whereas information of long-lived particles can be measured directly by detector interaction, short-lived particles can only be measured indirectly, what makes it a challenge to reconstruct them. But, since primary particles originate from the main collision point, they provide crucial information about the collision's properties [46, p. 33]. Contrary, secondary particles contain the data necessary to infer properties of short-lived particles that are created in or close to the primary vertex, providing information

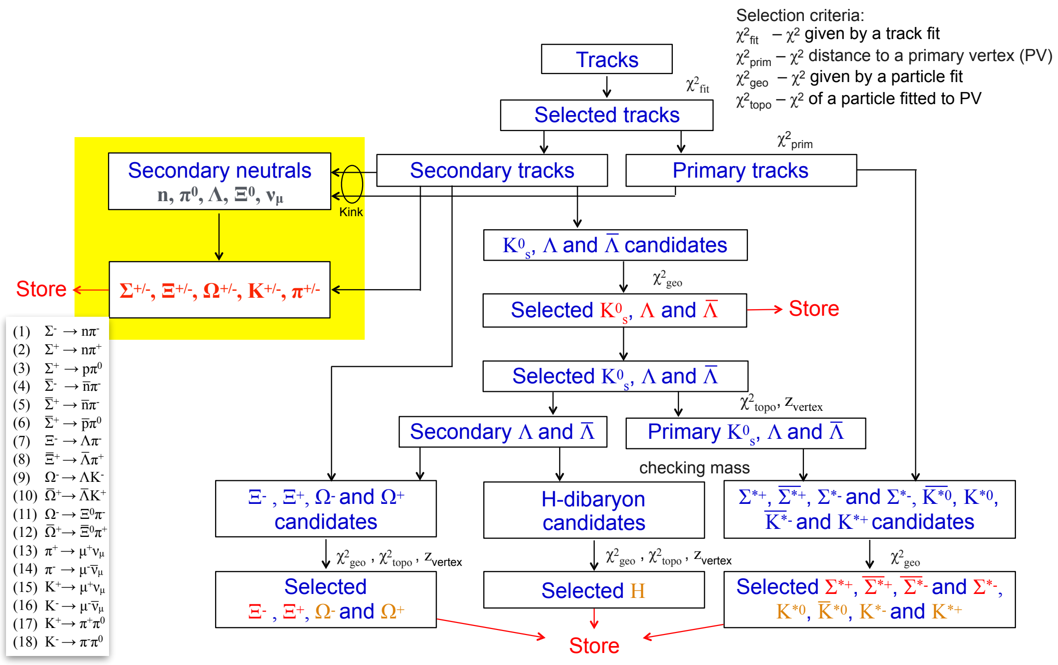

Selection criteria:
χ^2_{fit} – χ^2 given by a track fit
χ^2_{prim} – χ^2 distance to a primary vertex (PV)
χ^2_{geo} – χ^2 given by a particle fit
χ^2_{topo} – χ^2 of a particle fitted to PV
Tracks
χ^2_{fit}
Selected tracks
χ^2_{prim}
Secondary neutrals
n, π^0, Λ, Ξ^0, ν_μ
Secondary tracks
Primary tracks
Kink
K^0_s, Λ and $\bar{\Lambda}$ candidates
χ^2_{geo}
Store
$\Sigma^{+/-}$, $\Xi^{+/-}$, $\Omega^{+/-}$, $K^{+/-}$, $\pi^{+/-}$
Selected K^0_s, Λ and $\bar{\Lambda}$
Store
Selected K^0_s, Λ and $\bar{\Lambda}$
χ^2_{topo}, z_{vertex}
Secondary Λ and $\bar{\Lambda}$
Primary K^0_s, Λ and $\bar{\Lambda}$
checking mass
Ξ^-, Ξ^+, Ω^- and Ω^+ candidates
H-dibaryon candidates
Σ^{*+}, $\overline{\Sigma^{*+}}$, Σ^{*-} and Σ^{*-}, $\overline{K^{*0}}$, K^{*0}, K^{*-} and K^{*+} candidates
χ^2_{geo}, χ^2_{topo}, z_{vertex}
χ^2_{geo}, χ^2_{topo}, z_{vertex}
χ^2_{geo}
Selected Ξ^-, Ξ^+, Ω^- and Ω^+
Selected H
Selected Σ^{*+}, $\overline{\Sigma^{*+}}$, $\overline{\Sigma^{*-}}$ and Σ^{*-}, K^{*0}, $\bar{K}^{*0}$, K^{*-} and K^{*+}
Store
(1) $\Sigma^- \rightarrow n\pi^-$
(2) $\Sigma^+ \rightarrow n\pi^+$
(3) $\Sigma^+ \rightarrow p\pi^0$
(4) $\bar{\Sigma}^- \rightarrow \bar{n}\pi^-$
(5) $\bar{\Sigma}^+ \rightarrow \bar{n}\pi^-$
(6) $\bar{\Sigma}^+ \rightarrow \bar{p}\pi^0$
(7) $\Xi^- \rightarrow \Lambda\pi^-$
(8) $\bar{\Xi}^+ \rightarrow \bar{\Lambda}\pi^+$
(9) $\Omega^- \rightarrow \Lambda K^-$
(10) $\bar{\Omega}^+ \rightarrow \bar{\Lambda}K^+$
(11) $\Omega^- \rightarrow \Xi^0\pi^-$
(12) $\bar{\Omega}^+ \rightarrow \bar{\Xi}^0\pi^+$
(13) $\pi^+ \rightarrow \mu^+\nu_\mu$
(14) $\pi^- \rightarrow \mu^-\bar{\nu}_\mu$
(15) $K^+ \rightarrow \mu^+\nu_\mu$
(16) $K^- \rightarrow \mu^-\bar{\nu}_\mu$
(17) $K^+ \rightarrow \pi^+\pi^0$
(18) $K^- \rightarrow \pi^-\pi^0$

Figure 2.6: Selection scheme of the Kalman Filter Particle Finder (KFPF) package, including the recently added Missing-Mass method for the reconstruction of neutral particles (yellow). [21]

about particles of interest. Therefore, both types are relevant for the CBM experiment.

All in all, the KFPF package is capable of more than 150 decay-reconstructions [21], allowing the analysis of all particles relevant for the CBM experiment. The decays and decay chains can be seen in Figure 2.7.

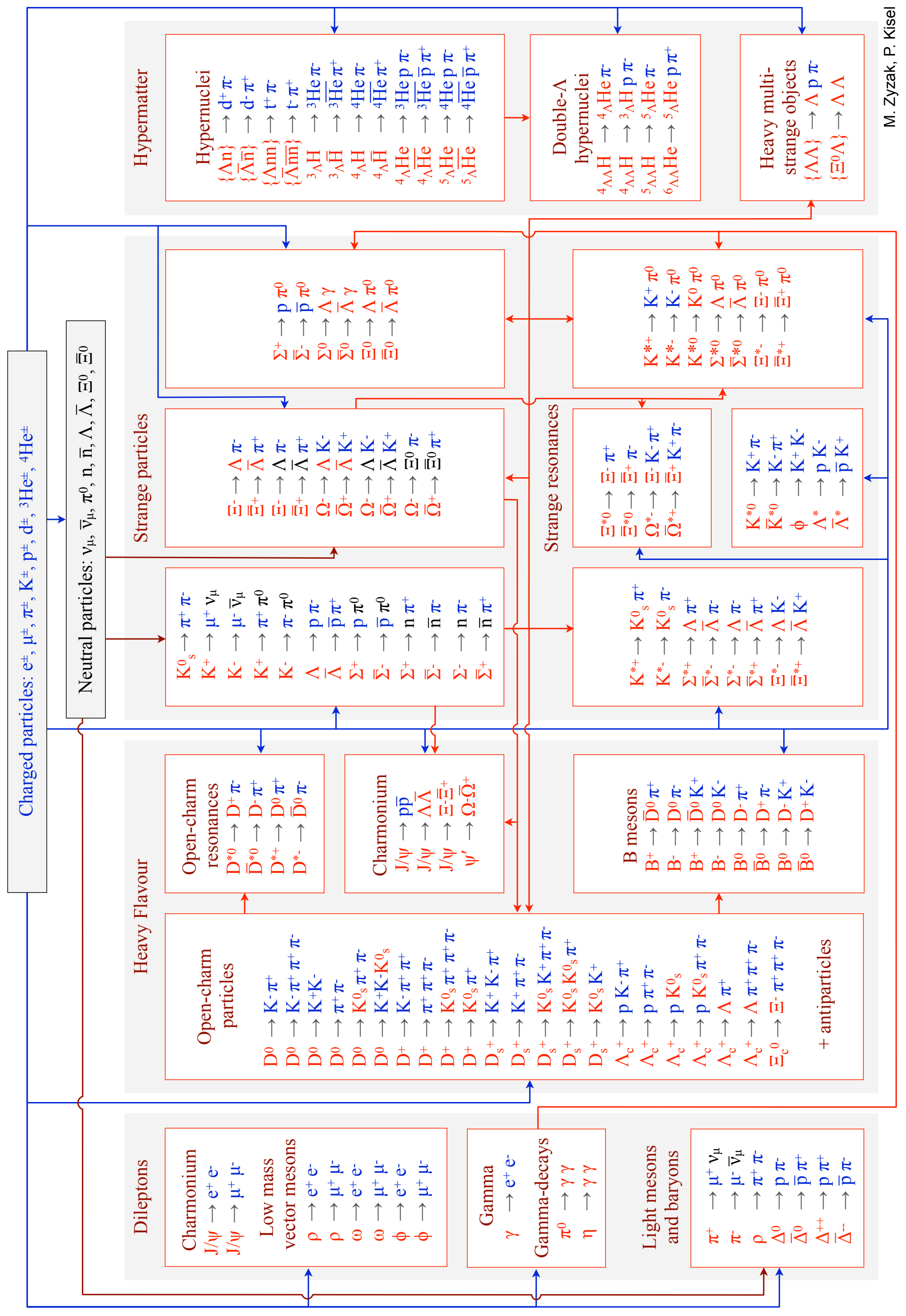


Figure 2.7: Decay scheme of particles reconstructed by the Kalman Filter Particle Finder (KFPF) package. The package is able to reconstruct more than 150 particle decays. The investigated particles in this thesis decay as follows: $K_s \rightarrow \pi^+\pi^-$; $\Lambda \rightarrow p\pi^-$. [21]

2.3.1 Particle Identification in the KFPF

In experiments such as the CBM experiment, the detection of rare, short-lived particles is crucial for understanding the underlying physics processes. However, the identification of these particles can be challenging due to the presence of secondary tracks produced by the decay of particles. The secondary tracks selected by the KFPF package are produced by decay products, so called *daughter particles*. Particles decay for various reasons: The particles are unstable, they collide with other particles or decay due to interaction with a detector. The decaying particle is called *mother particle* and might be a particle of interest, especially if it is a rare short-lived particle.

The KFPF package addresses this challenge by performing a competition among reconstructed mother particles. This competition involves the creation of all possible mother particles, followed by the removal of those that do not meet certain criteria. The criteria used for the competition are based on the properties of the daughter particles, such as their mass, momentum and direction of flight, and are designed to select the most likely mother particle.

In a first step, all possible mother particles are created. In a second step, the competition removes all created mother particles, if they do not meet certain criteria. The competition starts with the initialization of vectors that are used for the competition (see Listing 2.1). Here, `fParticles` is a vector including all reconstructed particles. The best matching mother particle ID is stored in the `bestMother` vector, whereas candidates that should be deleted are tagged in `deleteCandidate`.

```cpp
std::vector<ParticleInfo> particleInfo;
std::vector<bool> isUsed(fParticles.size());
std::vector<bool> deleteCandidate(fParticles.size());
std::vector<int> bestMother(fParticles.size());

for (unsigned int iParticle = 0;
     iParticle < fParticles.size();
     iParticle++)
{
  isUsed[iParticle] = false;
  deleteCandidate[iParticle] = false;
  bestMother[iParticle] = -1;
}
```

Listing 2.1: Vector initialization within the default mother particle competition in `KFParticleTopoReconstructor::SelectParticleCandidates()` by [47].

After initialization, all particles are first checked if they are considered for the competition. This is done by `UseParticleInCompetition(PDGCode)` that returns true, if the PDG code of the particle matches with listed codes within the method. In the existing competition, several particles are investigated, the method returns true for the PDG codes 310 (K_s), 3122 (Λ), and others. Within this method, all other particles are skipped for the competition processed later on. After that, an iteration over all reconstructed primary vertices is done and a χ^2-cut is applied, to check if the particle's production vertex is close to the primary vertex, which in turn would indicate that it is a primary particle. All secondary particles are flagged for deletion, so that only primary particles become selected for competition (see Listing 2.2).

```
for (unsigned int iParticle = 0;
     iParticle < fParticles.size();
     iParticle++)
{
  if (!UseParticleInCompetition(fParticles[iParticle].GetPDG())) continue;

  bool isSecondary = 1;

  for (int iPV = 0; iPV < NPrimaryVertices(); iPV++) {
    KFParticle tmp = fParticles[iParticle];
    tmp.SetProductionVertex(GetPrimVertex(iPV));

    if (tmp.Chi2() / tmp.NDF() < 5) isSecondary = 0;
  }

  if (isSecondary) deleteCandidate[iParticle] = true;
}
```

Listing 2.2: Removing all secondary-track particles for the mother particle competition in `KFParticleTopoReconstructor::SelectParticleCandidates()` by [47].

The KFPF particle competition only considers primary particles because they are the initial particles produced in the collision and have not undergone any interaction with the detector material. Primary particles can be accurately measured and identified by the detector, which makes them a reliable source for reconstructing mother particles. In contrast, secondary particles, which are produced by the decay of primary particles, may interact with the detector material and leave only partial or incomplete information about the properties of the original mother particle. Furthermore, by using only primary particles for the competition, the KFPF package can improve the accuracy of the reconstructed mother particles and reduce the potential for misidentifying rare short-lived particles.

Finally, the competition between all primary-particles begins. All particles flagged for deletion are skipped, as well as all particles that are not listed for the competition. Then, for each candidate, the absolute mass difference `dm1` to the known mass distribution peak is calculated (Listing 2.3, line 9). After that, all possible competition opponents are checked: Here, again, if flagged for deletion or if not selected for competition, they are skipped. For each opponent, the absolute mass difference `dm2` to the opponent's known mass distribution peak is calculated (Listing 2.3, line 19). As a next step, daughters are compared because the opponent should share both daughters. For the decays of K_s and Λ, since both decays raise π^-, the shared daughter is π^-. Now, both particles' mass values are checked if at least one of them is within the 3σ area around the peak (Listing 2.3, line 22). The final competition is done by the mass difference to the peak. The particle that is closer to its own peak, wins the competition, the other one is flagged for deletion (Listing 2.3, lines 23-32).

```
for (unsigned int iParticle = 0;
     iParticle < fParticles.size();
     iParticle++)
{
  if (deleteCandidate[iParticle])
    continue;

  if (!UseParticleInCompetition(fParticles[iParticle].GetPDG()))
    continue;


  float mass, massSigma;
  fParticles[iParticle].GetMass(mass, massSigma);
  float massPDG, massPDGSigma;
  KFParticleDatabase::Instance()->GetMotherMass(fParticles[iParticle].GetPDG(), massPDG, massPDGSigma);
  float dm1 = fabs(mass - massPDG) / massPDGSigma;

  for (unsigned int jParticle = iParticle + 1;
       jParticle < fParticles.size();
       jParticle++)
  {
    if (deleteCandidate[jParticle])
      continue;

    if (!UseParticleInCompetition(fParticles[jParticle].GetPDG()))
      continue;


    fParticles[jParticle].GetMass(mass, massSigma);
    KFParticleDatabase::Instance()->GetMotherMass(fParticles[jParticle].GetPDG(), massPDG, massPDGSigma);
    float dm2 = fabs(mass - massPDG) / massPDGSigma;
    if (!(fParticles[iParticle].DaughterIds()[0] == fParticles[jParticle].DaughterIds()[0] && fParticles[iParticle].DaughterIds()[1] == fParticles[jParticle].DaughterIds()[1]))
      continue;

    if (dm1 < 3.f || dm2 < 3.f) {
      if (dm1 < dm2) {
        deleteCandidate[jParticle] = true;
        bestMother[fParticles[iParticle].DaughterIds()[0]] = iParticle;
        bestMother[fParticles[iParticle].DaughterIds()[1]] = iParticle;
      }
      else {
        deleteCandidate[iParticle] = true;
        bestMother[fParticles[iParticle].DaughterIds()[0]] = jParticle;
        bestMother[fParticles[iParticle].DaughterIds()[1]] = jParticle;
        break;
      }
    }
  }
}
```

Listing 2.3: Particle competition of primary-particle candidates with shared daughters in `KFParticleTopoReconstructor::SelectParticleCandidates()` by [47].

After that part of the competition where K_s and Λ are cleaned as described, the existing competition includes further clean-up methods, for instance, for γ-decays. Since the presented neural networks will not perform the following clean-ups, the source code and procedure is not important in detail. However, they will place a role in evaluating the results.

2.3.2 Performance Measurements in the KFPF

As mentioned before, K_s-mesons and Λ-hyperons are investigated in particular. To measure the performance of all competition approaches, KFPF provides several plots. In this thesis, the histograms for mass distributions of total spectra, signal, background and ghosts are used. In these histograms, the x-axis represents the mass bins in GeV$/c^2$, and the y-axis shows the number of entries per mass bin. Generally, the relative differences are evaluated, as there are usually more Λ occurrences compared to K_s, but both particles should be identified with the same good quality.

All particles that are reconstructed correctly, are known as *signal*. Particles that were misclassified contribute to noise in the signal of other particles, which is called *background*. Furthermore, some particles that are reconstructed did not exist in the generated data — these are called *ghosts*. Since there are always small inaccuracies in calculations or measurements due to finite detector resolutions, the results will most likely never be flawless. Nevertheless, algorithm packages are developed to perform as well as possible to provide physicists high quality physics analysis.

Beside the already mentioned histograms, the signal-background ratio, significance and Armenteros-Podolanski plots are analyzed. The Armenteros-Podolanski plots are 3-dimensional scatter plots, where the color of a bin visualizes the number of entries like a heat-map, the x-axis represents the longitudinal momentum asymmetry α, y-axis the transverse momentum p_t of the oppositely charged decay products [48]. The longitudinal momentum asymmetry α is a measure of the difference in momentum of two daughter particles along the direction of the mother particle's momentum, divided by the sum of their momenta. Thus, $\alpha = (p_L^+ - p_L^-)/(p_L^+ + p_L^-)$, where p_L^+ and p_L^- are the longitudinal momenta of daughters. The transversal momentum refers to the momentum component of particles perpendicular to the direction of the beam axis.

The signal-background ratio (S/B-ratio) provides an indicator for the proportions of signal in comparison to background. A S/B-ratio of 1 indicates that for the investigated spectra, the number of background entries is as large as the signal entries. Obviously, such a low S/B-ratio should be avoided. The significance S/$\sqrt{\text{S/B}}$ measures the peak produced by the signal in comparison to the fluctuations in the background. In general, this is an important indicator in the analysis, since new particles might be hidden in the background. A significance of 1 indicates that the signal is not distinguishable compared to the background, whereas a significance of 5 is considered as a threshold that the signal is likely to be a real signal, and not produced by background fluctuations. Λ and K_s, however, are well studied. Their peaks are known, and the algorithms are already well performing on finding them. Hence, their significance values will be clearly higher even without competition.

The selected histograms and ratios should be sufficient to perform an evaluation of the presented approaches.

2.3.3 KFPF Performance: Existing Competition

For the existing method, a data set consisting of 13k events was used to evaluate the results. It is the same data set that will later be used to evaluate the performance of ANN4FLES within KFPF, to make the results as comparable as possible.

In Figure 2.8, one can see the total spectra results of the KFPF package for K_s and Λ without any competition (black) compared to the results with the already existing competition (green). It is shown in both cases, K_s and Λ, that the general amount of entries per mass bin is reduced significantly in the area beside the peaks. In this area, many particles survive other χ^2 cuts applied in KFPF, suggesting that these reconstructed particles are candidates for K_s and Λ. However, using the competition, one can see that many of these particles do not survive in a direct competition with another particle that matches similar criteria. Here, one of the particles is chosen as the best matching K_s or Λ and the other particle is rejected, what finally reduces the amount of falsely assumed K_s or Λ particles.

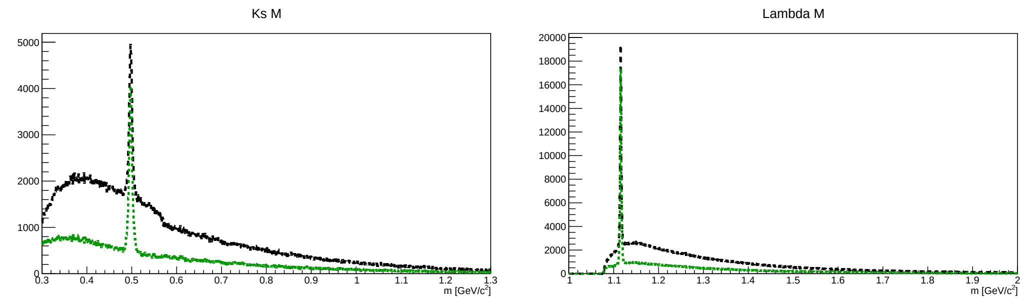


Figure 2.8: Histograms of total spectra for K_s and Λ masses. Comparison of existing mother particle competition (green) and no competition (black).

Investigating the signal histograms in Figure 2.9, in both cases a slight reduction of signal is indicated, using the competition. For Λ and K_s, that suggests that the competition not only rejects background particles, but in some cases signal particles are rejected by the competition as well. If not using any competition, these particles would not be rejected, but neither the background producing particles would be. However, the rejection of signal seems negligible in relation to the reduction of background.

Figure 2.10 confirms that the background is reduced by a huge amount, as these histograms show only the background. It is clearly visible that around the known mass of K_s (PDG mass $0.493\text{GeV}/c^2$) and Λ (PDG mass $1.116\text{GeV}/c^2$), the default competition has problems to identify background, as it is based on the comparison of mass distances to the peak. Therefore, for two particles that are both close to the peak, it is hard to distinguish the best matching mother particle based only on their peak distance, which leads to falsely selected mother particles. In case of K_s this problem seems worse, as the amount of entries in the background peak for the existing method

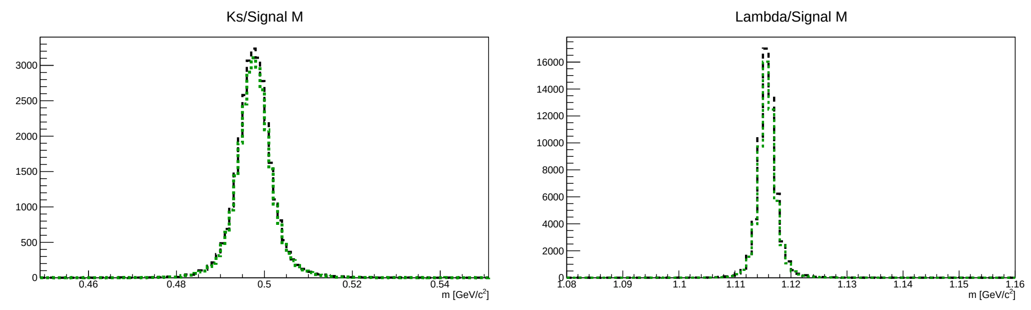


Figure 2.9: Histograms of signals for K_s and Λ masses. Comparison of existing mother particle competition (green) and no competition (black).

is almost as large as without competition. However, in general, the existing method improves the physics analysis quality by reducing background significantly.

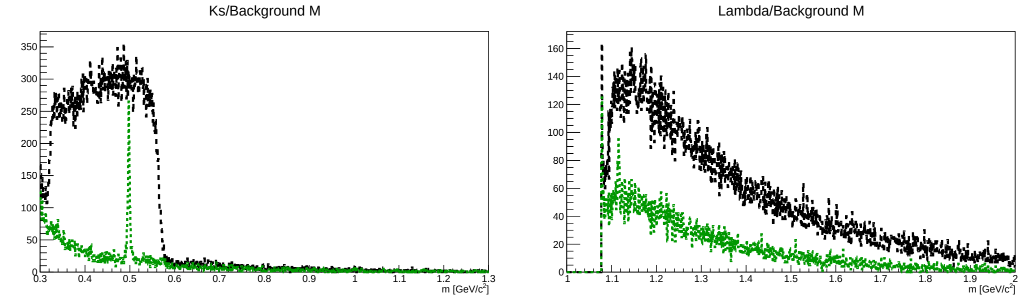


Figure 2.10: Histograms of background for K_s and Λ masses. Comparison of existing mother particle competition (green) and no competition (black).

In the ghost histogram (see Figure 2.11), it is shown that several ghost particles are partially responsible for the peak in the background after the competition, as they also show a peak around the PDG mass of K_s and Λ. However, in general, the competition reduces the number of ghosts significantly, therefore many mother particles that actually do not exist are removed by the competition, as they are not able to survive it. This behavior is expected, since all possible mother candidates are created first, and without competition, they are kept and all but one become ghosts, and therefore background.

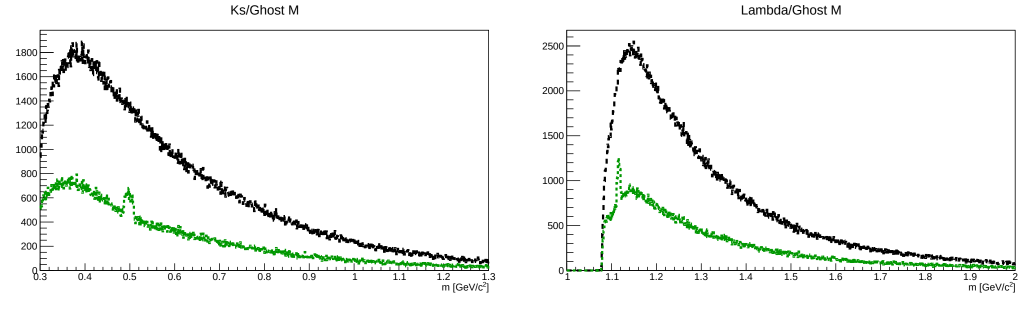


Figure 2.11: Histograms of ghosts for K_s and Λ masses. Comparison of existing mother particle competition (green) and no competition (black).

In Figure 2.12, the Armenteros-Podolanski plots are shown for K_s. On the left side, the plot is created without competition and two patterns are clearly visible. An arch ($= K_s$) over a large range of α within transverse momenta of $0.1\text{GeV}/c$ and $0.2\text{GeV}/c$. On the other hand, even more entries per bin illustrate another arch ($= \Lambda$) at α between 0.5 and approximately 0.82 with transverse momenta below $0.1\text{GeV}/c$. In the right plot, after the competition, the smaller arch is blurred out with a smaller amount of entries, whereas the wide arch is visible more clearly. Overall, the number of entries is more than halved. All together, it suggests that the competition removed background significantly, visualizing the wider arch of K_s more clearly, whereas many background entries in the range of Λ and the other blue areas were removed.

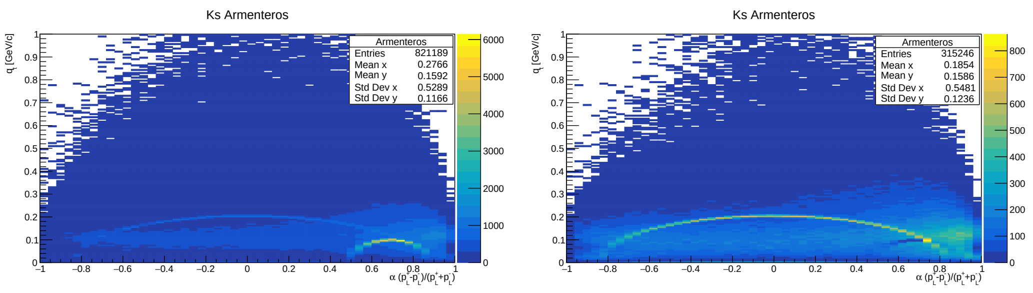


Figure 2.12: Armenteros-Podolanski plots with total spectra for K_s without (left) and with competiton (right).

For Λ (see Figure 2.13). The left plot without competition looks similar to the corresponding Armenteros-Podolanski plot of K_s. That suggests that both particles interact as background in the respective other particle's spectrum. Here, using competition, the arch of Λ is visible more clearly, whereas the arch of K_s has become invisible within the general background of Λ (blue area). It indicates that the background produced by K_s in the spectrum of Λ is reduced significantly by the existing competition.

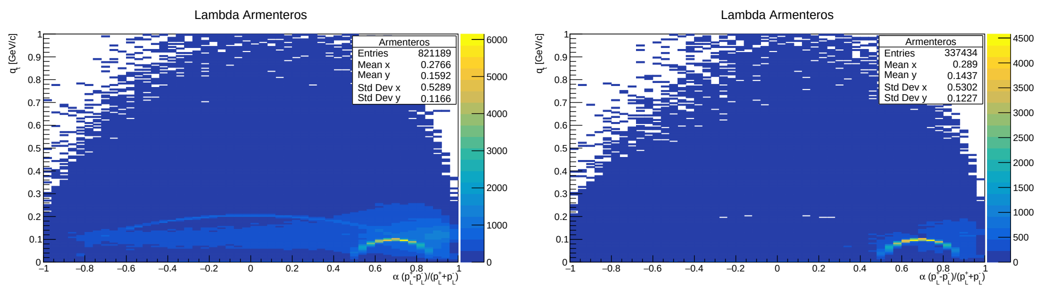


Figure 2.13: Armenteros-Podolanski plots for Λ without (left) and with competiton (right).

In general, the distribution visualized in the Armenteros-Podolanski plots can be explained by the decay products of the respective particles. The decay products of K_s are π^+ and π^-, both pions with the same mass but different charge. Since their mass is equally distributed when K_s decays, their momenta are also distributed symmetrically on average [23]. Therefore, they produce a symmetric arch in range $\alpha = -0.8$ to $\alpha = 0.8$. For Λ, however, the decay into proton p and pion π^- creates two daughter particles with different masses. Since the proton's mass is larger, on average it takes a larger part of the momentum compared to pion. Therefore, Λ's arch is placed on the right side of the plot.

Whereas all of these histograms and plots already indicate the performance of the competition, there are two ratios that have to be discussed: signal-background ratio and significance. In Figure 2.14, histograms of invariant masses for $K_s = \pi^+\pi^-$ and $\Lambda = \mathrm{p}\pi^-$ are shown. Both histograms show clearly the peaks of K_s and Λ centered at their known mass value.

The signal-background ratio for K_s of 1.08 suggests, that the amount of signal is almost as large as the amount of background, even though the background is usually distributed at larger distances from the peak. However, this amount of background will have negative impact on physics analysis and should be generally avoided. In case of Λ, the S/B ratio is better, as the amount of signal compared to background is approximately 3.56 times larger. This suggests that Λ is easier to separate from other particles than K_s.

The significance is a ratio to measure the peak-size compared to the fluctuation in the background. Both significance values are by far larger than 5 and, hence, one can argue with certainty that the peaks created by signal are different compared to the smaller peaks created by background fluctuation. A significance larger than 5 is considered as a true signal peak that can not be disregarded. In the case of K_s, without competition, the significance is 115 and for Λ 200, indicating that both signals are considered very unlikely to be produced by background fluctuation. Even without competition, the signals are clearly visible.

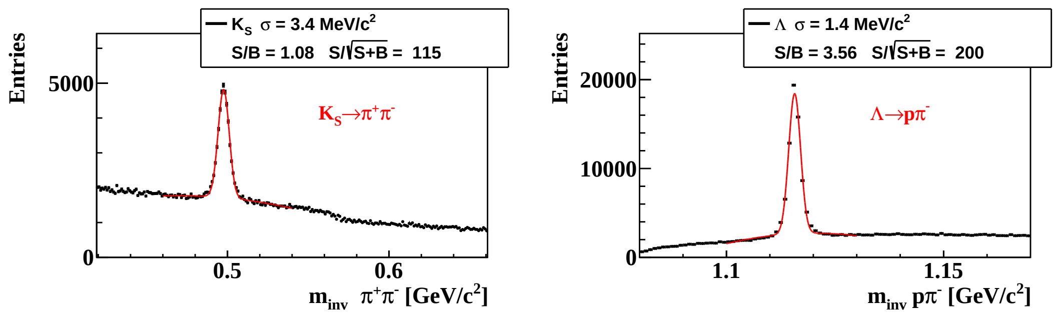


Figure 2.14: Invariant mass distributions of $K_s = \pi^+\pi^-$ and $\Lambda = \mathrm{p}\pi^-$ for KFPF without competition, including signal-background ratio and significance.

In Figure 2.15, the same plots are visualized, but after the competition. In both cases, the signal-background ratio is more than doubled. That confirms, that the competition rejected more background particles than rejecting signal particles. Here, the significance is also increased by approximately 30% in case of K_s and 6% in case of Λ. Both are indicating that the competition in general improves the performance of the KFPF package in regard to offering high quality data for physics analysis.

Summarized, the existing competition can be considered as a well-performing approach to reduce the background of K_s and Λ respectively. However, in the presented analysis, no other particles are considered. While the existing competition has been shown to be effective in reducing the background of K_s and Λ, the inclusion

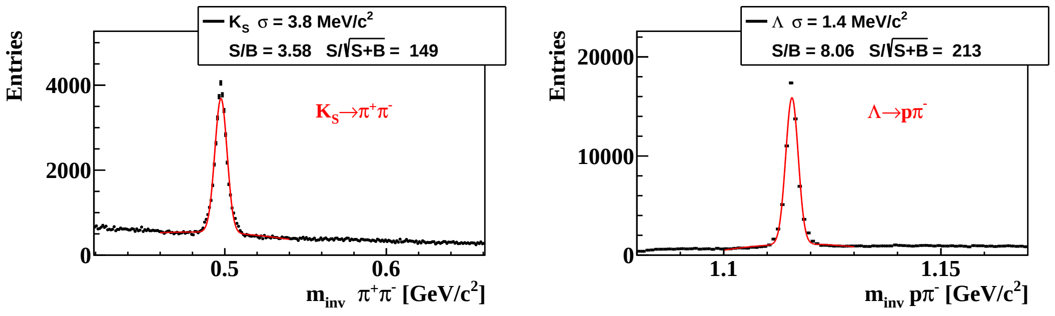


Figure 2.15: Invariant mass distributions of $K_s = \pi^+\pi^-$ and $\Lambda = \mathrm{p}\pi^-$ for KFPF with existing competition, including signal-background ratio and significance.

of other particles may lead to a more comprehensive and accurate evaluation of the competition's performance. By considering a wider range of particles, it may be possible to identify potential limitations in the competition that were not detected in this work. Nevertheless, if K_s and Λ are the particles of interest, the quality of physics analysis can be improved by enabling the competition.

Chapter 3

Neural Networks and Deep Learning

In recent years, the application of artificial neural networks has become very popular to solve tasks, especially in the areas of regression or classification problems. On the one hand, the hardware has become better and especially the fast and parallel tensor calculations on Graphical Processing Units (GPUs) make them very attractive to utilize. Furthermore, state of the art GPUs also include units that are designed for neural network calculations, since they are used to improve graphical performances of the graphics card itself (e.g. NVidia RTX with DLSS [49]). In non-private usages for workstations, there are GPUs (e.g. NVidia RTX A6000 [50]) designed in a way to increase the performance of ML algorithms even further by taking advantage of high performance tensor cores and GPU links that connect GPUs efficiently for more throughput. On the other hand, algorithms and the understanding of machine learning improved over time.

In Machine Learning (ML), a subset of artificial intelligence, there are several models to provide computers the ability to learn. Algorithms are used to allow computers to learn specific tasks by updating a set of parameters based on the machine's decisions or predictions, without explicitly programming the solution. A famous model is the Artificial Neural Network (ANN) or often called Neural Network (NN): A set of algorithms combined to mimic a biological brain and its learning abilities based on mathematical rules.

Nowadays, Multi-Layer Perceptrons (MLPs) are often referred to as Neural Network, since the connected layers build a network of linked vertices that model a brain's synapses and neurons. However, (Artificial) Neural Network is a more general terminology why it is often used to describe both: a Multi-Layer Perceptron which tends to have a small amount of hidden layers and deeper Neural Networks, including ones with extensions such as for example convolution- or pooling-layers. In general, however, the training of a neural network should be seen more as a approximation function optimization, rather than a brain simulation [51].

Contrary to the commonly flat MLPs, the training of deep neural networks containing many layers is referred to as Deep Learning (DL). Hence, DL is a subset of neural networks. Whereas the exact amount of layers for a deep network is not specified, usually more than two hidden layers are considered deep, but the amount may exceed even more than 1,000 layers - depending on the use case [52]. The depth of a neural network usually includes all layers, and therefore, for example in a Convolutional Neural Network, the convolution- and pooling-layers are counted too.

In 2016, Goodfellow [51, p. 167] described a Neural Network as a function composition, where each layer is described as a mathematical function.

$$N(\mathbf{x}) = N^{(n-1)}(N^{(n-2)}(...(N^{(0)}(\mathbf{x}))...)) \tag{3.1}$$

Supposing regulations such as activation functions, dropout and batch normalization as layers, this definition allows the description of many - if not all - neural networks. However, it is considered uncommon to count regulations into the network's depth. Nevertheless, using this definition, we can define a neural network and its behavior by the definition of each sub-function $N^{(i)}$.

In fact, this definition describes well what a neural network does: approximating a mathematical function. The network's weights are trained for statistical generalization. Starting with a random[1] initialization of weights, the network begins in an underfitted state. Any input results in a random output, based on the initialized weights. During training, after each input instance, the network adjusts its neuron's connections in such a manner, that for the given input, the inferred output is closer to the correct result than before. The more examples to train the network exist, the more general the learned features of the input data are.

3.1 Perceptron: The Smallest Neural Network

The probably most fundamental concept of neural networks, the Perceptron, was presented by Rosenblatt [53] in 1957. Although this first model was criticized for not being able to solve XOR, a simple logical gate for an exclusive disjunction, it builds the foundation of modern neural networks. Whereas the XOR-problem itself was actually a direct consequence of the problem of linear separability, it was proven in 1986 that a Multi-Layer Perceptron (MLP) is capable of solving non-linear problems, and therefore the XOR-problem [54].

In a perceptron, the neuron's output is defined by the weighted sum of inputs plus a bias value. Thus, using the definition by Goodfellow, we can describe a simple perceptron by

$$\begin{aligned} N(\mathbf{x}) &= N^{(0)}(\mathbf{x}) && (3.2)\\ &= b + \sum_{0}^{n-1} w_{i,j} x_i && (3.3) \end{aligned}$$

If the output value is larger than 0, the neuron is active and fires, otherwise it is inactive. The bias value b is a parameter independent of the input, but also learned during training and therefore adjusted similar to the weights w_i. The optimal slope to infer the classes correctly can be learned by adjusting weights, whereas the bias allows to shift the output.

[1]Randomness is crucial for breaking symmetry in weight updates. Otherwise, the network is not able to learn correctly.

3.1.1 Problem of Linear Separability

Generally, the more complex the architecture is, the harder it is to bring all parameters (most likely weights) in harmony to solve the task efficiently. From another point of view, some problems require more complex models to be solved at all. One of the problems that can occur with models that are too simple is the linear separability. In a classification task of c classes, the network learns up to $(c-1)$-many hyperplanes in the c-dimensional output space that separate areas corresponding to each class. An inferred output will then end up in one of these areas and as a consequence the corresponding class is predicted.

The OR-problem is linear separable, and therefore solvable by a simple single-layer perceptron with 2 input values and 1 output value. The task: Given a binary vector $x \in \{0,1\}^2$, return 1 if at least one of both values is 1, 0 otherwise. The respective perceptron's function can be described as:

$$N(\mathbf{x}) = w_0 x_i + w_1 x_1 + b \tag{3.4}$$

with weights w_i, bias value b and input values x_i.

One could easily argue that for $w_0 = w_1 = 1$ and $b = -0.5$, output $N(\mathbf{x}) = -0.5$ is only possible if $x_0 = x_1 = 0$. In all other cases, $N(\mathbf{x}) \geq 0.5$. Asking if $N(\mathbf{x}) > 0$ and therefore if the neuron fires, we have solved the problem. The 2-dimensional hyperplane (here line) that is now separating the two classes $(0,1)$, can be written as:

$$N(\mathbf{x}) \overset{!}{=} 0 \tag{3.5}$$

$$w_0 x_0 + w_1 x_1 + b \overset{!}{=} 0 \tag{3.6}$$

$$1 \cdot x_0 + 1 \cdot x_1 - 0.5 \overset{!}{=} 0 \tag{3.7}$$

$$x_0 = -x_1 + 0.5 \tag{3.8}$$

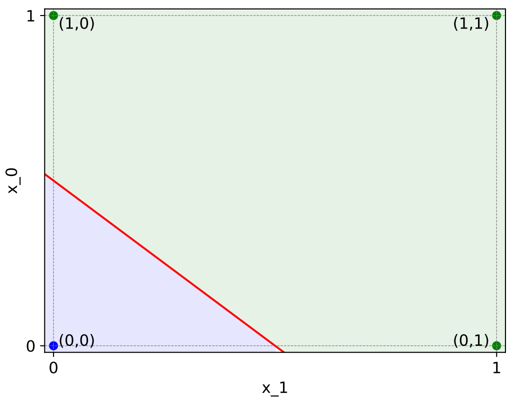


Figure 3.1: Visualization of the 2-dimensional hyperplane in red (Equation 3.8), separating the two classes 0 (blue area) and 1 (green area). Input pairs $(1,0), (0,1), (1,1)$ are cases where logical $x_1 \vee x_2$ should return 1. All these cases are above the red classification line. Pair $(0,0)$ case should return 0 and is located below the line. Therefore, all possible cases are classified correctly.

On the other hand, the XOR-problem is probably the most famous problem that is not solvable using a simple perceptron as defined before (Equation 3.4), as the solution is not linear separable, even if the problem itself is solved simply using other hardware or software solutions. The task: Given a binary vector $x \in \{0,1\}^2$, return 1 if $x_0 \neq x_1$, 0 otherwise. As we have seen before, the simple perceptron in Equation 3.4 is capable of learning a movable line. The cases, however, have changed such that $(0, 0)$ and $(1, 1)$ should both return 0, $(1, 0)$ and $(0, 1)$ output 1. Figure 3.1 illustrates, that it is not possible to separate both areas on the opposite sites of the plot (bottom left and top right), from the mid-cases (top left and bottom right) with a single line. The so called problem of linear separability.

3.2 Multi-Layer Perceptron: Breaking Linearity

As mentioned before, the Multi-Layer Perceptron (MLP) is able to solve the XOR problem by adding layers and non-linear activation functions; allowing to separate the outputs using multidimensional equations. Without non-linear activation functions, the whole MLP would remain a linear function. Hence, it would be possible to transform it into a single-layer perceptron [51, p. 171]. Applying an activation function $N^{(1)} = \varphi$ to our neuron's output, the extended perceptron can be described as:

$$N(\mathbf{x}) = N^{(1)}(N^{(0)}(\mathbf{x})) \tag{3.9}$$

$$= \varphi\left(b + \sum_{0}^{n-1} w_{i,j}x_i\right) \tag{3.10}$$

In this way, all neurons of a following layer can be calculated with their own related set of weights. As Figure 3.2 illustrates, using hidden units in a MLP, the following layer might be also connected to all neurons of another following layer, resulting again in weighted sums of the hidden layer neurons' outputs, activated afterwards. This can be repeated for arbitrary architectures, resulting in a more complex function $N(\mathbf{x})$.

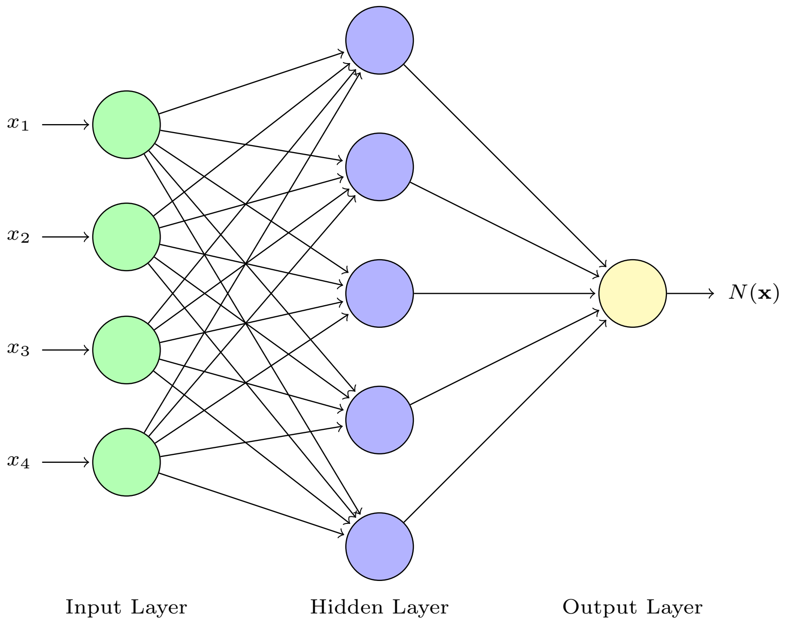


Figure 3.2: Illustration of a Multi-Layer Perceptron (MLP) with 4 input neurons forming the input layer (green), 5 neurons building one hidden layer (blue), and 1 output neuron in the output layer (yellow).

In classifiers, the output layer is often activated by either Logistic Sigmoid[2] or Softmax, depending on the number of classes classified. Both activation functions limit the neuron's output to a value between [0, 1]. In case of binary classification with a single neuron output layer, Logistic Sigmoid is chosen:

$$\varphi(x) = \sigma(x) = \frac{1}{1 + \exp(-x)}$$

This function provides a smooth transition between 0 and 1 with $\sigma(0) = 0.5$. Further, Sigmoid is converging to 1 for larger positive values and to 0 for larger negative values (see Figure 3.3). Compared to ArcTan, Logistic Sigmoid converges faster, whereas compared to TanH, Logistic Sigmoid converges slower, resulting in a sweet spot. Furthermore, it allows to interpret uncertainty in the classification visible directly by having output values[3] not equal to 0 or 1 but in between, whereas a classification with certainty is shown by output 0 or 1.

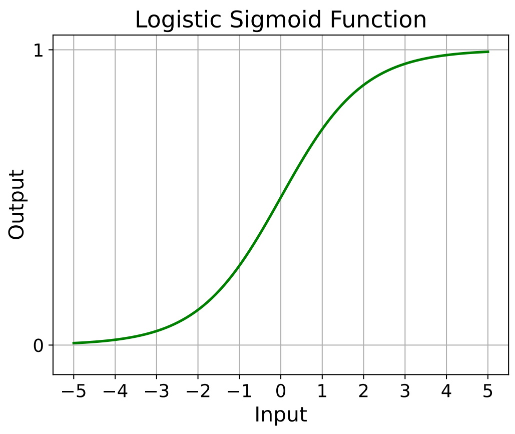


Figure 3.3: Plot of the Sigmoid activation function $\sigma(x) = \frac{1}{1+\exp(-x)}$. In binary classification tasks, this function is often in use when the output layer consists of a single neuron, inferring the class by output ≈ 0 or output ≈ 1 respectively. Within $x \in [-5, +5]$, uncertainty can be represented clearly in the neuron's output.

In n-class classification tasks, on the other hand, softmax is the way to go:

$$S_i(\mathbf{x}) = \frac{\exp(x_i)}{\sum_{j=0}^{n-1} \exp(x_j)}$$

This function is applied to the whole layer instead of a single neuron and does not only limit all neuron's values between [0, 1], but also limits the sum of all neurons to 1. Hence, one can interpret the output of a softmax activated layer as a probability distribution that reflects the likelihood of classes based on the given input.

3.3 Network Regularization Methods

Activation functions have a direct impact on the training results. First of all, they should be continuously differentiable everywhere in their domain. This requirement is a direct consequence of how learning rules for neural networks are defined. To update the network's parameters accordingly to the respective error thrown by a given

[2]Other Sigmoid functions are also valid, but the converging behavior and output space of Logistic Sigmoid makes it often the preferred choice.

[3]By definition, a converging function approaches, but never actually reaches the value it is converging to. However, in context of computer science, due to rounding errors using IEEE-754, the concept of ‘‘infinitely close’’ does not exist.

input, gradients are crucial, which are in fact calculated with the activation function's derivation.

Besides that, activation functions can lead to unexpected behavior. The Rectified Linear Unit (ReLU) function, is defined by

$$\text{ReLU}(x) = \max(0, x) \tag{3.11}$$

This function suppresses negative values, for positive values it behaves like the identity function $f(x) = x$. Although the derivative $\text{ReLU}'(0)$ is undefined, the function is one of the popular ones for hidden layer activation, used by several architectures [55, 56, 57]. For gradient calculation, $\text{ReLU}'(0) = 0$ is reasonable, as $\text{ReLU}(0) = 0$ and therefore the neuron would not have any impact to the following layers and the final result, therefore any weight update is unexpected anyway. However, ReLU provides non-saturating positive values and therefore can lead to better learning performance than sigmoid functions in the hidden layers that are limited in their output range. The problem with ReLU is, that it might lead to dead neurons: Assuming the input to a ReLU neuron is always negative, or a large bias neuron is dragging the neuron's value into negative range, its derivative value is always 0, and hence, there are no weight updates for this neuron [58]. In fact, the neuron has no impact and is considered dead. A possible solution for that problem is Leaky ReLU, that works similar but with a negative slope and a non-zero derivation everywhere[4] in its domain, avoiding neurons that are not trained entirely.

However, in general, activation functions are depending on the network's input and use case. ReLU, for example, was investigated and compared to several modified ReLU-functions, such as Leaky ReLU or Parametric ReLU, that can avoid neuron's inactivity. There, it was shown that negative non-zero variants of ReLU performed better [59]. In the present thesis, LReLU activation is used for hidden layers. Nevertheless, activation functions still belong to ongoing research in the field of Deep Learning [60] and newer activation functions are discovered continuously, such as Swish [61] and P-Swish [62].

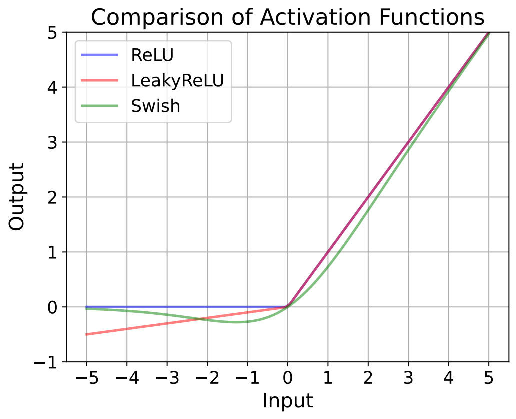


Figure 3.4: Visualizations of ReLU-, LReLU- and Swish-function, which are popular for hidden layer activation.

Beside activation functions, other regularization techniques are often in use. The most important methods are dropout, batch normalization, early stopping, as well as L1- and L2-regularization. Most of these regulators try to avoid a major learning problem: *overfitting* [63]. A neural network that is not or only weakly trained can be

[4]Assuming $\text{LReLU}'(0) \neq 0$ by implementation.

considered as *underfitted*, because its parameters are not tuned enough to provide a function approximation that models the data accurately. Contrary, a network that is trained too strongly on a specific data set, it is considered as overfitted. On the one hand, repetitions in training increase the accuracy of the training data set, since each instance of input adjusts the parameters such that next time it is provided to the network it will be recognized better. However, repeating the process too often will generally lead to a loss of generality. The network will then begin to train features of the data that are only specific for the training data set, but should not be considered as important when classifying the data in general - this is often referred to as learning *noise* [63, 64].

Using dropout, neurons are randomly disabled during training (see Figure 3.5) whereas other neurons are scaled up [65]. The probability to disable a neuron is a hyper-parameter that can be set in many neural network implementations. The up-scaling factor for active neurons is then set anti-proportional to the probability of dropping p: usually $1/(1-p)$. In this way, neurons are trained randomly, such that each neuron will learn different features to reduce co-adaption, and therefore, overfitting [65]. The technique can be applied layer-wise, dropped neurons are usually selected per instance or per mini-batch of data.

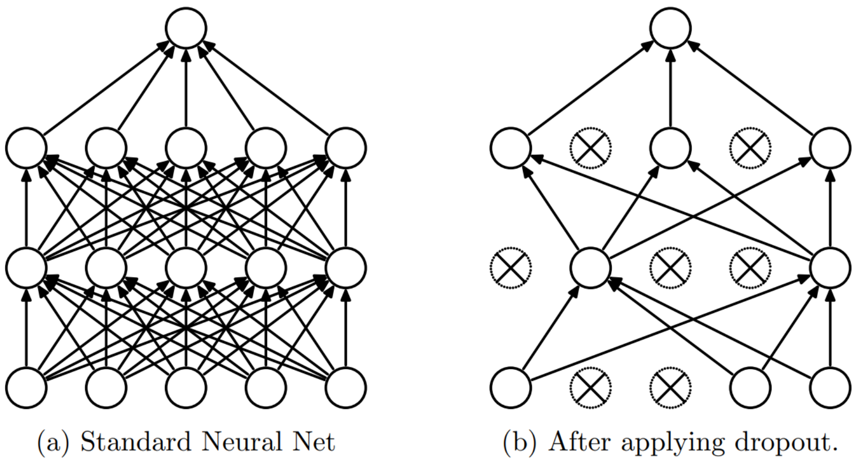


Figure 3.5: Visualization of dropout. On the left side, a standard fully connected neural network with two hidden layers is shown. On the right side, the network's state after applying dropout in the input- and hidden-layers is shown. Crossed neurons are dropped. [65]

Early-stopping is a technique that in fact stops the training, when the network starts to overfit [66]. An indicator for overfitting is the accuracy and loss of the validation (or test) set. Here, unseen data is used, and therefore it provides insights in the network's performance on general data. A decreasing accuracy or rising loss for the validation (test) set usually indicates that the network starts to learn the noise of training data. Then, early stopping can help to stop the training and keep the network in a state where generality is given. Depending on the exact implementation, one could reset the network to the best-performing state.

The regulators L1 and L2, also called *lasso* [67] and *ridge* [68], add a penalty to the weights. In particular, L1-regularization penalizes the absolute value of weights, such that some weights will be set to 0, removing a connection within the network over time. This, in fact, reduces the model's complexity and can therefore help to avoid overfitting. The regulator L2 penalizes the squared values of weights, encouraging the network to learn smaller weights, and thus, also reducing the model's complexity to avoid overfitting.

The idea of batch normalization is that the data is normalized over the batch for each layer, such that the data that is flowing through the network is more similar in values, which can increase the training process drastically [69]. Whereas the other mentioned techniques are implemented in particular to reduce overfitting, batch normalization is primarily used to provide a fast training process in deep neural networks. Nevertheless, batch normalization can help to avoid overfitting, because it improves generalization [70]. However, the exact reasoning why batch normalization is effective is not studied entirely yet and still an ongoing research topic [71]. Furthermore, although batch normalization is assumed to offer several positive aspects for neural network training, it is considered to perform worse when applying it together with dropout [72].

Summarized, there are many techniques to increase a network's performance by regularization. It is a challenging task to find a set of techniques that suit to solve a specific problem, thus providing high accuracy classification results without losing generality. Since there is no simple rule of thumb to achieve the best results, finding a suiting network architecture requires experience and testing.

3.4 Learning Paradigms of Machine Learning

In ML, there are three major learning paradigms which also apply for neural networks: supervised learning, unsupervised learning or reinforcement learning are selected depending on the network application.

In *supervised learning*, at least during training phase, the exact true outcome of a given input has to be known such that the machine is able to learn based on an accurate feedback. One is able to calculate the network's error directly based on its inferred output compared to the expected true results. Therefore, small mistakes can be handled by slight adjustments whereas large errors may lead to heavier changes within the parameters - usually allowing a fast learning process.

This might be a good approach to classify images where the result is clear, but imagine a robotic hand trying to grab a pencil. It is a non-trivial task to describe how wrong a grab attempt is, but it is quite simple to tell if a grab was successful or not. This is the area of *reinforcement learning*, where the true output is somehow describable in the form of yes/no or good/bad feedback. Based on this, one can not calculate exact errors as it is done in supervised learning, but one can roughly evaluate the network's output such that it is able to update its parameters accordingly. In reinforcement learning, it is quite possible that the training and inference phase are overlapping or never ending. For instance, the set of possible grasps might be infinite, such that the robotic hand's network can learn of every try - even while it is already in use. This can be done if the feedback can be calculated automatically, for example if sensors recognize the robotic hand is holding an object and a camera is used to identify the correct object by some deterministic approach (e.g. QR-Code) or another neural network.

The area of *unsupervised learning* is a solution for problems where one is unable to provide any feedback, because either the true output is unknown or it is too hard to describe. In this case, the network tries to learn based on the input data only. Self-organizing maps are such neural networks that try to figure out which parts of input data do or do not correlate to other parts of input data. For example, analyzing texts about nature these maps could recognize that `water`, `fish` and `whale` often appear together, which strengthens their respective nodes' connections (weights). As a result, the large weights pull the nodes of `water`, `fish` and `whale` together and places them in the same area on the map, whereas some other terms such as `desert` have a weak connection and are placed far away.

In this thesis, for the training of neural networks, simulated data is used and therefore the true outcomes are known (supervised learning). Due to the high quality of today's event simulations, possibilities are investigated to apply these pre-trained networks to real data in the future. But their performance has to be analyzed carefully: the exact outcomes of real particle collisions are unknown to a great extent and, therefore, errors can most likely be found during training only.

3.5 Supervised Learning: Neural Network Training

For supervised neural network training, the data set is divided into two or three data sets, depending on the approach. All approaches use a training data set to train the network's parameters (weights) accordingly to approximate the desired function. Then, a test set is used to evaluate the network's performance on unseen data that is not used during training. Sometimes, a third validation set is used to tune hyper-parameters, such as learning rate or dropout rate. In this thesis three data sets are used, training, validation and testing. A single training- and validation-phase is summarized as an epoch, which is repeated several times. The number of epochs is set as hyper-parameter.

The training of a neural network is an iterative adjustment of weights. The definition of Goodfellow, assuming a neural network is a function composition, where each layer is described as a mathematical function, is suitable to explain how this is done in neural networks:

$$N(\mathbf{x}) = N^{(n-1)}(N^{(n-2)}(...(N^{(0)}(\mathbf{x}))...)) \tag{3.12}$$

In the simplest case, the input tensor **x** contains one instance of input data of the training data set at a time, resulting in a single output tensor **y**. To train the network, the task is either to minimize the *loss* (sometimes error, costs) which is represented by a loss (error, cost) function, or to maximize the likelihood, that can also be defined by a function. Since a variant of gradient descent is used in the present work, the focus is set to minimizing the loss. Using gradient descent, the challenge is to find the global minimum for the loss function. However, finding the global minimum is a challenging task and in many cases a local minimum is sufficient for high accuracy[5] in the validation set. Further, since usually not all possible data can be provided for

[5]In classification, accuracy is defined as $A = \frac{TP+TN}{TP+TN+FP+FN}$ for TP=True Positives, TN=True Negatives, FP=False Positives, FN=False Negatives. It is an important criterion to measure the classification performance of a neural network.

training, finding the global minima might not be possible at all, due to the lack of freedom when moving in the output space.

The training is done by an algorithm called *backpropagation*, that uses an optimizer, such as standard gradient descent, with a learning rate η to calculate the new weights for each connection after evaluating the inference of an input instance and comparing it with the target value — the true outcome. For a better understanding, one could imagine that gradient descent tries to find a way downhill into a minima, e.g. in $\mathbb{R}^3$, and depending on the step size (learning rate) and the starting position (current weights plus input), this might work without jumping to the next hill (see Figure 3.6). However, since gradient descent updates only weights of neurons, it is essential to understand that its trajectory is confined to the parameter space (e.g. x-y plane in $\mathbb{R}^3$). The trajectory follows the direction of the steepest descent and can be visualized in the corresponding contour plot, where the shortness in distance to the next lower level is proportional to the steepness.

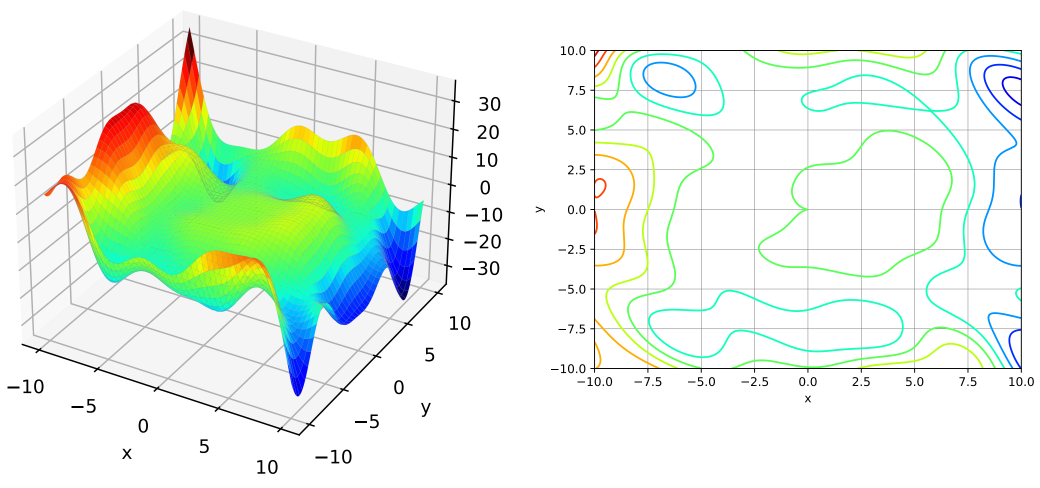


Figure 3.6: Visualization of a possible $\mathbb{R}^3$ output space that could represent a loss function's surface (left) with its corresponding contour-plot (right). Gradient descent tries to find a minima in the direction of steepest descent of the loss function. The trajectory of gradient descent is confined to the parameter space; here, x-y plane.

The general idea is to make steps through the output space, that are leading into a minimum. The direction is given by the gradients, as they provide information if one is moving up- or downhill. Therefore, the partial derivative of the loss function L w.r.t. a specific weight $w_{i,j}$ is required to update $w_{i,j}$ accordingly.

Remember the definition of the perceptron with an arbitrary activation function φ in the output layer neuron. We can calculate any neuron's output value o_j by:

$$o_j = \varphi\left(b_i + \sum_{0}^{n-1} w_{i,j} x_i\right) \tag{3.13}$$

The bias b_i is ignored for now. In ANN4FLES, the bias neuron is implemented as a neuron with a fixed output value of 1, where the actual trainable shift is implemented through the weights out into the next layer. Therefore, the bias is handled similar to any other neuron, beside the fact that it does not change its output value and has no connections going in. As a next step, the summation is substituted by function s_j:

$$s_j = \sum_{0}^{n-1} w_i x_i \tag{3.14}$$

The loss function is applied to the network's output $L(N(\mathbf{x}))$. Since $N(\mathbf{x})$ itself is a chain of functions, we use the chain rule twice to calculate the error with w.r.t. to $w_{i,j}$. First, let $o_j = N(\mathbf{x})$ such that o_j represents the output of the output neuron and, therefore, the output of the whole network. Then:

$$\frac{\partial L}{\partial w_{i,j}} = \frac{\partial L}{\partial o_j}\frac{\partial o_j}{w_{i,j}} = \frac{\partial L}{\partial o_j}\frac{\partial o_j}{s_j}\frac{\partial s_j}{w_{i,j}} \tag{3.15}$$

The last factor is trivial, since building the derivative of the sum w.r.t. $w_{i,j}$, all other weights are handled as constant factors and therefore thrown out. The remaining part is $w_{i,j}x_i$, such that:

$$\frac{\partial s_j}{w_{i,j}} = \frac{\partial}{\partial w_{i,j}} \sum_{0}^{n-1} w_{i,j} x_i = x_i \tag{3.16}$$

We substitute the other factors into the sensitivity δ_j. Depending on if the neuron j is an inner neuron (part of a hidden layer) or part of the output layer, δ_j is calculated differently: An inner neuron has impact to all neurons of the next layer, such that it's error can not be calculated directly, but indirectly through a summation over the sensitivities of the next layer. An output neuron's sensitivity can be calculated directly. Thus, we define δ_j as follows:

$$\delta_j = \frac{\partial L}{\partial o_j}\frac{\partial o_j}{\partial s_j} \tag{3.17}$$

For output layer neurons, both factors are trivial if loss- and activation-function are known. However, if j is an inner neuron, the derivative is less obvious due to the summation. Then:

$$\frac{\partial L}{\partial o_j} = \sum_{k \in K} \left(\frac{\partial L}{\partial s_k}\frac{\partial s_k}{\partial o_j} \right) \tag{3.18}$$

$$= \sum_{k \in K} \left(\frac{\partial L}{\partial o_k}\frac{\partial o_k}{\partial s_k}\frac{\partial s_k}{\partial o_j} \right) \tag{3.19}$$

$$= \sum_{k \in K} \left(\underbrace{\frac{\partial L}{\partial o_k}\frac{\partial o_k}{\partial s_k}}_{\delta_k} w_{j,k} \right) \tag{3.20}$$

Therefore, for target values **t** and output **o**, with loss function $L(\mathbf{t}, \mathbf{o})$ and activation function $\varphi(s)$:

$$\delta_j = \frac{\partial L}{\partial o_j}\frac{\partial o_j}{\partial s_j} = \begin{cases} \frac{\partial L(t_j, \varphi(s_j))}{\partial \varphi(s_j)} \frac{d\varphi(s_j)}{ds_j} & \text{if } j \text{ is an output layer neuron,} \\ \left(\sum_{k \in K} w_{j,k}\delta_k \right) \frac{d\varphi(s_j)}{ds_j} & \text{if } j \text{ is a hidden layer neuron.} \end{cases} \tag{3.21}$$

After calculating the gradients for each connection, the learning rule can be applied. For standard gradient descent, the learning rule is defined as:

$$\Delta w_{i,j} = -\eta \frac{\partial L}{\partial w_{i,j}} \tag{3.22}$$

Using this rule, the new weight results from:

$$w_{i,j}^{(\text{new})} = w_{i,j}^{(\text{old})} + \Delta w_{i,j} \tag{3.23}$$

In *batch gradient descent*, the standard (sometimes referred to as *vanilla*) gradient descent algorithm, the learning rule is applied after the whole data set is analyzed [73]. The gradients are summed over the whole data set. Therefore, each epoch the training consists of one weight update for each connection after seeing all data. This approach is usually (1) very slow, since neural networks' training often requires huge data sets and all information have to be seen before the machine learns anything about the data and (2) might not work at all for the full data set, if the data set does not fit into the computer's memory completely.

The *stochastic gradient descent* approach goes a different way by applying the learning rule after each instance of input [73]. Therefore, this approach learns step by step and the machine is updated with each new experience. This might lead to high fluctuation while learning new data, but compared to the full-batch approach, it allows to step out of local minima and find better solutions. Stochastic gradient descent can be regarded as a stochastic approximation of the gradient descent approach, as the sub-samples are trained randomly. Anyway, since each instance of data is updating the weights for this instance's best case, it often overshoots a good minimum for the full data set.

In *mini-batch gradient descent*, the focus is on finding better solutions by using samples of a given batch size [73]. Therefore, the learning rules are applied after each mini-batch, a subset of the whole data set. In this way, there exists another hyper-parameter to control the polarity of fast learning in the stochastic gradient descent approach versus the higher quality updates of applying rules only after seeing multiple data. In some use-cases, this approach is also extended by the already mentioned batch normalization, which increases the speed of the learning process usually even further [74].

Neural networks are often trained in mini-batches as it is considered faster and better performance-wise, applying their learning rules after each mini-batch. Then, in many implementations (e.g. PyTorch [75]) the tensor dimension is incremented, providing a new dimension for the mini-batch. From computer science perspective, mini-batch training allows to speed-up the training process even further by parallelizing it depending on the batch size. Although the gradients have to be summed up for each instance within a batch, the calculations itself are independent and therefore predestined for parallel Single-Instruction-Multiple-Data (SIMD) instructions. Further, the batch size simply adds a new dimension to the input tensor, keeping it possible to run on a GPU's tensor cores.

However, all three concepts are based on the gradient descent approach and the learning rate can be adjusted to increase or decrease the step size in each learning iteration. Over the years, these concepts were extended: modern architectures often use improved learning algorithms that use either look-ahead features (e.g. AWL [76]), momentum based weight updates (e.g. Nesterov [77]), adaptive learning rates (e.g. AdaDelta [78]) or a combination of them, usually as part of mini-batch training.

3.6 Information Theory and Cross-Entropy Loss

The network's performance is measured by a loss (or error, cost) function, that is then used to calculate the gradients for backpropagation. In classification, cross-entropy loss has become popular and there are several reasons. A first question one should ask is about what information is required to evaluate a network's output.

To evaluate loss functions, a neural network with two output neurons using softmax activation is discussed. In the binary case, where a 1-hot encoded vector is used as target values and a single class should be inferred with a given input, the class with the highest likelihood (based on softmax activation) is inferred. One could calculate the difference between output values and target values and use the difference as an indicator for the network's performance. It seems obvious, that the difference is an indicator for uncertainty. However, this indicator can be reasoned even more.

Goodfellow wrote:

''The basic intuition behind information theory is that learning that an unlikely event has occurred is more informative than learning that a likely event has occurred.'' [51, p. 71]

A low probability event contains a large amount of information, whereas high probability events are not surprising and therefore provide less information. A measure of information is the *self-information* or Shannon-information [79]:

$$h(x) = -\log(p(x)) \tag{3.24}$$

where a probability $p(x) = 1$ would provide 0 information and the amount of non-guaranteed events result in positive values. (see Figure 3.7)

An entropy of a random variable X describes the level of uncertainty inherent in the variable's possible outcome. In particular, it quantifies how surprising an event is by providing the average amount of self-information an observer would expect to gain about a random variable, when measuring it [79]. For discrete X, this uncertainty can be described as:

$$H\left(X\right) = -\sum_{x} p(x) \log\left(p(x)\right) \tag{3.25}$$

where a larger value of $H\left(X\right)$ describes higher uncertainty (see Figure 3.7).

Cross-entropy, however, is the entropy between two distributions. In supervised learning for classification, two distributions exist: one for the true outcomes **t** and one

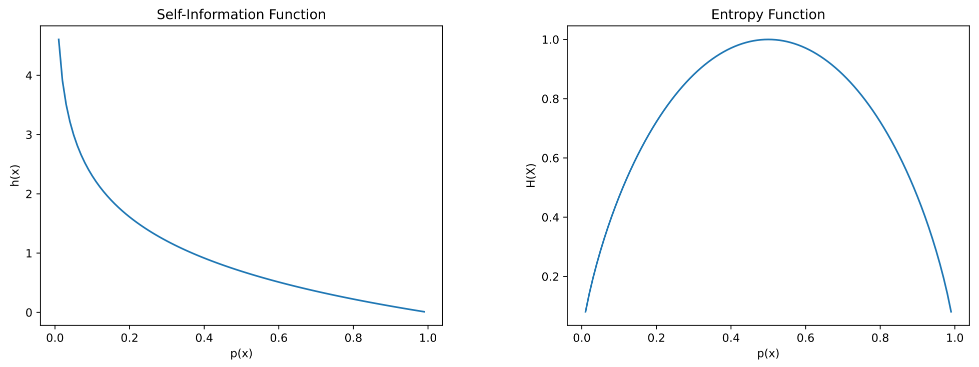


Figure 3.7: Visualization of the self-information function by Shannon (left) and the entropy as expected value of self-information.

for the inferred outcomes **o** by the neural network. As a loss function, cross-entropy is described as:

$$L_{\text{CE}}(\mathbf{t}, \mathbf{o}) = \sum_{c=0}^{n-1} t_c \log(o_c) \tag{3.26}$$

In case of binary classification, where only one label is 1 and all others are 0, this can be simplified to the binary cross-entropy loss BCE with c indicating the index of the true class:

$$L_{\text{BCE}}(t_c, o_c) = -t_c \log(o_c) \tag{3.27}$$

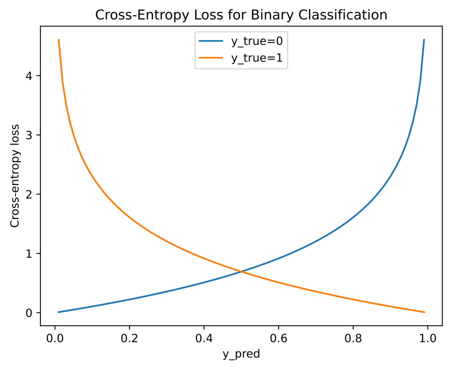


Figure 3.8: Visualization of the binary cross-entropy loss BCE. For each output neuron, the loss can be calculated, depending on if the neuron represents the true class (orange) or not (blue). Whereas small differences between prediction and true outcome lead to low loss values, the opposite case leads to an infinitely large error.

Back to softmax: Since softmax is a vector function that is applied to the whole output layer, we have to calculate the derivative $\frac{\partial oj}{\partial s_c}$ of j-th output w.r.t. its c-th input. Since the cases $j = c$ and $j \neq c$ have to be handled differently, we build the derivative

$j = c$ first:

$$\frac{\partial o_j}{\partial s_c} = \frac{\partial \frac{\exp(s_j)}{\sum_{i=1}^{S}\exp(s_i)}}{\partial s_c} \tag{3.28}$$
$$= \frac{\exp(s_j)\big(\sum_{i=1}^{S}\exp(s_i)\big) - \exp(s_j)\exp(s_c)}{\big(\sum_{i=1}^{S}\exp(s_i)\big)^2} \tag{3.29}$$
$$= \frac{\exp(s_j)}{\sum_{i=1}^{S}\exp(s_i)} \cdot \frac{\big(\sum_{i=1}^{S}\exp(s_i)\big) - \exp(s_c)}{\sum_{i=1}^{S}\exp(s_i)} \tag{3.30}$$
$$= \frac{\exp(s_j)}{\sum_{i=1}^{S}\exp(s_i)}\left(\frac{\sum_{i=1}^{S}\exp(s_i)}{\sum_{i=1}^{S}\exp(s_i)} - \frac{\exp(s_c)}{\sum_{i=1}^{S}\exp(s_i)}\right) \tag{3.31}$$
$$= o_j(1 - o_c) \tag{3.32}$$
$$\tag{3.33}$$

The case $j \neq c$ analog:

$$\frac{\partial o_j}{\partial s_c} = \frac{\partial \frac{\exp(s_j)}{\sum_{i=1}^{S}\exp(s_i)}}{\partial s_c} \tag{3.34}$$
$$= \frac{0 - \exp(s_c)\exp(s_j)}{\big(\sum_{i=1}^{S}\exp(s_i)\big)^2} \tag{3.35}$$
$$= -\frac{\exp(s_c)}{\sum_{i=1}^{S}\exp(s_i)} \cdot \frac{\exp(s_j)}{\sum_{i=1}^{S}\exp(s_i)} \tag{3.36}$$
$$= -o_c o_j \tag{3.37}$$

Building the derivative of cross-entropy loss, using chain rule for logarithm:

$$\frac{\partial L_{\text{CE}}(\mathbf{t}, \mathbf{o})}{\partial s_c} = \frac{\partial \sum_{c=0}^{n-1} t_j \log(o_j)}{\partial s_c} \tag{3.38}$$
$$= -\sum_{j=0}^{n-1} t_j \frac{\partial \log(o_j)}{\partial s_c} \tag{3.39}$$
$$= -\sum_{j=0}^{n-1} t_j \frac{\partial \log(o_j)}{\partial o_j} \cdot \frac{\partial o_j}{\partial s_c} \tag{3.40}$$
$$= -\sum_{j=0}^{n-1} t_j \frac{1}{o_j} \cdot \frac{\partial o_j}{\partial s_c} \tag{3.41}$$

Separating the case $j = c$ from the sum and inserting softmax derivatives for the specific cases:

$$\frac{\partial L_{\text{CE}}(\mathbf{t},\mathbf{o})}{\partial s_c} = \underbrace{-t_j \frac{1}{o_j}\frac{\partial o_j}{\partial s_c}}_{j=c} - \sum_{j\neq c} t_j \frac{1}{o_j} \cdot \frac{\partial o_j}{\partial s_c} \tag{3.42}$$

$$= -t_c \frac{1}{o_c} o_c(1-o_c) - \sum_{j\neq c} t_j \frac{1}{o_j} \cdot \frac{\partial o_j}{\partial s_c} \tag{3.43}$$

$$= -t_c(1-o_c) - \sum_{j\neq c} t_j \frac{1}{o_j} \cdot (-o_c o_j) \tag{3.44}$$

$$= -t_c(1-o_c) + \sum_{j\neq c} t_j o_c \tag{3.45}$$

$$= -t_c + t_c o_c + \sum_{j\neq c} t_j o_c \tag{3.46}$$

$$= -t_c + o_c\left(t_c + \sum_{j\neq c} t_j\right) \tag{3.47}$$

Since $\sum_j t_j = 1$ by definition because $\mathbf{t}$ is a 1-hot encoded vector, $t_c + \sum_{j\neq c} t_j = 1$, and therefore, $o_c \cdot 1 - t_c = o_c - t_c$, which is the difference between output value and target value for neuron c, as mentioned before.

Summarized, cross-entropy loss and softmax activation provide much information about the network's results and its certainty. Furthermore, to calculate the output layer's δ-values, cross-entropy and softmax can be simplified to a simple difference of output- and target values, making the combination predestined for many classification problems. Therefore, both are used in the output layer of the neural networks proposed by the present work.

3.7 ANN-Based Particle Identification in KFPF

Neural networks were already applied to the particle identification in the mother particle competition of KFPF. Previous work suggests that both competition approaches, the existing and their ANN-based approach, provide comparable results [23]. For performance measurement, histograms of total-, signal-, background- and ghost-mass distributions were used. For comparison, these results are recapitulated. Unfortunately, neither the exact neural network nor the root files to create the histograms and results are available, therefore, the presented results by [23] are shown.

However, previous work had two approaches, with and without MVD. As we compare only results with MVD enabled, only previously created plots using MVD are shown.

The neural network that was in use is a MLP (see Figure 3.9) trained on 10k PHSD simulated events, using the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm, which is a quasi-Newton method to find local minima of the loss function [80, 81]. The training set consisted of 7500 events, whereas the test set consisted of 2500 events, updating the weights over 500 iterations. Then, the best performing network in regard to training- and testing set accuracies was chosen. Beside the test set, the network's performance was evaluated within KFPF to see its classification performance within the package. The data set to evaluate the network includes 500k events. [23]

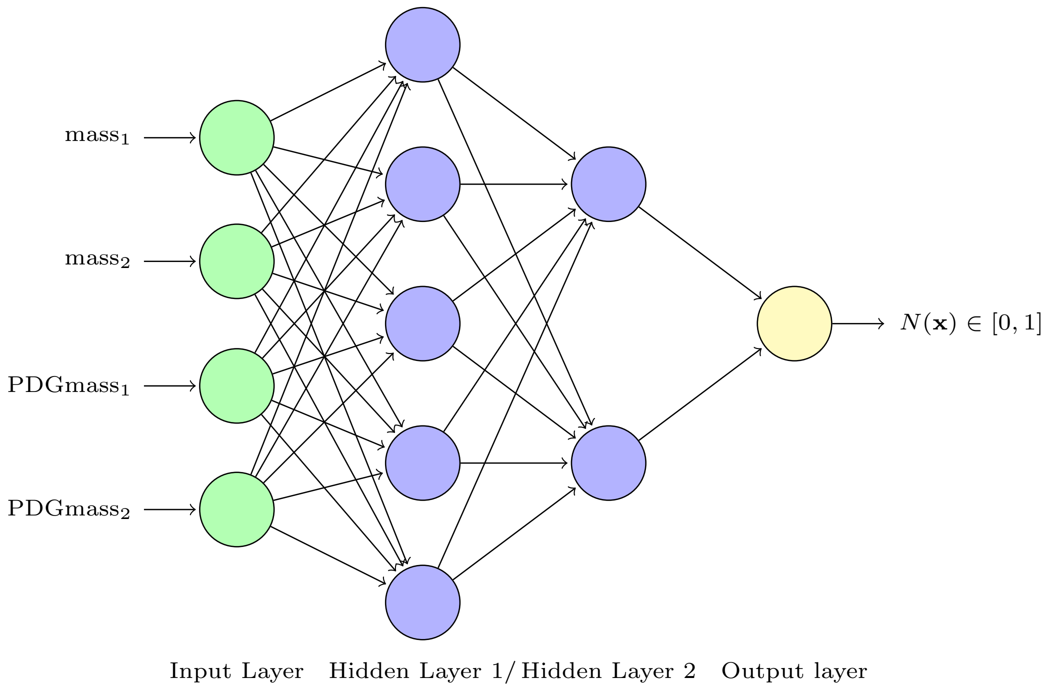


Figure 3.9: Neural network used for previous work [23], classifying the best fitting mother particle for π^- based on the mass and PDG mass values of both candidates.

In the histogram of total mass spectra (see Figure 3.10), one can see that the neural network approach (blue) performed slightly worse in comparison to the existing method (red). Over the whole range, one can see that the neural network based approach rejected less reconstructed particles than the existing competition, since it has more entries in almost every bin. However, in general that could also be due to less rejected signal in comparison to the existing method, but the other plots will show that it is not only the case.

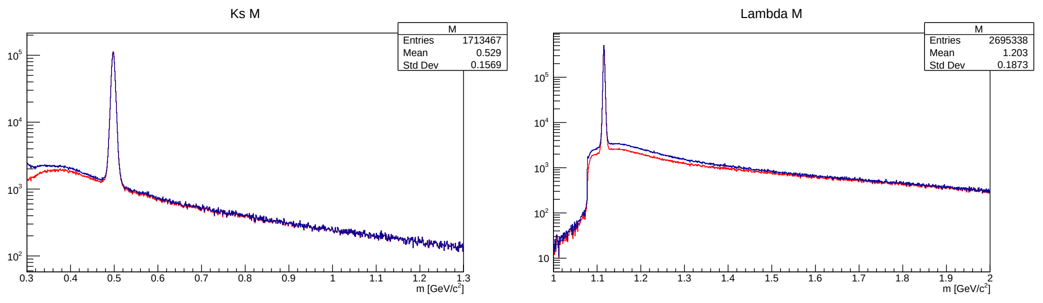


Figure 3.10: Total mass spectra of K_s (left) and Λ (right) using logarithmic scaling. These histograms were created by [23].

Figure 3.11 shows the reconstructed signal histograms. For Λ, the ANN approach of previous work seems to reconstruct more signal, as it has consistently slightly more or an equal amount of entries in the bins in comparison to the existing method.

For K_s it looks like that both approaches perform equally. However, since these plots do not use RBGa with an alpha lower than 1, one can only assume that the red line (existing approach) is always behind the blue line, but there might exist some areas where the red line is not existing, as there are no entries at all.

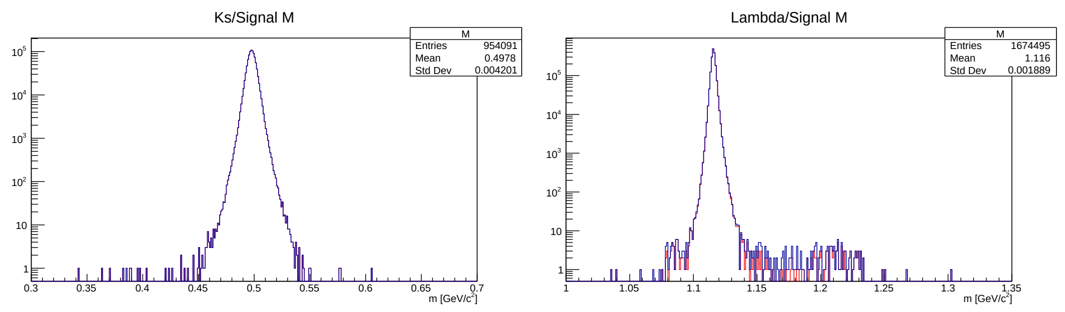


Figure 3.11: Reconstructed signal mass spectra of K_s (left) and Λ (right) using logarithmic scaling. These histograms were created by [23].

In Figure 3.12 the background of both competitions is shown. In the histogram of K_s, it is clearly visible in which area the neural network approach can shine: It is well performing on reducing the background at the mass distribution peak. However, it performs slightly worse for the general background with larger distance to the peak. For Λ, the neural network seems to reject less background particles over the whole range of the histogram. Here, contrary to K_s, within the peak area (PDG mass of Λ: $1.116\text{GeV}/c^2$), the neural network approach rejects less background particles in comparison to the existing method as well.

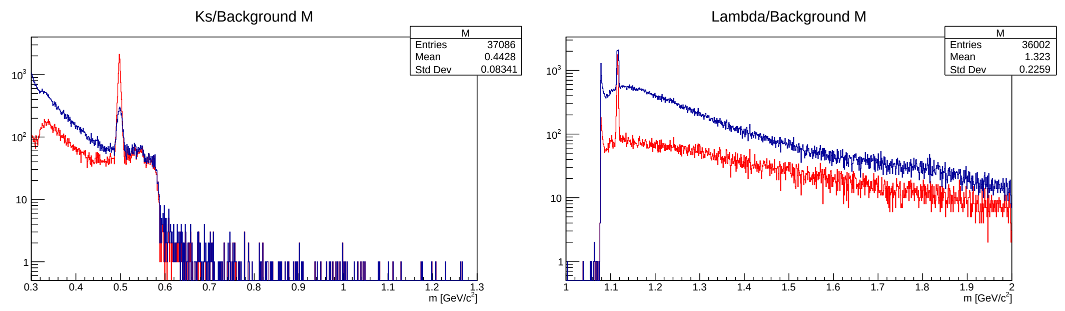


Figure 3.12: Reconstructed background mass spectra of K_s (left) and Λ (right) using logarithmic scaling. These histograms were created by [23].

Investigating the ghosts (see Figure 3.13), one can see where some of the increased background by the neural network approach originates. For Λ, over the whole range the number of ghosts is increased, indicating that the network approach rejects less ghost particles. For K_s, the illustration is similar: Beside the peak at K_s PDG mass of $0.493\text{GeV}/c^2$, the number of ghosts is increased.

In general, with the logarithmic scaling in mind, both competitions presented in this section, the existing method of KFPF and the ANN based approach by previous work offer comparable results as it was already mentioned by [23]. It was shown that a neural network can solve the task, which is usually done by KFPF's existing

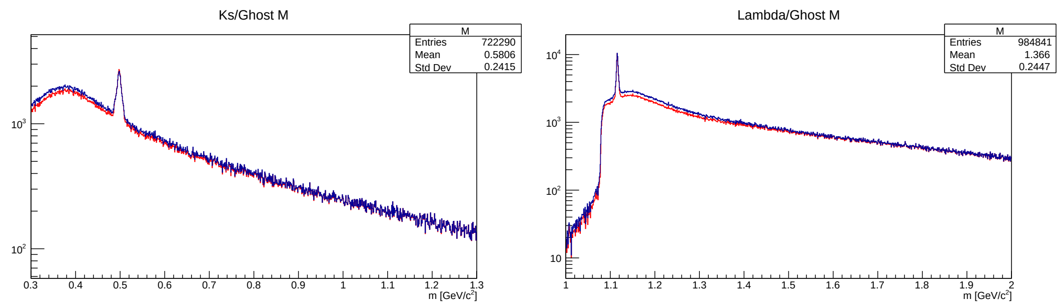


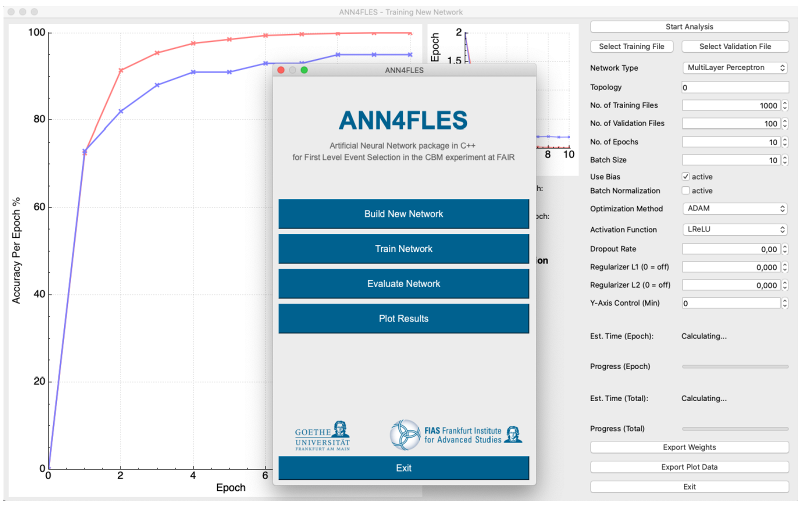
Figure 3.13: Reconstructed ghost mass spectra of K_s (left) and Λ (right) using logarithmic scaling. These histograms were created by [23].

competition.

Comparing the results of previous work in relation to the results presented in this thesis, care has to be taken that KFPF and FLES are complex packages. The exact results differ depending on the settings used, and not all settings of previous work are known in detail. Furthermore, in previous work, the PHSD model was used to create generated simulated data, whereas in the presented results the UrQMD model is underlying, and the numbers of events vary too. Both models should generate approximately equal results, but it is suggested to compare both models in context of the ANN-based approach. Therefore, however, the approach in this work tries to adapt and improve previous work to the best of knowledge and belief.

3.8 ANN4FLES: High Performance Neural Networks

The event selection in CBM requires a full event reconstruction that is processed on high performance computers. Although there are several user-friendly neural network toolkits such as PyTorch [82], Keras [83] or Tensorflow [84], a work group develops a high performance neural network package in C++ for applications in the First Level Event Selection called ANN4FLES [19, p. 161].


ANN4FLES - Training New Network
Accuracy Per Epoch %
Epoch
Epoch
ANN4FLES
ANN4FLES
Artificial Neural Network package in C++
for First Level Event Selection in the CBM experiment at FAIR
Build New Network
Train Network
Evaluate Network
Plot Results
GOETHE UNIVERSITÄT FRANKFURT AM MAIN
FIAS Frankfurt Institute for Advanced Studies
Exit
Start Analysis
Select Training File
Select Validation File
Export Weights
Export Plot Data
Exit

Network Type: MultiLayer Perceptron
Topology: 0
No. of Training Files: 1000
No. of Validation Files: 100
No. of Epochs: 10
Batch Size: 10
Use Bias: ✓ active
Batch Normalization: active
Optimization Method: ADAM
Activation Function: LReLU
Dropout Rate: 0,00
Regularizer L1 (0 = off): 0,000
Regularizer L2 (0 = off): 0,000
Y-Axis Control (Min): 0
Est. Time (Epoch): Calculating...
Progress (Epoch)
Est. Time (Total): Calculating...
Progress (Total)

Figure 3.14: The graphical user interface of ANN4FLES at its current state.

The development of an own package allows to adjust and design the source code in a way, that it is well aligned for the needs of the CBM experiment. However, the usage is not limited to CBM, as there are cooperations with other experiments (e.g. LHCb) and universities to build a platform together [85]. Furthermore, the package will be adjusted to provide all relevant information required for physicist, to interpret the results reasonably.

At its current state, the package provides multiple types of neural networks such as multi-layer perceptrons, convolutional neural networks, recurrent neural networks and graph neural networks. The architecture can be selected arbitrary, but, at current state, the network does not provide a network builder that can export full architectures into file, which is planned for the future. Then, the training process can be done without any programming knowledge via GUI, and, in case of CBM and high performance computers, the inference-only mode can be easily loaded via command line interface. However, if the network has to be included into existing C++ code, minor skills of programming are required to place it into the correct position. For the future, for classification it is planned to become analog to:

```cpp
// Initialization with architecture, settings and weights provided by
↪ files
Network network = Network(architecture, settings, weights);
...
// Infer predictions, returns class index
int pred = Network.Predict(input);
...

```

Listing 3.1: Example code on how ANN4FLES can be initialized and used within another package of the First Level Event Selection.

ANN4FLES has been compared against well-established neural network packages like PyTorch and performed comparable in regard to accuracy (see also Figure 4.3, Figure 4.13). Considering speedup, ANN4FLES in its current state can not be run on GPU. However, multi-threading and SIMD instructions are used to increase the network's performance, providing comparable results in regard to runtime on CPU.

In the present thesis, the GUI will be used to perform the neural networks training. Then, the trained weights of the network are exported to file, such that they can be imported within KFPF to have a fully functional neural network in inference-only mode.

Chapter 4

Deep Learning for Identification of Short-Lived Particles

In this chapter, the neural network approach using ANN4FLES of the present thesis is analyzed and discussed. First, a reproduction of previous work is aspired, thus, using a neural network based classifier for K_s and Λ based on the candidates' mass and PDG mass values. Then, since previous work suggested it as a follow-up approach, a neural network using mass plus transverse momenta is introduced and discussed.

4.1 Extraction of Training Data in KFPF

The extraction of training data for the neural network approach presented in this thesis is done by a modification of KFPF (see Figure 4.1). Supervised learning requires to know the true outcomes for each instance of input. In case of K_s/Λ-classification, for each pair of particles that compete against each other, the information about which particle is more likely to be the best fitting mother candidate is required. On real data, the following approach to extract training data is almost impossible, as there is no partitioning in signal and background. For real data, the true outcomes are predominantly unknown or only based on statistical evidence. However, using simulated generated data, information about particles that belong to signal or background is available.

Overall, 25k events generated by the UrQMD model were used in this thesis, 12k were used for training and testing, whereas 13k were used to evaluate the network's performance in ANN4FLES compared to the existing approach. All events are central Au+Au collisions at 10 GeV energy and therefore within the specifications of the CBM experiment. For training and testing, however, the amounts of Λ was reduced after extraction. As there are usually more existing Λ than K_s created in the collision, the training and validation was performed on an equal number of particles to ensure fair results.

In KFPF, there are several classes to measure the algorithm's performance and efficiency by comparing the reconstructed results to the simulated generated data. In particular, the `KFTopoPerformance` class includes a method `MatchParticles()` to match reconstructed particles with their corresponding simulated monte carlo particles. Thus, it is possible to verify for each particle that is reconstructed correctly.

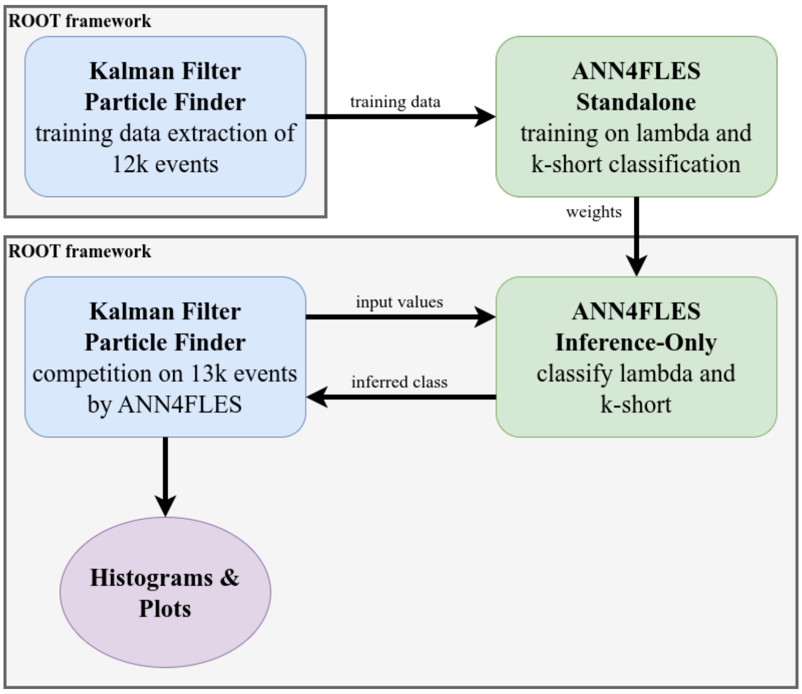

ROOT framework
Kalman Filter Particle Finder
training data extraction of 12k events
training data
ANN4FLES Standalone
training on lambda and k-short classification
weights
ROOT framework
Kalman Filter Particle Finder
competition on 13k events by ANN4FLES
input values
inferred class
ANN4FLES Inference-Only
classify lambda and k-short
Histograms & Plots

Figure 4.1: Visualization of the procedure: Starting with the extraction of data within the KFPF package, followed by the training in ANN4FLES standalone. Pre-trained network (here represented by weights) is loaded into the ANN4FLES package recently included in the ROOT framework. On a different set of data, the classification performance is then evaluated within KFPF.

To extract crucial information about the true outcome, this method is modified to extract the data, after particles are matched.

Including background particles in the training process would train the neural network directly on noise. Therefore, for training, only signal particles are considered. After the particles are matched, each particle is identified by its PDG code (K_s=310, Λ=3122). Here, the PDG code is truly known and not only a hypothesis, since all matched particles provide monte-carlo information. Then, for each particle, the daughter particles are considered. Since both, $K_s \rightarrow \pi^+\pi^-$ and $\Lambda \rightarrow$ pπ^-, raise pi^- in their decays, a shared pi^--daughter is searched for. One of the parents of pi^-, which is in this case either K_s or Λ, is not the correct parent and should be removed by a competition to reduce background. Therefore, particles that match these criteria are extracted.

The extracted information contain the true parent as a label for the neural network based classifier, and, depending on the approach, the following information. In case of classification on mass values only, for each K_s and Λ, the reconstructed mass values plus the particle's PDG mass is extracted. Following the suggestion of previous work [23], masses and transverse momenta are extracted for the other approach. To provide comparable results, all information are extracted from the same events.

4.2 ANN4FLES Implementation in KFPF

The ANN4FLES package is used and included into the KFPF package to provide a classification in inference-only mode, after the network was fully trained.

The network was included in place of the default competition and the previous neural network approach, within the `KFTopoReconstructor` class (see Listing 4.1). ANN4FLES is initialized, and the pre-trained weights are loaded. Similar to the default competition, a storage for used particles, candidates flagged for deletion and IDs of the best mothers is in use. Analog to previous work, only K_s and Λ are selected for the competition. Then, a χ^2 cut is applied with respect to the primary vertex. Therefore, only primary particles are selected for the competition, whereas others are flagged for deletion.

```
void KFParticleTopoReconstructor::SelectParticleCandidatesByNetwork() {
↪
  Int_t particleNum = fParticles.size();

  std::vector<bool> isUsed(fParticles.size());
  std::vector<bool> deleteCandidate(fParticles.size());
  std::vector<int>  bestMother(fParticles.size());

  Network net = Network();
  net.InitializeANN(topology, true);
  net.ImportNeuronWeights(weight_path);

  int pCounter = 0;

  for(unsigned int iParticle=0; iParticle<fParticles.size(); iParticle++)
  {
    isUsed[iParticle] = false;
    deleteCandidate[iParticle] = false;
    bestMother[iParticle] = -1;
  }

  for(unsigned int iParticle=0; iParticle<fParticles.size(); iParticle++) {
↪
    KFParticle tmp = fParticles[iParticle];
    if(!UseParticleInCompetitionForNetwork(fParticles[iParticle].GetPDG()))
↪   {
      continue;
    }
    tmp.SetProductionVertex(GetPrimVertex());
    if(tmp.Chi2()/tmp.NDF()>3)
    deleteCandidate[iParticle] = true;
  }

```

Listing 4.1: Initialization of ANN4FLES within the `KFTopoReconstructor` class in KFPF.

To perform a rough cut similar to the existing method in KFPF, the mass differences with respect to the known peak are calculated for K_s and Λ (see Listing 4.2, lines 10, 21, 27). Again, the following code snippet examines if the masses of at least one of the candidates is within the 3σ range of the known PDG mass for the respective particles, as it is statistically unlikely to find K_s and Λ with a distance to their peak larger than 3σ, as the particles follow a normal distribution.

```
1   for(unsigned int iParticle=0; iParticle<fParticles.size(); iParticle++) {
2     if(deleteCandidate[iParticle]) continue;
3
4     if(!UseParticleInCompetitionForNetwork(fParticles[iParticle].GetPDG()))
    ↪ continue;
5
6     float mass1, massSigma1;
7     fParticles[iParticle].GetMass(mass1, massSigma1);
8     float massPDG1, massPDGSigma1;
9     KFParticleDatabase::Instance()->GetMotherMass(fParticles[iParticle].
    ↪ GetPDG(), massPDG1, massPDGSigma1);
10    float dm1 = fabs(mass1 - massPDG1)/massPDGSigma1;
11
12    for(unsigned int jParticle=iParticle+1; jParticle<fParticles.size();
    ↪ jParticle++) {
13      if(deleteCandidate[jParticle]) continue;
14
15      if(!UseParticleInCompetitionForNetwork(fParticles[jParticle].GetPDG()))
    ↪  continue;
16
17      float mass2, massSigma2;
18      float massPDG2, massPDGSigma2;
19      fParticles[jParticle].GetMass(mass2, massSigma2);
20      KFParticleDatabase::Instance()->GetMotherMass(fParticles[jParticle].
    ↪ GetPDG(), massPDG2, massPDGSigma2);
21      float dm2 = fabs(mass2 - massPDG2)/massPDGSigma2;
22
23      if(! (fParticles[iParticle].DaughterIds()[0] == fParticles[jParticle].
    ↪ DaughterIds()[0] &&
24      fParticles[iParticle].DaughterIds()[1] == fParticles[jParticle].
    ↪ DaughterIds()[1]) ) continue;
25      if (fParticles[iParticle].GetPDG() == fParticles[jParticle].GetPDG())
    ↪ continue;
26
27      if(dm1 < 3.f || dm2 < 3.f){
28        ... //PART 2
29
```

Listing 4.2: Part 1 of the main loop. Calculation of mass differences with respect to the peak.

However, if one of the particles is likely to be either K_s or Λ, the neural network based competition starts. First, the input of the neural network is stored into a vector that is then fed into ANN4FLES for classification (Listing 4.3, line 2-3). Here, `InferOutput(input)` returns an integer that corresponds to the predicted class. If the value of the first neuron is larger, 0 is returned and, hence, Λ is classified (Listing 4.3, line 5-10). Contrary, if the second neuron's value is larger, 1 is returned to classify K_s (Listing 4.3, lines 11-16). After the classification is performed, a clean-up for particles that were forced to skip the competition is done by checking if both daughters have a valid mother particle, and, if not, they are flagged for deletion. Finally, the PDG code of all particles that were flagged for deletion is set to an invalid code of -1 to indicate they are rejected (Listing 4.3, line 30-32).

```
1          ... // proceeding
2          std::vector<float> inputs_ann = {mass1, massPDG1, mass2, massPDG2};
3          int out = net.InferOutput(inputs_ann);
4
5          // lambda
6          if (out == 0) {
7            deleteCandidate[iParticle] = true;
8            bestMother[fParticles[iParticle].DaughterIds()[0]] = jParticle;
9            bestMother[fParticles[iParticle].DaughterIds()[1]] = jParticle;
10         }
11         // kshort
12         else if (out == 1) {
13             deleteCandidate[jParticle] = true;
14             bestMother[fParticles[iParticle].DaughterIds()[0]] = iParticle;
15             bestMother[fParticles[iParticle].DaughterIds()[1]] = iParticle;
16         }
17       }
18     }
19   }
20   for(int iParticle=0; iParticle<int(fParticles.size()); iParticle++) {
21     if(deleteCandidate[iParticle]) continue;
22
23     for(int iDaughter=0; iDaughter<fParticles[iParticle].NDaughters();
   ↪ iDaughter++)
24     if(bestMother[fParticles[iParticle].DaughterIds()[iDaughter]] !=
   ↪ iParticle && bestMother[fParticles[iParticle].DaughterIds()[iDaughter]]
   ↪  > -1) {
25       deleteCandidate[iParticle] = true;
26       break;
27     }
28   }
29
30   for(unsigned int iParticle=0; iParticle<fParticles.size(); iParticle++)
31     if(deleteCandidate[iParticle]) fParticles[iParticle].SetPDG(-1);
32   }
```

Listing 4.3: Part 2 of the main loop, including the neural network based competition.

4.3 Deep Learning Classification: m + PDGm

As previous work already showed that a neural network can perform comparable to the default approach of KFPF, ANN4FLES is used to implement a slightly more complex neural network for this classification task (see Figure 4.2).

After using the same architecture as it was used in previous work, a neural network with a more complex architecture was finally chosen, as it provides accuracies of up to 98.6% for the validation set (see Figure 4.3). It was shown on multiple well-known data sets, that the results of other neural network packages offered comparable results to ANN4FLES, which makes one confident that the implemented maths in ANN4FLES are correct [19, p. 161]. However, the opportunity was taken to

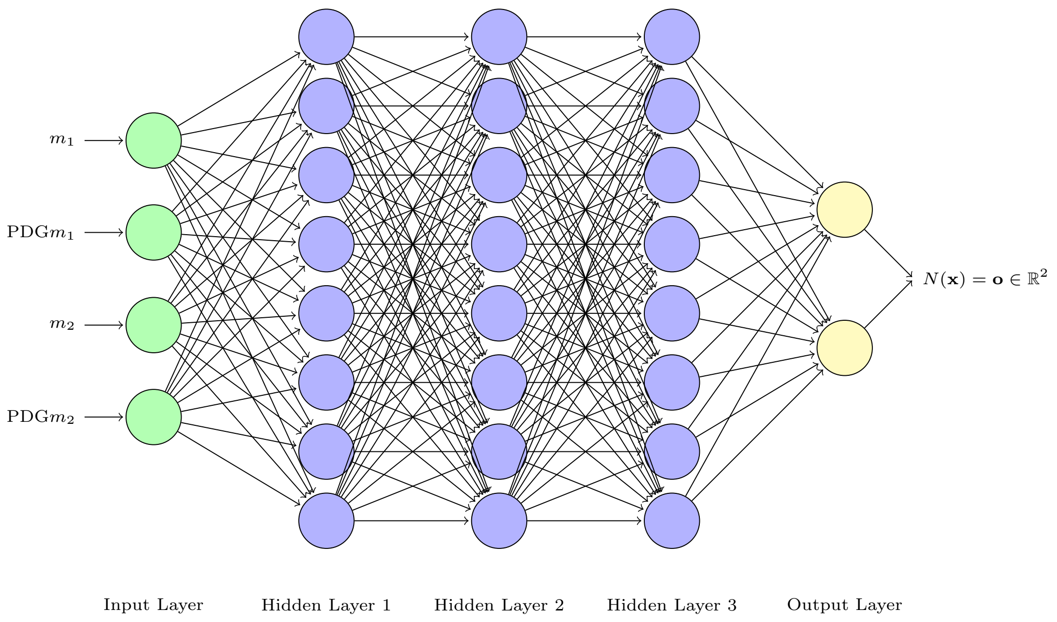


Figure 4.2: Deep learning architecture used to classify K_s and Λ, using 3 hidden layers with 8 neurons each, hidden activation function LReLU. Output layer consists of two neurons with softmax activation and cross entropy loss. Here, based on m and PDGm for each candidate.

compare ANN4FLES with PyTorch on the extracted data set too. An identically constructed neural network was implemented in PyTorch to ensure valid results. The PyTorch architecture provided equally high accuracy values and again confirmed the classification performance of ANN4FLES.

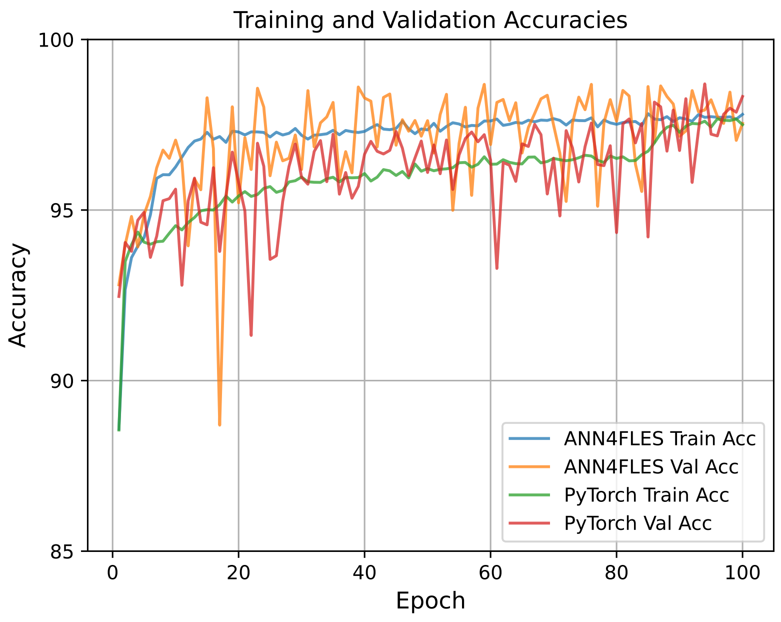


Figure 4.3: ANN4FLES and PyTorch accuracies for training and validation, using mass and PDG mass over 100 epochs with a peak performance of 98.6% in the validation set. The classification is based on m and PDGm of each candidate. ANN4FLES seems to learn faster than PyTorch here. However, both achieve a high accuracy on validation set.

An error of only 1.4% on the validation set suggests that the classification performance of the more complex ANN4FLES architecture is better, since the error rate of previous work is given with more than 10%. Although the generators for simulated events should perform equally, further investigations are suggested, as previous work used the PHSD model to generate data for the neural network training and validation, whereas in the present thesis the UrQMD model was chosen. The ANN4FLES architecture uses ADAM optimizer based on mini-batch gradient descent with a batch size of 50, whereas previous work used the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm to train the neural network [23, 80, 81]. The initialization in previous work is unknown, but the ANN4FLES network used the same initialization as PyTorch, according to their website [86].

To find well performing architecture and settings, different learning rates in the range of 0.001 and 0.005 were tested. For ADAM, the default β values were chosen according to their paper [87]. The final results are accomplished by a network that was trained with a learning rate of 0.003, whereas lower learning rates performed almost equally, larger learning rates tend to perform slightly worse. It is assumed that a larger learning rate does not allow to move as deep into a minimum as with using lower rates, since it overshoots the minima slightly.

Beside the rates, different layer sizes and network depths were tested. Here, the main focus was set on to find a balance between network size and results. On the one hand, a more complex structure could lead to an even better performance. However, the raw classification performance is already good, such that much deeper networks were not tested in this thesis. A deep neural network increases the amount of parameters and calculations, slowing the process, which could be negative with respect to the fast algorithms required for FLES. On the other hand, less complex structures seem to perform worse. Nevertheless, it is suggested to extend ANN4FLES with an automatized architecture finder, that changes hyper-parameters and tests the network's performance fully automatic for a given range of parameters, such as layer sizes, depth and feed forward time. That can help to find a matching architecture for specific requirements.

However, in Figure 4.4, the total spectra histograms for K_s and Λ are shown. The results using KFPF without competition and with the existing method are visualized for comparison. The ANN4FLES approach is colored red, and one can see that it seems to reduce the number of entries over the whole range. In areas with larger distance to the peak, this is likely to be background. Nevertheless, there is also a reduction of entries at the peak area for K_s. To evaluate the real performance, other histograms has to be taken into account.

Investigating the signal histograms in Figure 4.5, in both cases, a slight reduction of signal is indicated compared to running KFPF without competition. For K_s, the ANN4FLES approach rejected slightly more signal particles compared to the existing method, whereas for Λ the results are the other way around: Here, the existing method rejects more signal particles than ANN4FLES. In comparison to Figure 4.4, one can now state that the reduction in the peak area is mostly due to correctly rejected background particles. Altogether, both competitions reduce the signal only by an negligible amount.

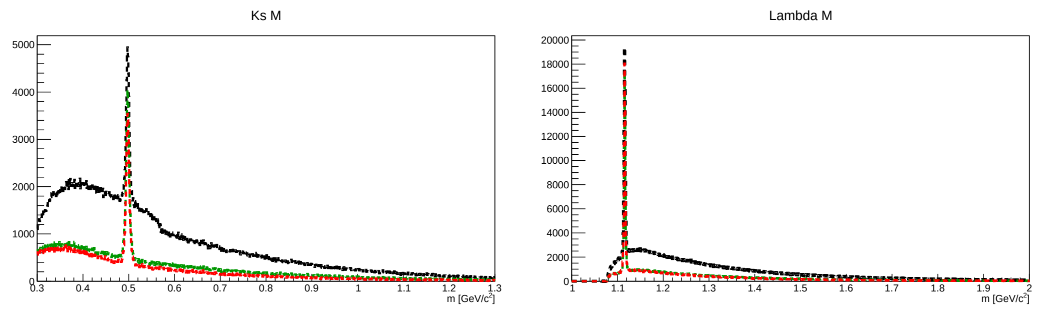


Figure 4.4: Histograms of total spectra for K_s (left) and Λ (right) masses. Comparison of existing mother particle competition (green), no competition (black) and competition by ANN4FLES (red).

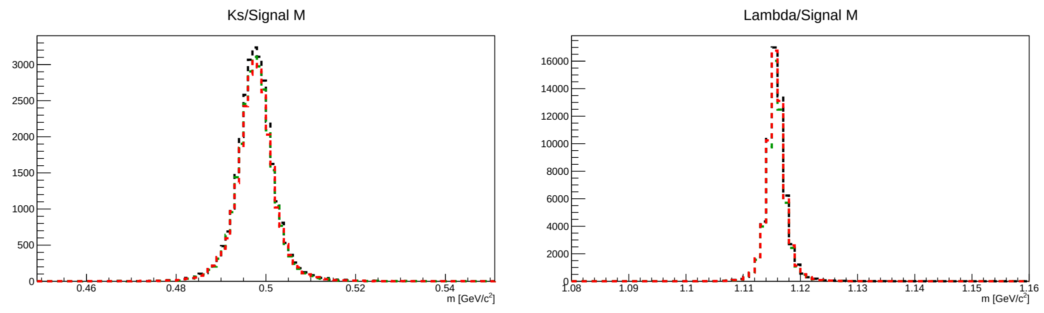


Figure 4.5: Histograms of signals for K_s (left) and Λ (right) masses. Comparison of existing mother particle competition (green), no competition (black) and ANN4FLES (red).

In Figure 4.6, the background is shown. Based on the histogram for Λ, it is difficult to see which competition is better. Both reduce the background significantly, but around the peak the existing method seems to perform better, as we can see a peak for ANN4FLES at Λ's PDG mass $1.116\text{GeV}/c^2$, whereas in the whole range, ANN4FLES seems to reduce the background slightly more. For Λ, this histogram indicates a tie between ANN4FLES and the existing method. However, evaluating K_s, there is no peak at the PDG mass of $0.493\text{GeV}/c^2$ by ANN4FLES. The neural network seems to perform well in rejecting K_s background around the known mass. Here, ANN4FLES shines over the whole range of mass bins.

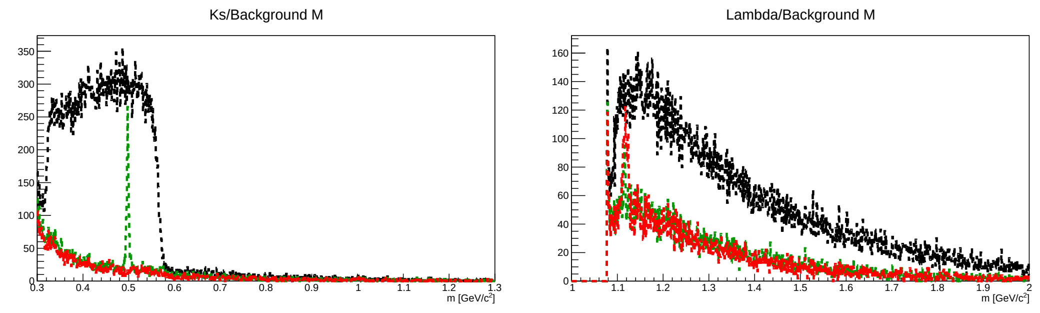


Figure 4.6: Histograms of background for K_s (left) and Λ (right) masses. Comparison of existing mother particle competition (green), no competition (black) and ANN4FLES (red).

In the ghost histograms (see Figure 4.7), ANN4FLES seems to be ahead in both cases Λ and K_s. Although, similar to the existing method, ANN4FLES has a peak around the PDG mass bin, the existing method has more ghosts over the whole range and within the peak area. In case of K_s, the network again shines, having a slight distance to the existing approach. Thus, the neural network rejects more ghost particles than the default approach and therefore helps to reduce the background.

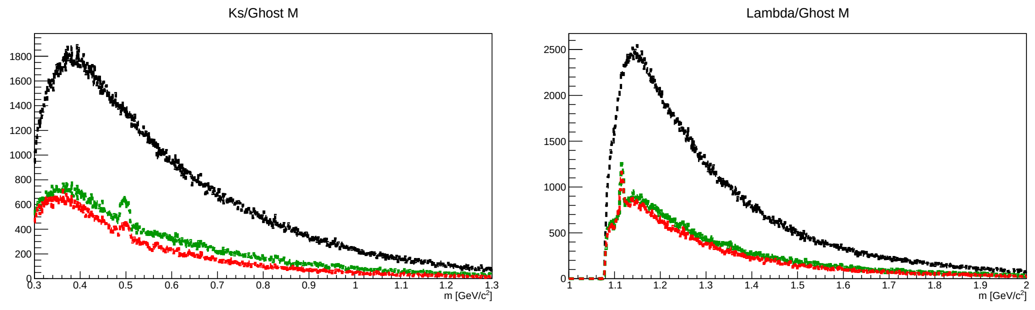


Figure 4.7: Histograms of ghosts for K_s (left) and Λ (right) masses. Comparison of existing mother particle competition (green) and no competition (black) and ANN4FLES (red).

The Armenteros-Podolanski plots in Figure 4.8 show a comparison to the existing method against the ANN4FLES approach. It was already shown that, without competition, the amount of background is huge enough to blur the arch K_s almost completely. In a comparison between both competitions, however, the arch is clearly visible in both approaches. The most noticeable area is at $\alpha = 0.75$ and transverse momentum 0.1 GeV/c. Using the existing competition, the overlapping zone of K_s and Λ has more entries, whereas ANN4FLES seems to not count them into the K_s distribution. Altogether, both competitions have difficulties in this area, as it is hard to distinguish between both particles if their properties almost match for both cases. Considering the symmetry that should exist on average for K_s, the overlapping area has too few entries with ANN4FLES but too many entries in the existing method.

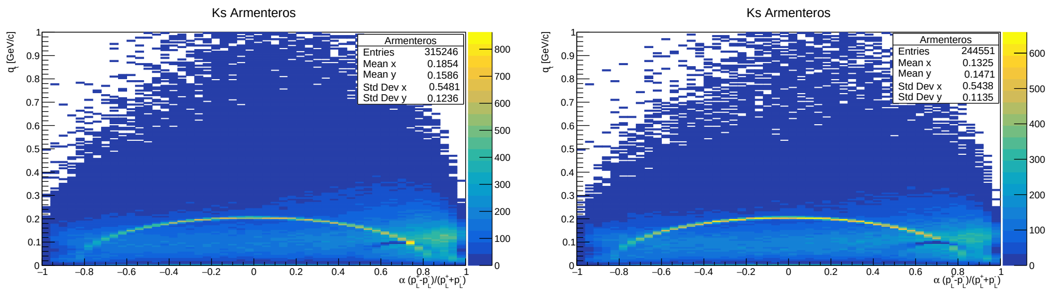


Figure 4.8: Armenteros-Podolanski plots with total spectra for K_s with existing competition (left) and with ANN4FLES based competition (right).

For Λ's Armenteros-Podolanski plot (see Figure 4.9), both competitions seem to perform comparable again. Whereas the existing method has slightly fewer entries within the yellow colored areas, some background bins where K_s's arch is expected are removed completely. The ANN4FLES approach in turn has more entries in the arch of Λ, but the area where the arch of K_s is expected, it has up to 200 entries per bin. Due to rounding behavior per bin, based on this plot only, it is difficult to argue which competition is better.

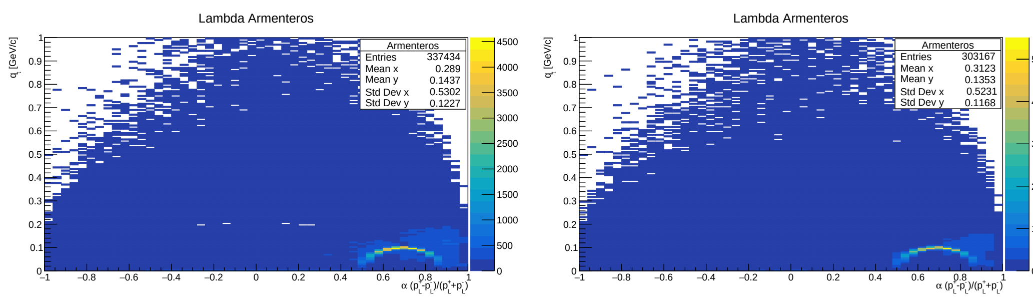


Figure 4.9: Armenteros-Podolanski plots for Λ with the existing competition (left) and with ANN4FLES based competition (right).

For this comparison, some histograms have shown that ANN4FLES seems to perform better in background and ghost rejection, especially for K_s. Analog to the comparison of the existing approach to KFPF without competition, the signal-background ratio and significance are compared now. Figure 4.10 shows the invariant mass spectra of $K_s = \pi^+\pi^-$ and $\Lambda = p\pi^-$ for the existing competition again. It was already shown that the existing competition improved S/B-ratio and significance values, hence, KFPF without competition is not illustrated again.

As already mentioned, the goal is to have the S/B-ratio and significance as large as possible, where for the significance a value larger than 5 is considered as a threshold to distinguish between signal and background with certainty. With a significance of 149 and 213 for K_s and Λ respectively, the clear signal is given in both cases using the default competition.

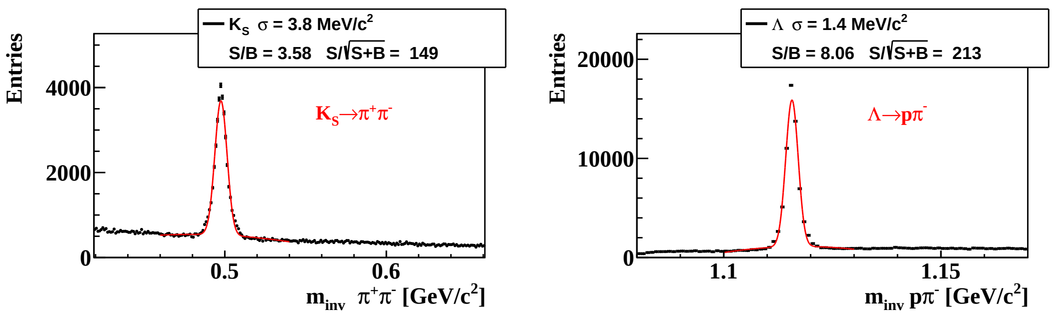


Figure 4.10: Invariant mass distributions of $K_s = \pi^+\pi^-$ and $\Lambda = p\pi^-$ for the default competition, including signal-background ratio and significance.

The following results are achieved by the ANN4FLES approach (see Figure 4.11). Comparing the S/B-ratio for K_s, one can see that the ANN approach improved the value by around 16%. For Λ, however, the S/B-ratio has been reduced by around 2%. Considering both particles, ANN4FLES has reduced the background successfully even further. However, for K_s the significance was also decreased by around 3%, whereas the significance of Λ was increased by 1%. That indicates, that even if the background was reduced successfully on average, the fluctuation in the background has been increased on average in comparison to the existing method. Nevertheless, both significances are high enough to consider these particles as a clear signal.

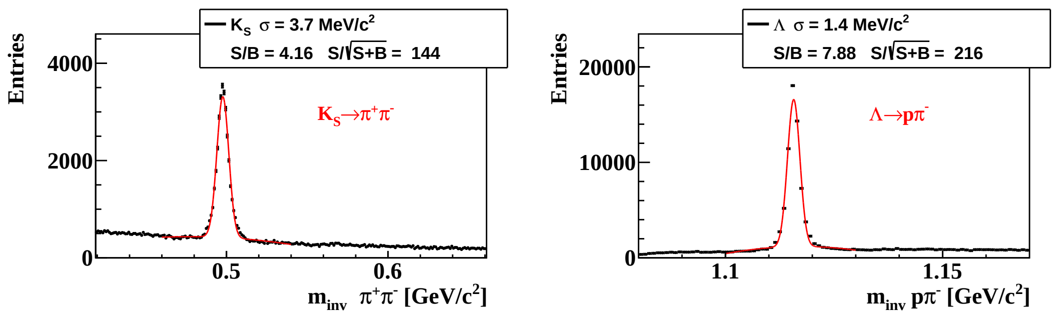


Figure 4.11: Invariant mass distributions of $K_s = \pi^+\pi^-$ and $\Lambda = \mathrm{p}\pi^-$, including signal-background ratio and significance for the ANN4FLES approach.

Summarized, the ANN4FLES based competition using mass and PDG mass can perform comparable to the existing method. In general, it reduces background better than the existing approach, even though the significance is decreased slightly.

4.4 Deep Learning Classification: $m + p_t$

Since previous work suggested an approach using mass m and transversal momentum p_t of both candidates to classify between K_s and Λ, that approach is introduced in this section. The same ANN4FLES architecture (see Figure 4.12) was trained on m and p_t. Now, new results are compared to the ANN4FLES approach using m and PDGm.

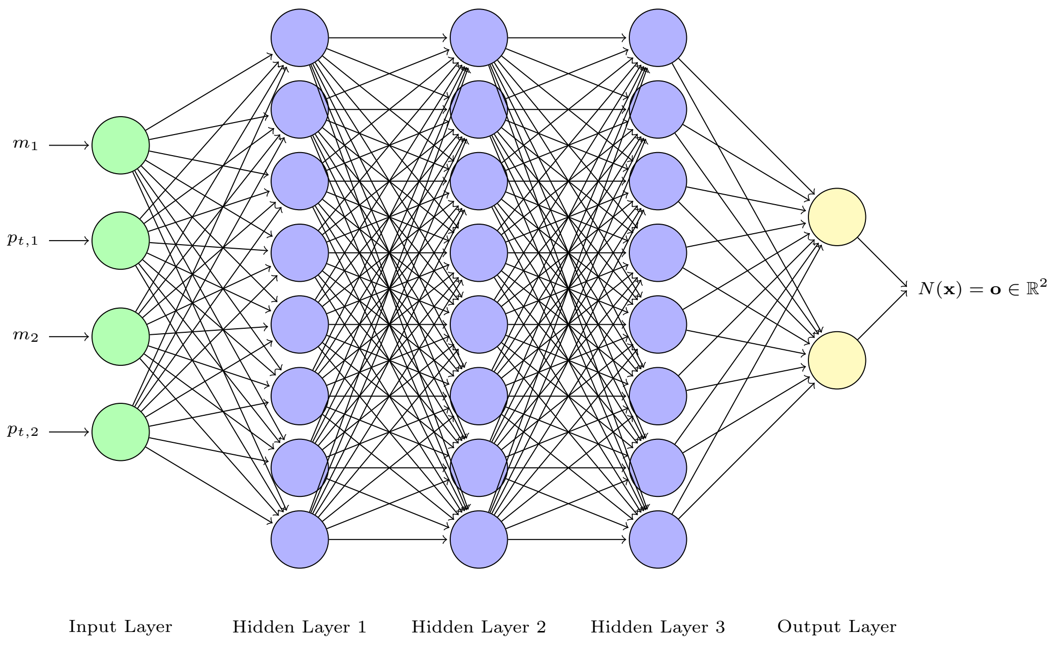


Figure 4.12: Deep learning architecture used to classify K_s and Λ, using 3 hidden layers with 8 neurons each, hidden activation function LReLU. Output layer consists of two neurons with softmax activation and cross entropy loss. Here, based on m and p_t of each candidate.

Again, a comparison network is implemented using PyTorch. This time, the PyTorch network seems to perform slightly better using K_s's and Λ's mass and transverse momentum (see Figure 4.13). However, again, both networks achieve accuracy values on the test set with more than 98%. The raw classification performance did not change, using the transverse momenta instead of PDG masses. However, one will see that there are slight differences in the histograms.

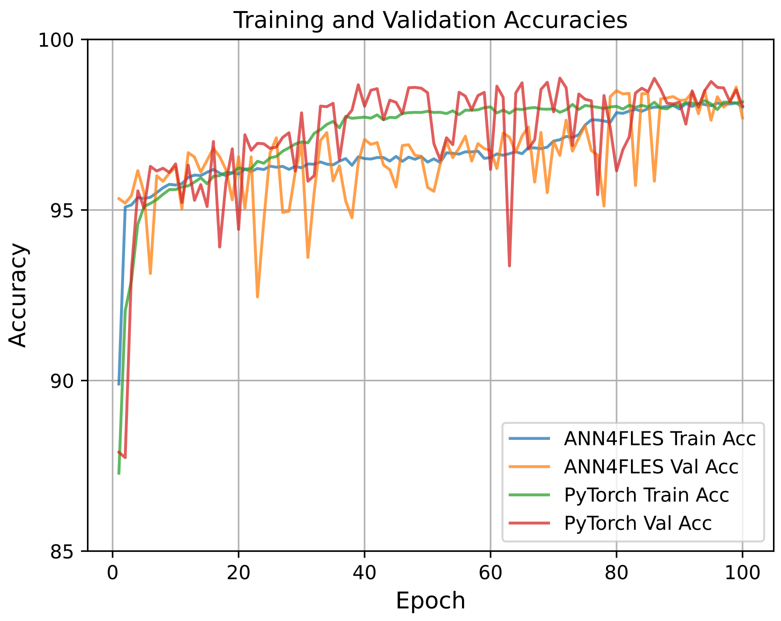


Figure 4.13: ANN4FLES and PyTorch accuracies for training and validation, using mass and transverse momentum over 100 epochs with a peak performance of more than 98% in the validation set. Here, classification based on m and p_t of each candidate. PyTorch seems to learn slightly faster than ANN4FLES on m and p_t. However, both achieve high accuracy on the validation set again.

First, again, the total mass spectra are evaluated (Figure 4.14), as they provide a rough overview over the results. For K_s one can see that the new approach using p_t is slightly ahead in the peak area and besides that performs equally compared to the use of PDGm. For Λ, contrary, it seems vise versa. In the peak area the approach with PDGm is ahead, whereas the performance around the peak is equally again. The reason might be, that the network is now focused more on Λ instead of K_s. As it is likely to have multiple local minima in the loss function, one might have a better performance on classifying Λ whereas another minima might perform better on K_s, without changing the average performance. To evaluate it further, it is in general suggested to review the performance on K_s and Λ independently, to find differences and possible reasons.

Figure 4.15 indicates the same. For Λ, the previous approach on PDGm performed better, whereas for K_s performed better on transverse momentum p_t. However, it is also clearly visible that both ideas perform equally, since the results relative differences are minimal.

In the background histograms (see Figure 4.16), however, it is shown that around

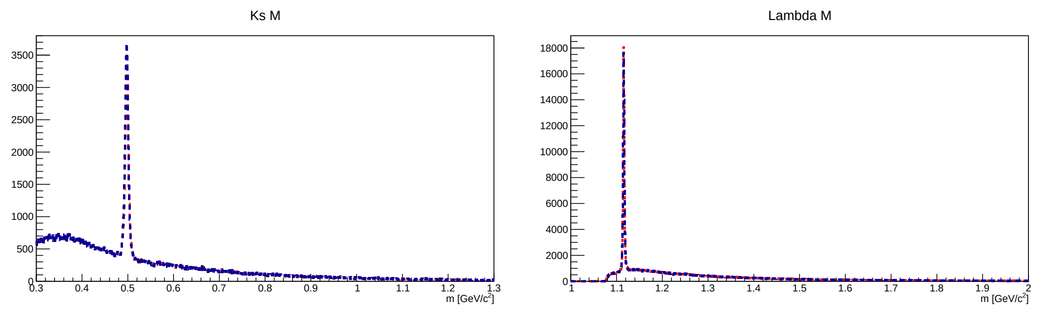


Figure 4.14: Histograms of total mass spectra for K_s and Λ. Comparison of ANN4FLES based on m and PDGm (red) against m and p_t (blue).

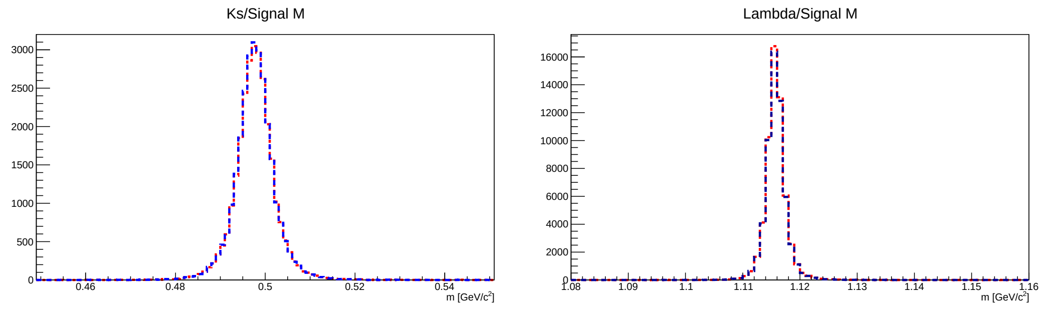


Figure 4.15: Histograms of signal mass spectra for K_s and Λ. Comparison of ANN4FLES based on m and PDGm (red) against m and p_t (blue).

the peak of K_s (PDG mass $0.493\text{GeV}/c^2$), the new approach performed worse. In Figure 4.14, the entries in K_s that were more compared to the approach using PDGm are mostly based on non-rejected background, as one can see in Figure 4.16. For Λ, the background was decreased in the peak area, indicating why there were fewer entries around the peak in Figure 4.14.

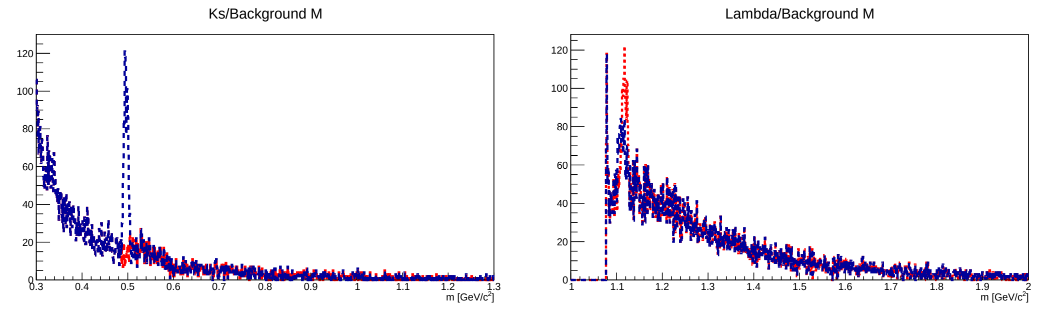


Figure 4.16: Histograms of background mass spectra for K_s and Λ. Comparison of ANN4FLES based on m and PDGm (red) against m and p_t (blue).

The ghost histogram (Figure 4.17) indicates, that the background increase in K_s (here: Figure 4.16) is likely to be based on other particles instead of ghosts. The ghost histograms are comparable, whereas the PDGm approach has slightly fewer ghosts in the peak area of K_s, but contrary more ghosts within the peak area of Λ.

The Armenteros-Podolanski plots for K_s confirm the already identified problems of the new approach. Whereas on the left plot of Figure 4.18, the PDGm approach, the

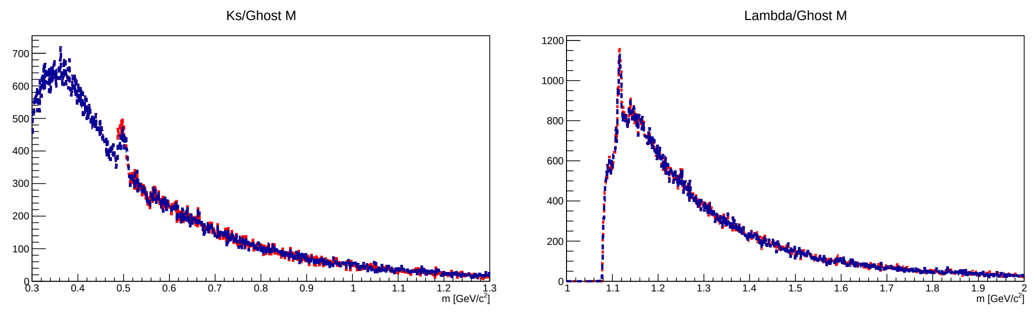


Figure 4.17: Histograms of ghost mass spectra for K_s and Λ. Comparison of ANN4FLES based on m and PDGm (red) against m and p_t (blue).

background produced by Λ was rejected quite good, the p_t approach contrary offers the same pattern as the already existing method of KFPF without neural networks. Here, the Λ background is more visible after the competition with p_t compared to PDGm. That also leads to a high amount of entries in the overlapping zone of K_s and Λ in the Armenteros plots, whereas in the PDGm approach, the arch of K_s is the brightest pattern.

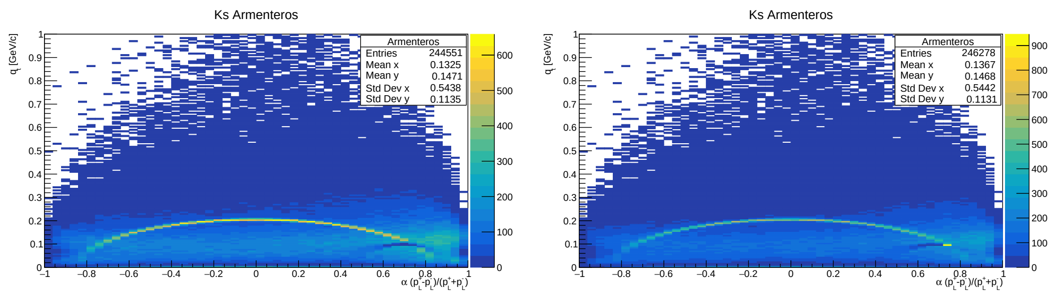


Figure 4.18: Armenteros-Podolanski plots of total spectra for K_s with existing ANN4FLES competition using PDGm (left) and p_t (right).

For Λ's Armenteros-Podolanski plots, however, it is difficult to mention differences, as both plots look very similar in regard to visible patterns. The p_t approach has slightly more entries indicated by the same color compared to the PDGm approach. Nevertheless, in both plots the pattern of K_s is gone, since the background produced by K_s is as large as the produced background by other particles.

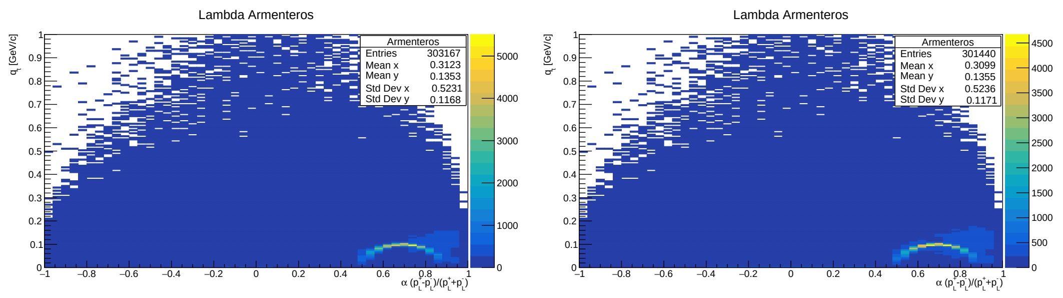


Figure 4.19: Armenteros-Podolanski plots of total spectra for Λ with existing ANN4FLES competition using PDGm (left) and p_t (right).

Finally, the invariant mass histograms are shown (Figure 4.20), including the S/B-ratio and significance values of the new results. For the ANN4FLES approach with mass and PDG mass, the values for S/B-ratio for K_s and Λ were 4.16 and 7.88 respectively - for significance, 144 and 216. The existing method had a S/B-ratio of 3.58 (K_s) and 8.06 (Λ), and significance 149 and 213 respectively. Hence, compared to the other ANN4FLES approach, the S/B-ratio was increased by the network based on transverse momentum and mass. But in comparison to the existing method, it performed better on K_s only. The significance of Λ dropped from 216 to 213, whereas the significance of K_s increased from 144 to 149. Thus, the fluctuation in the background of Λ in relation to the signal has become larger using p_t. On the other hand, the background fluctuation of K_s was reduced slightly.

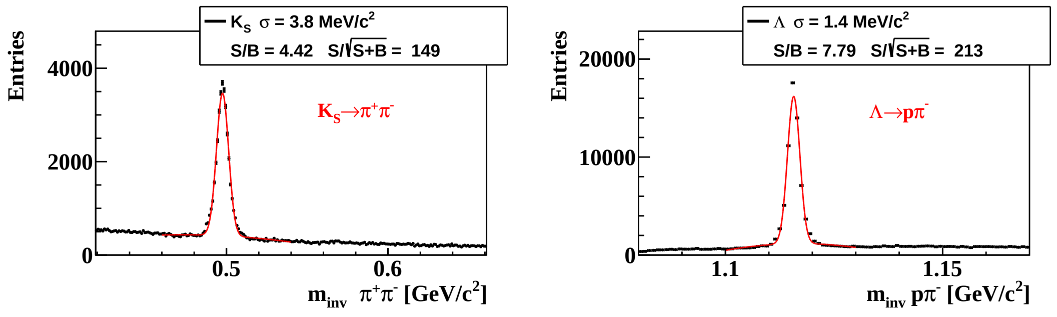


Figure 4.20: Invariant mass distributions of $K_s = \pi^+\pi^-$ and $\Lambda = \mathrm{p}\pi^-$, including signal-background ratio and significance for the ANN4FLES approach with mass and transverse momentum.

Summarized, using either PDG mass or transverse momentum does not offer huge differences in the final results. Both approaches provide comparable and good results and, at least for K_s, they always achieve higher S/B-ratios. They both accomplished their task in improving the S/B-ratio, but so did the existing method of KFPF.

Chapter 5

Conclusion

Scientists around the globe use heavy-ion physics experiments to gain insights in the structure of our universe. The future CBM experiment at FAIR will complement the already existing heavy-ion research programs of other facilities by its unique capabilities to investigate rare short-lived particles with a high interaction rate of up to 10 MHz and research of matter under extreme baryonic densities.

It was shown that the reconstruction of short-lived particles requires a full event reconstruction, as, due to their short lifetime, they will not appear in any detector responses directly and can, hence, only be measured indirectly. Combined with the high interaction rate, fast algorithm packages for online reconstruction are required. The data streams produced by detector responses are too large to be stored entirely and since short-lived particles are of scientists' interest, the whole event selection has to be performed in real-time, including the reconstruction of decay chains. The Kalman Filter Particle Finder is such a library that is able to reconstruct more than 150 decays.

In recent years, machine learning algorithms and neural networks in particular have become popular in almost all research disciplines. Neural networks provide a powerful tool whose capabilities have not been fully explored yet. In heavy-ion physics experiments, these tools can be used for several tasks, such as within the reconstruction of events, to classify physics data or to solve algorithmic problems that are otherwise computational expensive.

Therefore, the application of neural networks within the packages required for the CBM experiment is investigated. On the one hand, they can improve the physicists' analysis. Neural networks and machine learning algorithms can be used to classify particles, dynamically apply cuts to data, or to find patterns that are difficult to find otherwise. Furthermore, one can use generated simulated data to train a neural network by supervised learning. Here, the true outcomes are known and the improvements in high quality physics models increase the powerful capabilities even more. This allows to produce an almost arbitrary amount of training data, which is not possible in many cases where neural networks are in use.

On the other hand, neural networks are complex systems that require precise analyses. Although the general principles and mathematics of neural networks are known, it is almost impossible to make precise predictions about complex and deep architectures. That behavior is what gives neural networks the image of a black-box. However, the ongoing researches in the machine learning areas provide ever more certainty and the increase in machine learning algorithm applications indicate their potential.

The work group developed a fast C++ package called ANN4FLES, providing several neural network architectures for the usage in the CBM experiment. In the present thesis, the standalone version of ANN4FLES is used to train neural networks for the specific tasks. Then, pre-trained networks are included into the FLES package for the first time. A neural network based approach for the identification of short-lived particles in a mother particle competition of the KFPF package was studied. Previous work has already shown that neural networks can solve the task, with results comparable to the existing method.

The studied particles are $K_s \rightarrow \pi^+\pi^-$ and $\Lambda \rightarrow \mathrm{p}\pi^-$. Both, K_s-meson and Λ-hyperon are neutral particles that consist of strange quarks. Since theoretical predictions assume that increased strangeness production might be an indicator for deconfined matter and both particles are abundantly present in the events at energy ranges of the CBM experiment, they are considered as reliable carriers of information about the collision's properties.

The existing method is based on the distances of the particle candidate's mass values to the respective known mass distribution peak. The first neural network based approach follows the idea of using mass and PDG mass values as an input for the neural network, to classify the best matching mother particle. The present thesis has shown that the neural network based approach using particle candidate's mass and PDG mass can be improved by a more complex neural network architectures.

However, previous work also suggested to adjust the approach by using different input data for the neural network. In this thesis, it was also shown that the approach using particle's mass and transverse momentum works comparable to the presented approach using mass and PDG mass values.

Recapitulating the results using ANN4FLES, both competitions worked as expected and compared to KFPF without competition, both seem to increase the quality of the reconstruction slightly. For instance, it was shown that the signal-background-ratio was increased on average by both approaches. This indicates that both neural networks rejected background particles more than signal particles. The differences between the ANN4FLES approach with mass and PDG mass in comparison to mass and transverse momentum were negligibly small.

Nevertheless, the existing method in KFPF also produced almost equal results. For the identification of K_s and Λ in particular, the neural network approaches can help to increase the quality of data, but for the general usage of the CBM experiment, the neural network based approach is not considered as required. Furthermore, other particles' quality is not studied in this thesis. The neural networks will likely have an effect on other particles as well, and a classification on K_s and Λ only will most likely have negative effects to other particles if they do not take part in another competition.

Generally, for future work it has to be considered to investigate how neural networks perform in classifying either all candidates that are selected for the competition or by increasing their possibilities with thresholds. Even though the former approach requires a more complex architecture, the currently studied approaches classify only binary between K_s and Λ. This, in fact, limits the networks' performances as they are not allowed to identify the input as something else than the respective particles and forbids to reject an input completely, classifying it as background. If a particle is considered as a possible candidate, in binary classification, the prediction is done regardless of uncertainties. Here, thresholds can be used, such that only particles that are classified with certainty survive the competition and others are rejected. This might increase the physics quality even more.

Furthermore, it has to be studied how much impact the competition has in regard to the final results. The raw classification performance of both networks with an accuracy of more than 98% suggests that the classification in K_s and Λ is not the limiting factor. Either, the already mentioned multi-classification can improve the final results even more, or it has to be considered if the competition in general is limited with its impact to final results.

Summarized, all presented particle competitions provide a good performance in rejecting more background particles than signal particles for the studied decays $K_s \rightarrow \pi^+\pi^-$ and $\Lambda \rightarrow \mathrm{p}\pi^-$. Hence, all competitions increase the physics analysis quality for K_s-mesons and Λ-hyperons. The usage of deep learning techniques to solve the task efficiently leads to promising results and should be studied in more detail.

Chapter 6

Zusammenfassung

Forschende auf der ganzen Welt nutzen Schwerionen-Experimente, um Erkenntnisse über die Struktur des Universums zu gewinnen. Das zukünftige CBM-Experiment an der FAIR-Teilchenbeschleunigeranlage wird bestehende Forschungsprogramme anderer Einrichtungen dahingehend ergänzen, dass CBM einzigartige Möglichkeiten bieten wird, kurzlebige Teilchen zu erforschen. Ein weiterer Schwerpunkt des Experiments ist die Erforschung von Phasenübergängen der Materie unter extrem hohen baryonischen Dichten.

Das Aufspüren seltener Teilchen erfordert eine häufige Wiederholung des Experiments, weshalb das CBM-Experiment Teilchenkollisionen mit einer Interaktionsrate von bis zu 10 MHz durchführen wird. Da bei diesen Kollisionsraten nicht alle Kollisionen vollständig abgespeichert werden können, birgt diese Herangehensweise zusätzliche Herausforderungen. Seltene kurzlebige Teilchen werden nicht im Detektorsystem erkannt, da ihre Lebensdauer nicht ausreicht, um einen der Detektoren zu erreichen. Dies bedeutet, dass für die Erkennung ebendieser Teilchen eine vollständige Rekonstruktion der Teilchenkollision nötig ist - inklusive der Rekonstruktion von Zerfallsketten. Nach der vollständigen Rekonstruktion kann man anhand der vermuteten Teilchen das entsprechende Ereignis abspeichern.

Für diese Selektion von Ereignissen sind Hochleistungsalgorithmen entwickelt worden, die innerhalb des First Level Event Selection Pakets verschiedenste Aufgaben bewältigen. Die Rekonstruktion von Zerfallsketten wird dabei von dem Kalman Filter Particle Finder (KFPF) Paket durchgeführt.

In der vorliegenden Arbeit wird eine Methode des KFPF Pakets mit vortrainierten neuronalen Netzen erweitert, um die Leistung einer seiner Kernaufgaben zu verbessern: die Identifikation von Partikeln. Partikel, die direkt aus der Kollision heraus entstehen, sind Mutterpartikel. Durch die enormen Energien, die auf die Teilchen wirken, durch Kollisionen mit anderen Teilchen oder durch Kollisionen mit den Detektoren, zerfallen diese Teilchen häufig. Die daraus resultierenden Partikel werden Tochterpartikel genannt. Genau hier knüpft die vorliegende Arbeit an.

Zur Rekonstruktion dieser Zerfallsprozesse werden mögliche Mutterpartikel-Kandidaten erzeugt. Anschließend durchlaufen diese möglichen Kandidaten einen Wettbewerb, in dem jeweils der best-passendste Kandidat ausgewählt wird, währenddessen der andere verworfen wird. Fälschlicherweise selektierte Mutterpartikel produzieren Störsignale, wodurch die physikalische Analyse negativ beeinträchtigt wird. Daher ist ein guter Wettbewerb wichtig, um Störsignale zu reduzieren.

Kurz gesagt: Die bestehende Wettbewerbsmethode vergleicht die Distanz der rekonstruierten Partikelmasse mit der für den Partikeltyp typische Partikelmasse (PDG-Masse). Der Partikel, der näher an seiner Soll-Masse liegt, wird selektiert.

Es werden zwei Deep Learning Ansätze präsentiert, die diese Aufgabe übernehmen sollen. Dabei werden insbesondere die beiden neutralen Teilchen Λ (Hyperon) und K_s (Meson) betrachtet. Der erste Ansatz nutzt, ähnlich zur bestehenden Methode, die rekonstruierten Partikelmassen und Soll-Massen der entsprechenden Kandidaten und klassifiziert damit, ob es sich um ein Λ oder K_s Mutterteilchen handelt.

Es wird gezeigt, dass beide Ansätze eine gute Klassifizierungsleistung bieten. In der Test-Phase des neuronalen Netzes erkennen die neuronalen Netze beider Ansätze die Partikel mit einer Genauigkeit von über 98%. Anschließend werden die vortrainierten Netze innerhalb des KFPF Pakets eingesetzt und mit den von KFPF bereitgestellten Leistungswerkzeugen bewertet. Auch hier schneiden die neuronalen Netze gut ab.

Zwischen beiden Netzen gibt es bezüglich der Erkennungsraten nur vernachlässigbar kleine Unterschiede und auch innerhalb des KFPF Pakets ist die Leistung der beiden Ansätze vergleichbar. Im Vergleich zur bestehenden Methode erwiesen sich die neuronalen Netze leicht besser in Bezug auf die Reduktion von Störsignalen, also, falsche Partikel wurden im Durchschnitt besser herausgefiltert. Allerdings wurden hier auch nur Λ- und K_s-Partikel berücksichtigt, sodass der Einfluss auf andere Partikel nicht bewertet werden kann. Es wird daher empfohlen, die Klassifizierung auf mehrere Partikel auszuweiten, um andere Partikel in die Analyse mit einbeziehen zu können.

List of Figures

References

[1] E. Rutherford, ‘‘The scattering of α and β particles by matter and the structure of the atom,’’ *Philos. Mag.*, vol. 21, p. 669, 1911.

[2] G. Zweig, *An SU(3) model for strong interaction symmetry and its breaking. Version 2*, pp. 22–101. 2 1964.

[3] M. Gell-Mann, ‘‘A Schematic Model of Baryons and Mesons,’’ *Phys. Lett.*, vol. 8, pp. 214–215, 1964.

[4] MissMJ, ‘‘Standard model of elementary particles.’’ `https://en.wikipedia.org/wiki/File:Standard_Model_of_Elementary_Particles.svg`, 2013. Source: PBS NOVA [1], Fermilab, Office of Science, United States Department of Energy, Particle Data Group. Copyright MissMJ, licensed under Creative Commons Attribution 3.0 Unported license.

[5] I. Kisel, *Superphysics and Supercomputers: Introduction to Experimental Physics*. August 2021.

[6] U.S. Department of Energy, ‘‘DOE Explains: Quarks and Gluons.’’ `https://www.energy.gov/science/doe-explainsquarks-and-gluons`. Accessed on April 1, 2023.

[7] E. Fokas, G. Kraft, H. An, and R. Engenhart-Cabillic, ‘‘Ion beam radiobiology and cancer: Time to update ourselves,’’ *Biochimica et Biophysica Acta (BBA) - Reviews on Cancer*, vol. 1796, no. 2, pp. 216–229, 2009.

[8] C. Shen, ‘‘Relativistic Heavy-Ion Collisions.’’ `https://chunshen1987.github.io/`, accessed 2023.

[9] ‘‘CBM - Inside a Neutron Star.’’ `https://www.gsi.de/forschungbeschleuniger/fair/forschung/cbm_im_inneren_eines_neutronensterns`. Accessed on April 1, 2023.

[10] ‘‘STAR - Physics.’’ `https://www.star.bnl.gov/central/physics/`. Accessed on April 1, 2023.

[11] G. Aarts, F. Attanasio, B. Jäger, E. Seiler, D. Sexty, and I.-O. Stamatescu, ‘‘QCD at nonzero chemical potential: recent progress on the lattice,’’ 2014.

[12] K. C. Meehan, ‘‘The fixed-target experiment at STAR,’’ *Journal of Physics: Conference Series*, vol. 742, p. 012022, aug 2016.

[13] ‘‘Powering CERN.’’ `https://home.cern/science/engineering/powering-cern`. Accessed on April 1, 2023.

[14] I. Kisel and for CBM Collaboration, ‘‘Event Topology Reconstruction in the CBM Experiment,’’ *Journal of Physics: Conference Series*, vol. 1070, p. 012015, aug 2018.

[15] I. Kisel, I. Kulakov, and M. Zyzak, ‘‘Standalone First Level Event Selection Package for the CBM Experiment,’’ *IEEE Transactions on Nuclear Science*, vol. 60, no. 5, pp. 3703–3708, 2013.

[16] F. Sergeev, E. Bratkovskaya, I. Kisel, and I. Vassiliev, ‘‘Deep learning for quark-gluon plasma detection in the CBM experiment,’’ *International Journal of Modern Physics A*, vol. 35, p. 2043002, Nov. 2020.

[17] A. Seryakov and D. Uzhva, ‘‘Convolutional Neural Network for Centrality Determination in Fixed Target Experiments,’’ *Physics of Particles and Nuclei*, vol. 51, pp. 331–336, 2020.

[18] Łukasz Kamil Graczykowski, M. Jakubowska, K. R. Deja, and M. Kabus, ‘‘Using Machine Learning for Particle Identification in ALICE,’’ 2022.

[19] P. Senger and V. Friese, ‘‘CBM Progress Report 2022,’’ Tech. Rep. CBM PR 2022, Darmstadt, 2022.

[20] M. Zyzak, *Online selection of short-lived particles on many-core computer architectures in the CBM experiment at FAIR*. PhD thesis, J. W. Goethe University, Frankfurt (Main), 2016.

[21] P. Kisel, *KF Particle Finder Package: Missing Mass Method for Reconstruction of Strange Particles in CBM (FAIR) and STAR (BNL) Experiments*. PhD thesis, Goethe U., Frankfurt (main), 2023.

[22] S. Gorbunov, *On-line reconstruction algorithms for the CBM and ALICE experiments*. doctoralthesis, Universitätsbibliothek Johann Christian Senckenberg, 2013.

[23] A. Banerjee, I. Kisel, and M. Zyzak, ‘‘Artificial neural network for identification of short-lived particles in the CBM experiment,’’ *Int. J. Mod. Phys. A*, vol. 35, no. 33, p. 2043003, 2020.

[24] C. Sturm and H. Stöcker, ‘‘The Facility for Antiproton and Ion Research FAIR,’’ *Physics of Particles and Nuclei Letters*, vol. 8, pp. 865–868, 2011.

[25] ‘‘FAIR - Facility Overview.’’ `https://www.gsi.de/en/researchaccelerators/fair/the_machine`, n.d. Accessed April 2, 2023.

[26] H. Stöcker, T. Stöhlker, and C. Sturm, ‘‘FAIR - Cosmic Matter in the Laboratory,’’ *Journal of Physics: Conference Series*, vol. 623, p. 012026, may 2015.

[27] A. Einstein, *Zur Elektrodynamik bewegter Körper*. Leipzig: Teubner, 1905.

[28] V. Friese and for the CBM Collaboration, ‘‘The high-rate data challenge: computing for the CBM experiment,’’ *Journal of Physics: Conference Series*, vol. 898, p. 112003, oct 2017.

[29] Teklishyn, Maksym, ‘‘The silicon tracking system of the cbm experiment at fair,’’ *EPJ Web Conf.*, vol. 171, p. 21003, 2018.

[30] B. Friman, C. Höhne, J. Knoll, S. Leupold, J. Randrup, R. Rapp, and P. Senger, eds., *The CBM Physics Book: Compressed Baryonic Matter in Laboratory Experiments*. Lecture Notes in Physics, Springer Berlin, Heidelberg, 1 ed., 2011.

[31] J. Seguinot and T. Ypsilantis, ‘‘Photo-ionization and Cherenkov ring imaging,’’ *Nuclear Instruments and Methods*, vol. 142, no. 3, pp. 377–391, 1977.

[32] C. Bergmann, ‘‘Development and Test of a Transition Radiation Detector Prototype for CBM @ FAIR,’’ Master’s thesis, Westfälische Wilhelms Universität Münster, 2009.

[33] N. Herrmann, ed., *Technical Design Report for the CBM Time-of-Flight System (TOF)*. Darmstadt: GSI, 2014.

[34] I. E. Korolko, M. S. Prokudin, and Y. M. Zaitsev, ‘‘The CBM ECAL,’’ *Journal of Physics: Conference Series*, vol. 798, p. 012164, jan 2017.

[35] F. Guber and I. Selyuzhenkov, eds., *Technical Design Report for the CBM Projectile Spectator Detector (PSD)*. Darmstadt: GSI, 2015.

[36] V. Friese, ‘‘Simulation and reconstruction of free-streaming data in CBM,’’ *Journal of Physics: Conference Series*, vol. 331, p. 032008, 12 2011.

[37] Workman, R. L. et al. (Particle Data Group), ‘‘Review of particle physics,’’ *Prog. Theor. Exp. Phys., 083C01*, vol. 2022, 2022.

[38] J. Rafelski and B. Müller, ‘‘Strangeness production in the quark-gluon plasma,’’ *Phys. Rev. Lett.*, vol. 48, pp. 1066–1069, Apr 1982.

[39] J. de Cuveland and V. L. (for the CBM Collaboration), ‘‘A First-level Event Selector for the CBM Experiment at FAIR,’’ *Journal of Physics: Conference Series*, vol. 331, p. 022006, dec 2011.

[40] I. Kisel, I. Kulakov, and M. Zyzak, ‘‘Standalone first level event selection package for the cbm experiment,’’ in *2012 18th IEEE-NPSS Real Time Conference*, pp. 1–6, 2012.

[41] W. Cassing and E. L. Bratkovskaya, ‘‘Parton transport and hadronization from the dynamical quasiparticle point of view,’’ *Physical Review C*, vol. 78, sep 2008.

[42] M. Bleicher, E. Zabrodin, C. Spieles, S. A. Bass, C. Ernst, S. Soff, L. Bravina, M. Belkacem, H. Weber, H. Stöcker, and W. Greiner, ‘‘Relativistic hadron-hadron collisions in the ultra-relativistic quantum molecular dynamics model,’’ *Journal of Physics G: Nuclear and Particle Physics*, vol. 25, p. 1859, sep 1999.

[43] V. Akishina and I. Kisel, ‘‘Parallel 4-Dimensional Cellular Automaton Track Finder for the CBM Experiment,’’ *J. Phys. Conf. Ser.*, vol. 762, no. 1, p. 012047, 2016.

[44] V. Akishina, *Four-dimensional event reconstruction in the CBM experiment*. PhD thesis, J. W. Goethe University, Frankfurt (Main), 2017.

[45] J. Kubát, ‘‘Reconstruction of strange hadrons in collisions of nuclei at RHIC,’’ Master’s thesis, Czech Technical University in Prague, Faculty of Nuclear Sciences and Physical Engineering, Praha 1 - Staré Město, Břehová 7 - PSČ 115 19, August 2020.

[46] ‘‘GSI Scientific Report 2016,’’ Tech. Rep. GSI Report 2017-1, Darmstadt, 2017.

[47] I. Kisel and I. Kulakov and M. Zyzak, ‘‘KFParticleTopoReconstructor.cxx.’’ Git repository, 2013. https://git.cbm.gsi.de/f.uhlig/KFParticle/-/blob/master/KFParticle/KFParticleTopoReconstructor.cxx, Accessed: April 5, 2023, Version 1.0.

[48] Y. Gorbunov, ‘‘Star - armenteros podolanski plots.’’ https://www.star.bnl.gov/~gorbunov/main/node48.html, 2010. Accessed on April 1, 2023.

[49] NVIDIA, ‘‘Deep Learning Super Sampling (DLSS).’’ https://www.nvidia.com/de-de/geforce/technologies/dlss/, March 05 accessed 2023.

[50] NVIDIA, ‘‘NVIDIA RTX A6000.’’ https://www.nvidia.com/en-us/design-visualization/rtx-a6000/, March 05 accessed 2023.

[51] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[52] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Weinberger, ‘‘Deep networks with stochastic depth,’’ 2016.

[53] F. Rosenblatt, ‘‘The perceptron: a probabilistic model for information storage and organization in the brain,’’ *Psychological review*, vol. 65 6, pp. 386–408, 1958.

[54] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, ‘‘Learning internal representations by error propagation,’’ 1986.

[55] A. Krizhevsky, I. Sutskever, and G. E. Hinton, ‘‘Imagenet classification with deep convolutional neural networks,’’ in *Advances in Neural Information Processing Systems* (F. Pereira, C. Burges, L. Bottou, and K. Weinberger, eds.), vol. 25, Curran Associates, Inc., 2012.

[56] K. He, X. Zhang, S. Ren, and J. Sun, ‘‘Deep residual learning for image recognition,’’ 2015.

[57] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, ‘‘Deepface: Closing the gap to human-level performance in face verification,’’ in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1701–1708, 2014.

[58] J. Eshraghian and W. Lu, ''The fine line between dead neurons and sparsity in binarized spiking neural networks,'' 01 2022.

[59] B. Xu, N. Wang, T. Chen, and M. Li, ''Empirical evaluation of rectified activations in convolutional network,'' 05 2015.

[60] P. Ramachandran, B. Zoph, and Q. V. Le, ''Searching for activation functions,'' *CoRR*, vol. abs/1710.05941, 2017.

[61] A. D. Rasamoelina, F. Adjailia, and P. Sinčák, ''A review of activation function for artificial neural network,'' in *2020 IEEE 18th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, pp. 281–286, 2020.

[62] M. A. Mercioni and S. Holban, ''P-Swish: Activation Function with Learnable Parameters Based on Swish Activation Function in Deep Learning,'' in *2020 International Symposium on Electronics and Telecommunications (ISETC)*, pp. 1–4, 2020.

[63] X. Ying, ''An overview of overfitting and its solutions,'' *Journal of Physics: Conference Series*, vol. 1168, p. 022022, feb 2019.

[64] Z. Xie, F. He, S. Fu, I. Sato, D. Tao, and M. Sugiyama, ''Artificial Neural Variability for Deep Learning: On Overfitting, Noise Memorization, and Catastrophic Forgetting,'' *Neural Computation*, vol. 33, pp. 2163–2192, 07 2021.

[65] 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, pp. 1929–1958, 2014. Submitted 11/13; Published 6/14.

[66] Y. Yao, L. Rosasco, and A. Caponnetto, ''On early stopping in gradient descent learning,'' *Constructive Approximation*, vol. 26, no. 2, pp. 289–315, 2007.

[67] R. Tibshirani, ''Regression shrinkage and selection via the lasso,'' *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[68] A. E. Hoerl and R. W. Kennard, ''Ridge regression: Biased estimation for nonorthogonal problems,'' *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.

[69] S. Ioffe and C. Szegedy, ''Batch normalization: Accelerating deep network training by reducing internal covariate shift,'' in *Proceedings of the 32nd International Conference on Machine Learning* (F. Bach and D. Blei, eds.), vol. 37 of *Proceedings of Machine Learning Research*, (Lille, France), pp. 448–456, PMLR, 07–09 Jul 2015.

[70] P. Luo, X. Wang, W. Shao, and Z. Peng, ''Towards understanding regularization in batch normalization,'' *CoRR*, vol. abs/1809.00846, 2018.

[71] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry, ''How does batch normalization help optimization?,'' in *Advances in Neural Information Processing Systems* (S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds.), vol. 31, Curran Associates, Inc., 2018.

[72] X. Li, S. Chen, X. Hu, and J. Yang, ‘‘Understanding the disharmony between dropout and batch normalization by variance shift,’’ in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[73] S. Ruder, ‘‘An overview of gradient descent optimization algorithms,’’ 2017.

[74] S. Ioffe and C. Szegedy, ‘‘Batch normalization: Accelerating deep network training by reducing internal covariate shift,’’ *CoRR*, vol. abs/1502.03167, 2015.

[75] PyTorch Contributors, ‘‘PyTorch Documentation: Data Handling.’’ `https://pytorch.org/docs/stable/data.html`, 2021. Accessed: March 25, 2023.

[76] A. Zhu, Y. Meng, and C. Zhang, ‘‘An improved adam algorithm using look-ahead,’’ in *Proceedings of the 2017 International Conference on Deep Learning Technologies*, ICDLT ’17, (New York, NY, USA), p. 19–22, Association for Computing Machinery, 2017.

[77] Y. E. Nesterov, ‘‘A method of solving a convex programming problem with convergence rate $O(1/k^2)$,’’ in *Doklady Akademii Nauk*, vol. 269, pp. 543–547, Russian Academy of Sciences, 1983.

[78] M. D. Zeiler, ‘‘ADADELTA: An Adaptive Learning Rate Method,’’ 2012.

[79] D. S. Jones, *Elementary Information Theory*. Oxford New York: Clarendon Press ; Oxford University Press, 1979.

[80] C. Broyden, ‘‘A new double-rank minimisation algorithm. preliminary report,’’ *American Mathematical Society, Notices*, vol. 16, p. 670, 1969.

[81] R. Fletcher, ‘‘A new approach to variable metric algorithms,’’ *The Computer Journal*, vol. 13, pp. 317–322, 01 1970.

[82] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, ‘‘Pytorch: An imperative style, high-performance deep learning library.’’ `https://pytorch.org/`, 2019. Accessed April 2, 2023.

[83] F. Chollet and J. Allaire, ‘‘Keras.’’ `https://github.com/keras-team/keras`, 2015. Accessed April 2, 2023.

[84] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, ‘‘Tensorflow: A system for large-scale machine learning.’’ `https://www.tensorflow.org/`, 2016. Accessed April 2, 2023.

[85] Frankfurt Institute for Advanced Studies, ‘‘ANN4EUROPE - Artificial Neural Networks for the Data-Driven Revolution in European Science.’’ `https://fias.institute/de/projekte/ann4europe/`, n.d. Accessed March 28, 2023.

[86] ‘‘torch.nn.Linear - pytorch 1.9.0 documentation.’’ `https://pytorch.org/docs/stable/generated/torch.nn.Linear.html`, 2023. Accessed March 30, 2023.

[87] D. P. Kingma and J. Ba, ‘‘Adam: A method for stochastic optimization,’’ in *3rd International Conference for Learning Representations*, 2015. Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.