AGH University of Science and Technology
Faculty of Physics and Applied Computer Science

Doctoral thesis

Adam Dendek

Machine learning based long-lived particle reconstruction algorithm for Run 2 and upgrade LHCb trigger and a flexible software platform for the UT detector read-out chip emulation

Supervisor: **dr hab. inż. Tomasz Szumlak**

Cracow, January 2021

Declaration of the author of this dissertation:

Aware of legal responsibility for making untrue statements I hereby declare that I have written this dissertation myself and all the contents of the dissertation have been obtained by legal means.

Declaration of the thesis Supervisor:

This dissertation is ready to be reviewed.

Dedication

I dedicate this thesis to my Mother for nursing me with affections and love and her dedicated partnership for success in my life.

Machine learning based long-lived particle reconstruction algorithm for Run 2 and upgrade LHCb trigger and a flexible software platform for the UT detector read-out chip emulation

Streszczenie

Niniejsza rozprawa doktorska składa się z opisu dwóch projektów badawczych zrealizowanych w ramach współpracy LHCb. Pierwszy z nich jest dedykowany opisowi prac nad algorytmem do rekonstrukcji śladów pochodzących od cząstek długożyciowych. W ramach prowadzonych badań zdecydowano się zastosować nowatorskie metody uczenia maszynowego w celu poprawy czystości i wydajności rekonstrukcji. Projekt ten jest jednym z pierwszych, który używa zaawansowanych modeli uczenia maszynowego w ramach systemu wyzwalania (tryggera) wysokiego poziomu. W ramach studiów nad analizą wydajności testowanych modeli wykonano nowatorską analizę interpretowalności predykcji modeli uczenia maszynowego.

Druga część pracy przedstawia zaprojektowanie i zaimplementowanie platformy do emulacji i monitoringu algorytmów przetwarzania surowych danych zbieranych przez projektowany detektor UT (ang. Upstream Tracker) w ramach modernizacji detektora LHCb. W wyniku tych prac została dostarczona aplikacja TbUT. Aplikacja ta była wykorzystywana podczas szeregu testów na wiązce, których celem było sprawdzenie poprawności projektowanych sensorów oraz elektronicznego układu odczytu front-end Salt. W przyszłości oprogramowanie to będzie wykorzystywane między innymi do wykonywania kalibracji i monitorowania poprawności działania detektora UT.

Rozprawa doktorska rozpoczyna się od wstępu, który skupia się przedstawieniu eksperymentu LHCb oraz wyjaśnieniu zasady działania każdego z elementów detektora, jak również motywacji do jego modernizacji. W kolejnym rozdziale zostały przedstawione aspekty teoretyczne dotyczące Modelu Standardowego ze szczególnym uwzględnieniem oddziaływań słabych oraz problemu łamania symetrii kombinowanej *CP*, będącej motywacją do powstania eksperymentu LHCb. Rozdział trzeci skupia się na przedstawieniu i dogłębnej analizie algorytmów uczenia maszynowego. Dyskutowane są zarówno matematyczne podstawy wybranych modeli, jak również procesu ich trenowania oraz optymalizacji ich hiper parametrów. Czwarty rozdział jest dedykowany przedstawieniu prac w ramach poprawy algorytmu rekonstrukcji śladów cząstek długożyciowych. Rozdział ten składa się z przedstawienia algorytmu rozpoznawania wzorców oraz studiów nad dwoma klasyfikatorami opartymi o algorytmy uczenia maszynowego.

Rozdział piąty przedstawia wstęp teoretyczny dotyczący oddziaływania promieniowania oraz materii, czy zasady działania krzemowego detektora promieniowania jonizującego, po czym przedstawione jest oprogramowanie TbUT. Szósty rozdział przedstawia analizę danych zebranych podczas testów na wiązce w szczególności skupiając się na problemie współdzielenia ładunku. Rozprawa kończy się podsumowaniem i wnioskami zebranymi w rozdziale siódmym.

Author's Contribution

During my P.h.D studies, I made a contribution to the 1000 person LHCb particle physics experiment, thus the work of this thesis is a combination of the author's own contribution and the contribution of others. I have been involved in two main projects.

The first one was related to the improvement of the Downstream Tracking algorithm described in Chapter 4. The goal of this project was to enhance one of the track reconstruction algorithms dedicated to finding so-called, long-lived neutral particles via the application of Machine Learning. This project has allowed me to get practical knowledge on building a full Machine Learning pipeline and track reconstruction algorithm. I was responsible for preparing, cleaning, and visualization of the training data, selecting the classification model, monitoring and visualization training progress, validating, and interpretation of the model's prediction. The algorithm has passed a very demanding testing procedure and has subsequently been commissioned and added to the real-time event trigger system of the LHCb experiment. Secondly, I was a member of the LHCb UT testbeam team. Within this project, I made a significant contribution to modernizing the LHCb detector by implementing the whole monitoring processing chain for the Upstream Tracker (UT). This software, called TbUT, is described in Chapter 5. It was used to analyze the data collected during the number of testbeam campaigns as well as it will be a core component of the future monitoring and calibration tool for the UT detector. Besides implementing the aforementioned software, I took part in data-taking shifts at CERN. I made a contribution to this data analysis, which is a topic of Chapter 6.

In addition to the research duties, I have an excellent teaching opportunity at both undergraduate and graduate levels. I was responsible for AGH course Python in the Enterprise. During this course, the students have the opportunity to get familiar with such concepts as Unit Testing, Design Patterns, Continuous Integration, and Machine Learning. The key part of my teaching duties was supervising group projects. The projects were focused mostly on image processing using Deep Learning (car plates recognition, sudoku solver using camera image captured, emotion detection).

The list of papers with my contribution, conferences, and summer schools that I participated in is provided below.

Publications

- **Machine learning based long-lived particle reconstruction algorithm for the LHCb experiment,**
  NeuralPS workshop "Machine Learning and the Physical Sciences" (paper,poster);

- **PatLongLivedTracking: a tracking algorithm for the reconstruction of the daughters of long-lived particles in LHCb;**
  LHCb-PUB-2017-001;

- **Emulation and Calibration of the SALT Read-out Chip for the Upstream Tracker for Modernised LHCb Detector;**
  Acta Phys. Pol. B 46 (2015) 1263-1269;

- **Testbeam studies of pre-prototype silicon strip sensors for the LHCb UT upgrade project;**
  Nucl.Instrum.Meth. A806 (2016) 244-257;

- **Signal coupling to embedded pitch adapters in silicon sensors;**
  Nucl.Instrum.Meth. A877 (2018) 252-258.

Conferences

- **ML in PL Conference 2019** Machine learning in High Energy Physics (recording);

- **76th LHCb Analysis and Software week** Building and validating MVAs, How to build more reliable ML models (slides);

- **8th International Conference on New Frontiers in Physics (ICNFP 2019);** Machine Learning techniques used in LHCb analyses and online applications;

- **LHCP, Bologna, 4-9 June 2018** A tracking algorithm for the reconstruction of the daughters of long-lived particles in LHCb;

- **Connecting the Dots/Inteligent Tracker; 2017;** Deep Neural Nets and Bonsai BDTs in the LHCb pattern recognition;

- **XXII Cracow Epiphany Conference on Physics in LHC Run II; 2016;** Calibration and monitoring of the SALT readout ASIC for the LHCb UT detector;

- **XXI Cracow Epiphany Conference on future of High Energy Collider; 2015;** Emulation and calibration of the SALT readout chip for the UT tracker for modernised LHCb detector.

Summer schools

- **Wolfram Summer School**, July 2017, Waltham MA, USA,
  - Worked on project DeepLaetitia: Deep Reinforcement Learning That Makes You Smile (project summary);
- **Second Machine Learning in High Energy Physic Summer School 2016**, July 2016, Lund Sweden
- **The 3rd Asia-Europe-Pacific School of High Energy Physics**, October 2016, Beijing China
- **The 38th CERN School of Computing**, September 2015, Kavala Greece

Acknowledgments

If somebody who is about to submit an application for a Ph.D. position asked me what I think about this idea, I would tell him/her that a Ph.D. study can be compared to a marathon race. In my case, it took six years to complete it, and it wasn't painless. I had a lot of moments where I was within an inch of resigning. However, I reached the moment when my thesis is done, and I am waiting for reviews. Therefore, I would like to say "thank you" to some people who helped me be where I am. First things first, I would like to thank my supervisor, prof. T. Szumlak. You gave me an opportunity to work on a project that involved applying machine learning techniques in a complicated scientistic scenario that forced me to study them. This knowledge will drive my future career. Secondly, I would like to thank You for allowing me to be your teaching assistant (AGH course "Python in the Enterprise"). As Richard Feynman once said, "If you want to master something, teach it," and for me, teaching and supervising a team of undergraduate students was a very productive time. According to the surveys, most of the students enjoyed this course and thought of it as one that helped them to find a better job. Finally, I would like to thank You for the review of this document.

Secondly, I wish to thank all members of the Upstream Tracker Testbeam team. The experience of testbeams at CERN was one that shaped me as a researcher. Special thanks go to prof. Steven Blusk. You are a great researcher, and you showed me a proper way how to approach scientific problems. Our cooperation, even though not successful as it should have been, taught me an unforgivable lesson. Moreover, I would like to thank Adam Davis. You supported me for so long, and you were kind and patient in answering my silly question. And finally, I would like to mention Constantin "Stan" Weisser. I learn from you one vital skill - to be bold and not being afraid of asking questions.

These acknowledgments would not be complete without a special thanks to my old-time best friends from high school (sorted in alphabetically ascending order using the first name as a key) Bartek, Dawid, Karol, Konrad, Szymon, Tomasz, Wiktor. I hope that we stay together regardless of the distance that may separate us.

I also wish to thank Łukasz Fulek for many discussions about physics, programming, and life in general.

This is eventually time to thank my fiancée Kasia, for sharing your life with me and for making each every minute we spend together a very special moment.

Ostatni paragraf w tej sekcji podziękowań chciałbym dedykować mojej Mamie. Zdaję sobię sprawę, że nie jestem w stanie słowmi określić swojej wdzięczności za wszystko, co od Ciebie otrzymałem. Pomimo, że dzielą nas duże odległości jesteś zawsze przy mnie, kiedy potrzebuję pomocy albo porady. W szczególności jestem wdzięczny za danie mi możliwości podejmowania oraz ponoszenia konsekwencji samodzielnych wyborów.

Contents

Success is to be measured not so much by the position that one has reached in life as by the obstacles which he has overcome while trying to succeed.

Booker T. Washington

Preface

Dear Reader!

I am glad you are reading this dissertation.

Writing the thesis is not an easy task. I have to summarize six years of my professional life into about two hundred pages, so let me start by presenting my PhD project scope. Be aware that the preface section contains several expressions that are not explicitly explained. However, the remained of this thesis assures you will find a detailed discussion on most of them. I put a lot of effort into making this thesis as clear and understandable as possible.

All of my activities during the doctoral studies were related to one of four biggest, currently operating experiments at The European Organization for Nuclear Research CERN (fr. Organisation européenne pour la recherche nucléaire). It is called LHCb, and it stands for the Large Hadron Collider beauty experiment. The most vital tool that is at the disposal of the LHCb collaboration is the LHCb spectrometer, shown in Figure 0.0.1. Since there is not much to see besides the supporting steel structures, I also added a schematic cross-section of the experimental setup in Figure 0.0.2. All major components, also called sub-detectors, are described in Chapter 1.

From the physics point of view, the LHCb physics program is primarily focused on studying *CP* violation, and rare phenomena in B (beauty) and C (charm) meson decays and searching for New Physics. Chapter 2 is dedicated to providing a brief description of the physics behind the LHCb experiment. The high-quality physics results obtained during the LHC Run 1 and Run 2, proved excellent performance of the detector. The list of outstanding physics results published by the LHCb Collaboration is extraordinary. For instance, LHCb collaboration was able to measure the very rare processes, such as $B_0 \rightarrow \mu\mu$, occurring for once for every ten billion B_0 mesons [90] and the very first measurement of pentaquark state [51].

Until now, no physics phenomena beyond the Standard Model's prediction have been found. No new heavy particle was discovered, apart from the Higgs boson, and the precision measurements may be the only way to detect the new effects at LHC. However, to study such processes, the collection of a significant amount of data is vital. Unfortunately, the data collection rate is limited by the current detector design, in particular, by the throughput of the trigger system, which is described in section 1.3.3. The LHCb detector is undergoing a major upgrade to overcome this limitation. The crucial part of the Upgrade project is the replacement of the entire readout system, which is currently limited by the hardware Level-0 trigger. In consequence, the high-level trigger (HLT) can only process data at a rate equal to 1.1 MHz. The new upgraded system will allow the full event readout at the LHC clock rate (40 MHz). The machine bunch structure will be chosen in that way that the crossing rate at the LHCb collision point will occur with the frequency of 30 MHz, and the HLT system will process each event in real-time.

This goal can be achieved by replacing both read-out electronics and sensitive elements of the detectors. One of the most challenging parts of the Upgrade is research and development related to the design and test of the new tracking detector called Upstream Tracker. This silicon micro-strip detector will be placed just before the bending magnet, and it is supposed to replace the current TT tracker. The detailed description of the UT detector can found in section 1.5. The replacement of the current TT detector is motivated by three facts. First of all, the TT design doesn't allow the survival of the expected radiation dose deposited under the upgrade data-taking conditions, particularly in the inner, close to the proton-beam region. Secondly, the current sensor's granularity could lead to unacceptably high occupancy. Finally, the front-end Beetle chip, which is an essential part of the read-out system, cannot process the raw data at the beam crossing rate (40 MHz). What makes the situation worse is that the front-end hybrids, which were designed to support the Beetle chip, are part of the mechanical structure of the detector and cannot be replaced without damaging them. Besides, the new detector is designed to improve the LHCb acceptance.

I was personally involved in the activities connected to the testing and verification of the UT silicon sensors. I participated in the number of the testbeam experiments, I designed and implemented a complete emulation platform for the raw data processing and data analysis.

The testbeam experiments play a vital role in the new detector R' &D' process. It is crucial to quantify

Figure 0.0.1: View of the detector LHCb. The image was taken from the CERN public website.

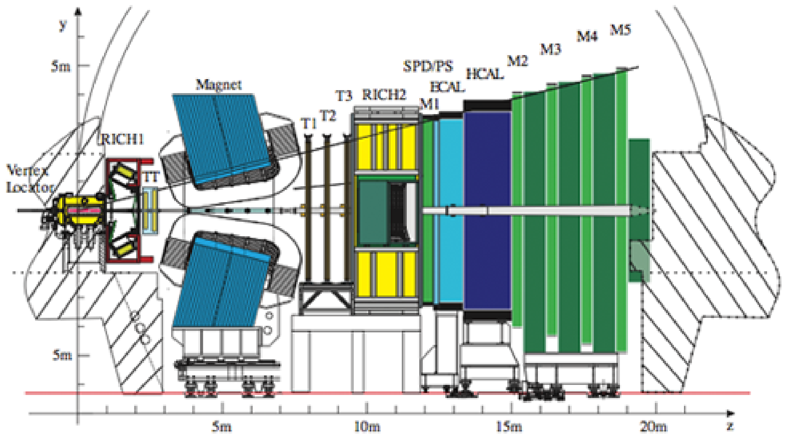


Figure 0.0.2: The layout of the LHCb detector, viewed from the side. The LHCb detector components from left to right: proton-proton interaction point, Vertex Locator Velo, Ring Imaging Cherenkov detector one (RICH1), TT, Magnet, T stations, RICH2, electromagnetic and hadronic calorimeter (ECAL and HCAL), and muon stations. Figure taken from [17].

the performance of the various sensors that have been subjected to the maximal radiation dose expected for a given sensor during the whole lifetime of the UT detector. Furthermore, the testbeams provide realistic test-beds to confirm the expected performance of the entire data read-out chain, including the front-end ASICs. During the testbeams, we collected the data, which allowed us to study, for instance, Landau distribution as a function of the bias voltage, cluster sizes versus bias voltage, and resolution vs angle. All of the mentioned studies were performed for both irradiated and unirradiated sensors. The detailed description of the testbeam data analysis is a topic of Chapter 6.

Before we could analyze the testbeam data, we had to design and develop software for raw data processing. I was the leading developer and the one who was responsible for software maintenance. Its flexible design allows the process of data collected by the various DAQ electronics during the entire R' &D' phase. The detailed description of the mentioned framework is a subject of Chapter 5. Furthermore, the software will be used to monitor the performance of the data collected during the entire UT detector's life. It will be a crucial part of the future platform to detector calibration.

Moreover, as a member of the LHCb collaboration, I was involved in an improvement of the Downstream Tracking algorithm using the computational intelligence approach. You will find a more detailed description of the tracking algorithm in Chapter 4. Briefly, the tracking is a procedure that is designed to reconstruct the trajectory of the particles that were created as a result of the proton-proton collisions using nothing more but the electronics signals provided by the position-sensitive detectors. The reconstruction algorithm is executed as a part of a real-time system, namely trigger procedure. Therefore its time budget is minimal. However, due to the number of particles created during each beam crossing, the previous implementation of the tracking procedures often made mistakes. Those mistakes correspond to reconstructions of the fake, also called the ghost tracks. To avoid such a situation, we decided to leverage Machine Learning and Deep Learning techniques. I enhanced the tracking procedure by adding the Machine Learning classifier, which was trained to distinguish whether the partially reconstructed track is true or not. As far as I know, the LHCb is the only one currently operating a High Energy Physics experiment, which makes use of advanced Machine Learning models as a part of the online trigger. During the development, I familiarized myself with the concept of building the entire Machine Learning pipeline using open-source tools like sklearn, XGBoost, and PyTorch. Such technologies are widely used in both academia and industry. If you want to know more about Machine Learning and the procedure, how to build and deploy the model, please take some time to read the second part of Chapter 3.

I hope you enjoy reading this thesis.

In that sense, this new knowledge has all to do with honor and country, but it has nothing to do directly with defending our country except to help make it worth defending.

Robert R. Wilson during the Congressional Testimony
on building Fermilab's first accelerator

1
The LHCb Experiment and its Upgrade

This chapter is split into two parts, and the first one is dedicated to present a detailed description of the LHCb detector that was deployed during both Run I and Run II. This part starts by presenting the CERN (The European Organization for Nuclear Research). Then a basic understanding of how the LHC (Large Hadron Collider) works is explained. The mentioned section is followed by a description of the LHCb detector, which consists of a paragraph dedicated to each sub-detectors. The final section discusses the LHCb Upgrade by presenting its motivation and summarizing all changes that have been planned.

1.1 CERN

The European Organization for Nuclear Research CERN is the world's largest scientific organization in the field of High Energy Physics. It was established in 1954 by twelve western European countries to create one of the very first European joint ventures. Currently, the CERN associate 22 members state, including Poland. The Institution is based on the Swiss-France border very close to Geneva. The CERN's primary focus is to design and construct instruments to study the fundamental building blocks of matter and its interactions.

1.2 Large Hadron Collider

The Large Hadron Collider (LHC) is the world's largest circular particle accelerator. It is installed in the 26.7 km long tunnel that was constructed for the previous experiment, Large Electron Positon Collider (LEP) [80]. The tunnel is situated about 100 m below the ground. The LHC is designed to accelerate

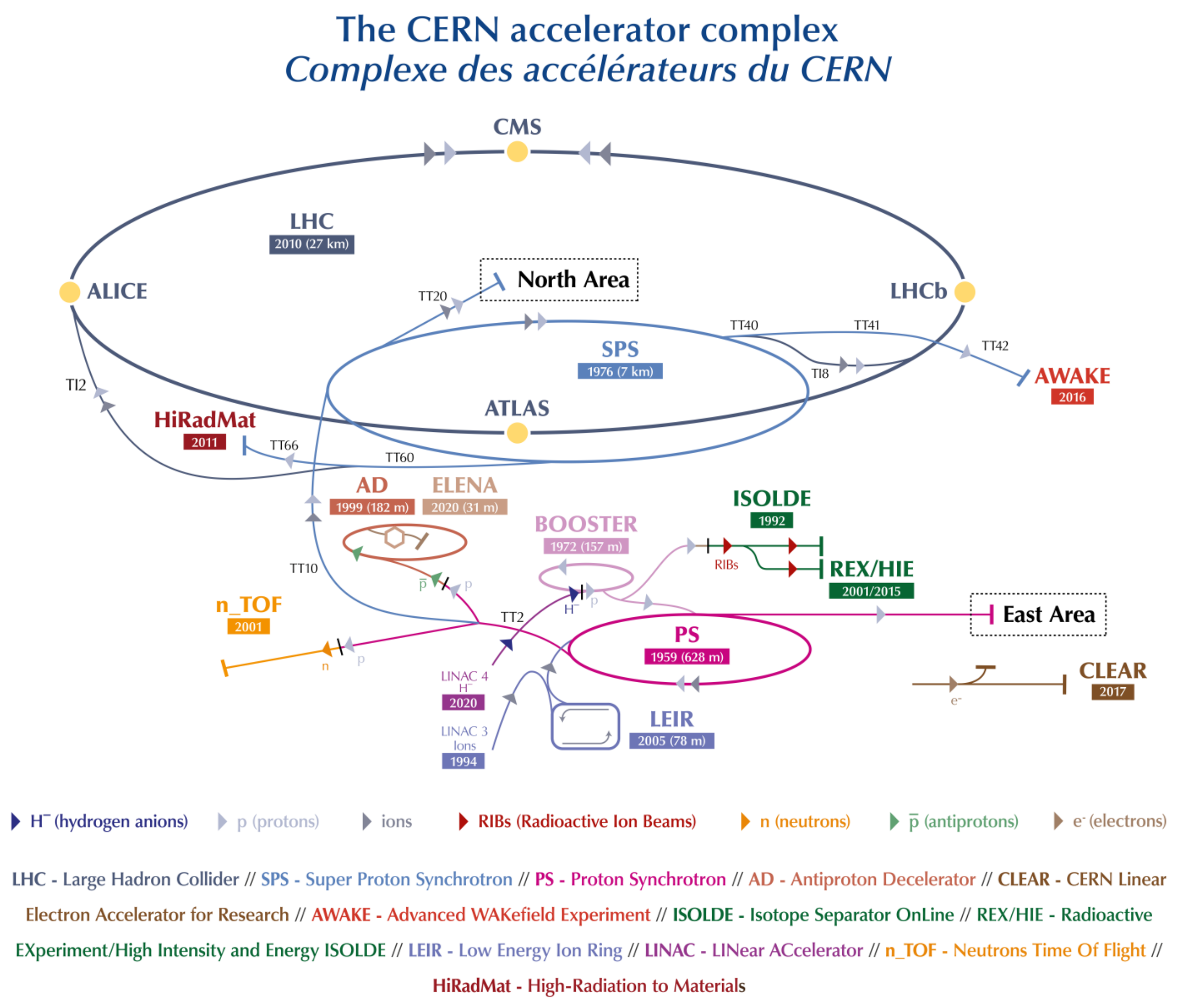

The CERN accelerator complex
Complexe des accélérateurs du CERN
CMS
LHC
2010 (27 km)
ALICE
North Area
LHCb
TT20
TT40
TT41
TT42
SPS
1976 (7 km)
TI8
AWAKE
2016
TI2
HiRadMat
2011
TT66
ATLAS
TT60
AD
1999 (182 m)
ELENA
2020 (31 m)
ISOLDE
1992
BOOSTER
1972 (157 m)
RIBs
REX/HIE
2001/2015
TT10
n_TOF
2001
TT2
East Area
PS
1959 (628 m)
CLEAR
2017
LINAC 4
2020
LINAC 3
Ions
1994
LEIR
2005 (78 m)
H⁻ (hydrogen anions)
p (protons)
ions
RIBs (Radioactive Ion Beams)
n (neutrons)
p̄ (antiprotons)
e⁻ (electrons)
LHC - Large Hadron Collider // SPS - Super Proton Synchrotron // PS - Proton Synchrotron // AD - Antiproton Decelerator // CLEAR - CERN Linear Electron Accelerator for Research // AWAKE - Advanced WAKefield Experiment // ISOLDE - Isotope Separator OnLine // REX/HIE - Radioactive EXperiment/High Intensity and Energy ISOLDE // LEIR - Low Energy Ion Ring // LINAC - LINear ACcelerator // n_TOF - Neutrons Time Of Flight // HiRadMat - High-Radiation to Materials

Figure 1.2.1: The LHC accelerator complex showing all accelerator facilities and the four main experiments, denoted by a yellow dots. The North Area is the location where all testbeam data were collected, see chapter 6. Figure taken from [95]

the protons and heavy ions. At the nominal performance, the LHC delivers the two protons beams of energy 6.5 TeV. This corresponds to the center-of-mass collision energy of 13 TeV. To achieve this performance, the particle acceleration is done by a series of accelerators. Each of them progressively boosts the energy of the beam. Figure 1.2.1 shows the LHC accelerator complex.

The entire boosting process starts from the small red bottle full of hydrogen. It is shown in Figure 1.2.2. This bottle is the only and sufficient proton source for the entire, massive LHC acceleration system. This shows how sophisticated and resource-efficient LHC is. Then, the hydrogen atoms are ionized by the external electric field to yield the protons. These particles are injected into the Liniac2, the first linear accelerator in the chain, to boost its energy to the 50 MeV. After that, the beam is inserted into the Proton Synchrotron Booster, followed by the Proton Synchrotron (PS), which pushes the beam to the energy of 25 GeV. The next step in the acceleration sequence is performed by the Super Proton

Figure 1.2.2: The LHC proton source

Synchrotron (SPS). It accelerates the beam to the energy of 450 GeV. [1]

The protons are finally injected into two beam pipes of the LHC. The beam in one pipe circulates clockwise while the beam in the second pipe circulates anticlockwise. It takes 4 minutes and 20 seconds to fill each LHC ring, and 20 minutes for the protons to reach their maximum energy of 6.5 TeV. The two beams interact inside four detectors – ALICE [53], ATLAS [52], CMS [44], and LHCb.

One of the key parameters that describe a particle accelerator (note, that we consider here the circular machine), except for beam energy, is the quantity called instantaneous luminosity. This quantity expresses the ability to produce the required number of interactions by an accelerator, and formally it is the proportionality factor between the number of events per second (also called the event rate) $\frac{dN}{dt}$ and the interaction cross-section σ:

$$\frac{dN}{dt} = L \times \sigma \tag{1.1}$$

The unit of the luminosity is $cm^{-2}s^{-1}$. In practice, the integrated luminosity $\mathcal{L}_{int}$ is often used. Based on this quantity, one can estimate the number of expected events for a given process.

The relationship between the luminosity and the beam parameters for a circular machine, assuming that the beam profile is distributed according to the Gaussian distribution, and there is negligible energy loss during the bunch-bunch collisions is given by:

$$L = \frac{N_b^2 \cdot n_b \cdot f_{rev} \cdot \gamma_r}{4\pi \cdot \varepsilon_n \cdot \beta^*} \cdot F \tag{1.2}$$

where N_b is the number of particles per bunch, n_b the number of bunches per beam, f_{rev} is the revolution frequency, γ_r is the relativistic gamma factor, ε_n the normalized transverse beam emittance, β^* the beta function at the collision point, and F the geometric luminosity reduction factor which originates from the crossing angle at the interaction point.

The LHC was built to deliver a peak luminosity of $10^{34} cm^{-2}s^{-1}$ by colliding 2808 bunches containing

[1] The beam form SPS was used during the testbeam experiments, which is the topic of Chapter 6.

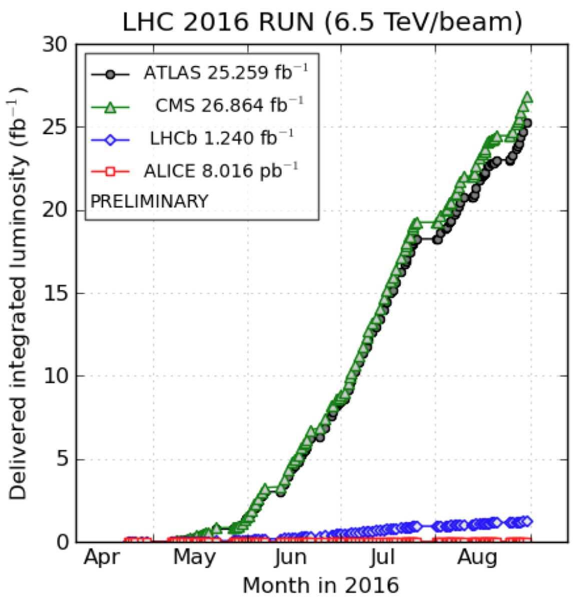


Figure 1.2.3: The integrated luminosity delivered to LHC experiments

approximately $1.1 \cdot 10^{11}$ protons per bunch with the bunch crossing rate of 40MHz (also called the machine clock). For more details on the LHC machine, see [62].

Figure 1.2.3 presents the integrated luminosity delivered to each of the LHC experiments. It is visible that LHCb operates at a significantly lower luminosity level that the remaining general-purpose experiments. The LHCb detector was designed to operate at a luminosity of $2 \cdot 10^{32} cm^{-2}s^{-1}$, which is about two orders of magnitude less than the luminosity delivered to ATLAS and CMS experiments. This is done by purpose since the LHCb experiment focuses primarily on precision, indirect measurements. Operating at a lower luminosity produces fewer interactions, or primary vertices (PVs), per bunch crossing. An increasing number of PVs produce complications in physics analysis, such as tracks being identified coming from the wrong PV. Moreover, operating at lower luminosities induces less radiation damage in the detectors operating very close to the proton beam. The LHCb collaboration implemented the luminosity levelling technique, described in [64], to meet the occupancy requirement.

1.3 LHCb detrctor

The heart of the LHCb experiment is its detector. It was built to produce proton-proton collisions at a centre-of-mass energy of $\sqrt{s}$ =14 TeV. It is located in the cavern that previously was used to host the LEP's experiment Delphi [42]. The unique feature of the LHCb detector is its design. It is significantly different from other general-purpose detectors like ATLAS or CMS, which look like multi-layer barrels surrounding the collision point (so-called 4π geometry). On the contrary, the LHCb is a forward single-arm spectrometer, that was designed to cover the pseudorapidity range of $2 < \eta < 5$ [17]. The pseudorapidity is a spatial coordinate describing the angle of a particle relative to the beam axis. The pseudorapidity can be calculated from the following formula:

$$\eta = -\ln\left[\tan\left(\frac{\theta}{2}\right)\right] = \frac{1}{2}\ln\left(\frac{|\vec{p}| + p_t}{|\vec{p}| - p_t}\right)$$

where θ is the angle between a particle's three-momentum $\vec{p}$ and the positive direction of the beam axis. The p_t (transverse momentum) is a component of the $\vec{p}$ transverse to the beamline. Equation 1.3 allows finding the relationship between the parameter η and the angle θ. When the angle θ gets smaller, then the η rises.

The choice of such layout was motivated by the LHCb physics programme, particularly the angular distribution of the $b\bar{b}$ pairs produced by proton-proton interactions at the LHC energies, fly predominantly into the forward and backward cones (see Figure 1.3.1). LHCb geometrical coverage corresponds to only 4% of whole solid angle, but it can detect approximately 25% of all produced beauty hadrons.

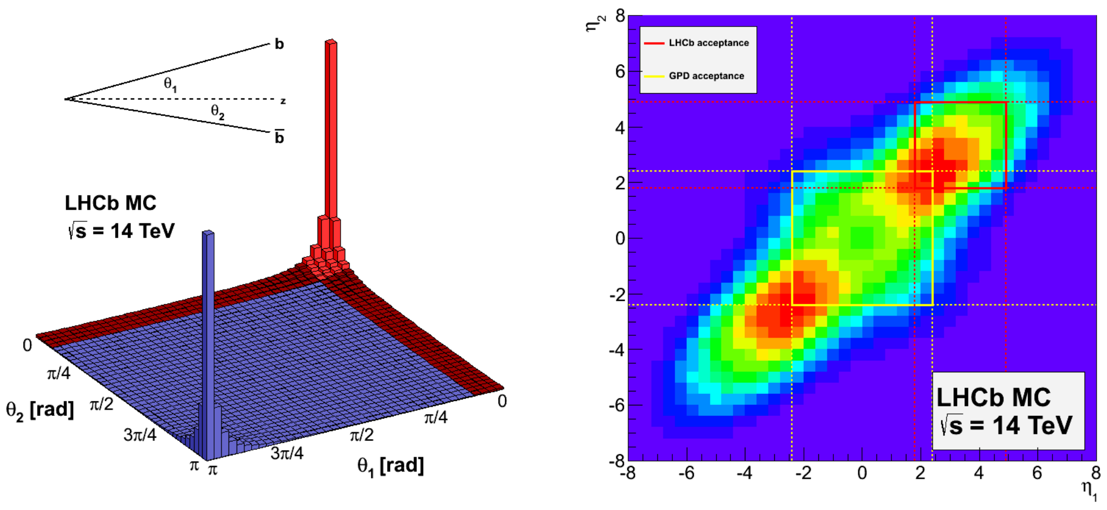


Figure 1.3.1: Production of $b\bar{b}$ quarks at LHC at 14 TeV. The left plot presents the production of the $b\bar{b}$ quarks as a function of polar angle θ. The right plot shows the same distribution as a function of the rapidity of each quark. The pseudorapidity region inside a yellow square corresponds to the acceptance region of the ATLAS and CMS detectors, while the red box highlights the acceptance region of LHCb. Figure taken from [61].

The LHCb, like all of the currently operating High Energy Physics experiments, consists of several sub-detectors, each of which was carefully designed to provide highly efficient system, capable of detecting physics phenomena beyond the Standard Model (BSM). To correctly identify and reconstruct the decays and their kinematical properties, the LHCb detector needs to provide an excellent vertex reconstruction precision, momentum resolution, and particle identification. The following section is dedicated to describing each of the sub-systems.

1.3.1 LHCb tracking sub-system

The tracking system was designed to reconstruct the trajectory of charged particles by combining information from a set of tracking stations. The reconstructed track information is used to estimate the momentum of the charged particles. This estimation is possible due to the magnet's installation, which creates a magnetic field used to bend a particle trajectory. The LHCb tracking system is composed of the Vertex Locator (Velo), the Tracker Turicensis (TT), and the three tracking stations T1, T2, and T3; see Figure 0.0.2. The brief description of each tracking sub-detectors is a topic of this subsection.

1.3.1.1 Velo

The Vertex Locator (Velo) [21] is a silicon strip detector that is located close to the proton-proton collision point, and it is dedicated to providing precise measurements of the position of the primary and secondary vertices [2], which are essential to identify b and c hadrons, which typically traverse about 1 cm at LHCb. The Velo active area is just about 8 mm away from the beamline, which is the world record. Additionally, the Velo allows measuring the Impact Parameter of charged particle's trajectories. Impact Parameter is a transverse distance of closest approach between a particle trajectory and a vertex, most commonly the primary proton-proton interaction vertex, see Figure 1.3.2. The Impact Parameter is widely used in many LHCb data analysis to make a selection that significantly reduces the contamination from the light-quark backgrounds.

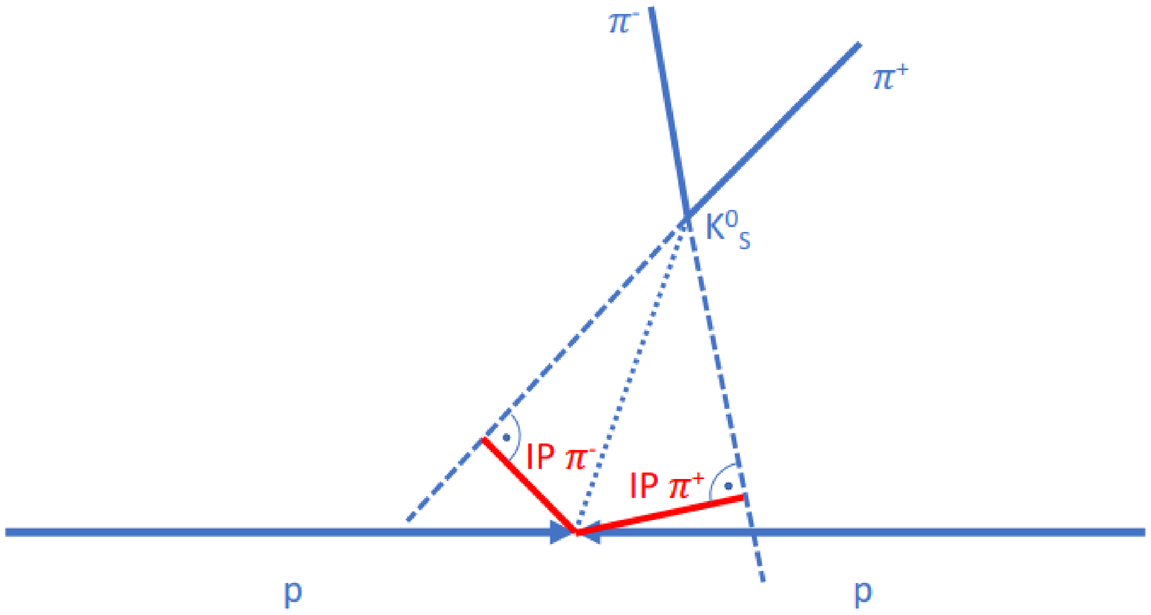


Figure 1.3.2: Graphical interpretation of the Impact Parameter (denoted in red). Figure presents a topology of the K^0_S decaying to two pions.

The Velo detector is comprised of twenty-one silicon tracking stations positioned along the beam axis (z-axis). Each of the tracking stations is divided into two retractable halves, called modules, each

[2] The secondary vertex is a point of decay of short-lived particles, which was created in the primary interactions.

consisting of two silicon microstrip sensors. Figure 1.3.3 presents the layout of the Velo detector. All Velo sensors operate in the vacuum. The Velo vacuum is separated from the beam vacuum by a thin aluminium layer called RF foil.

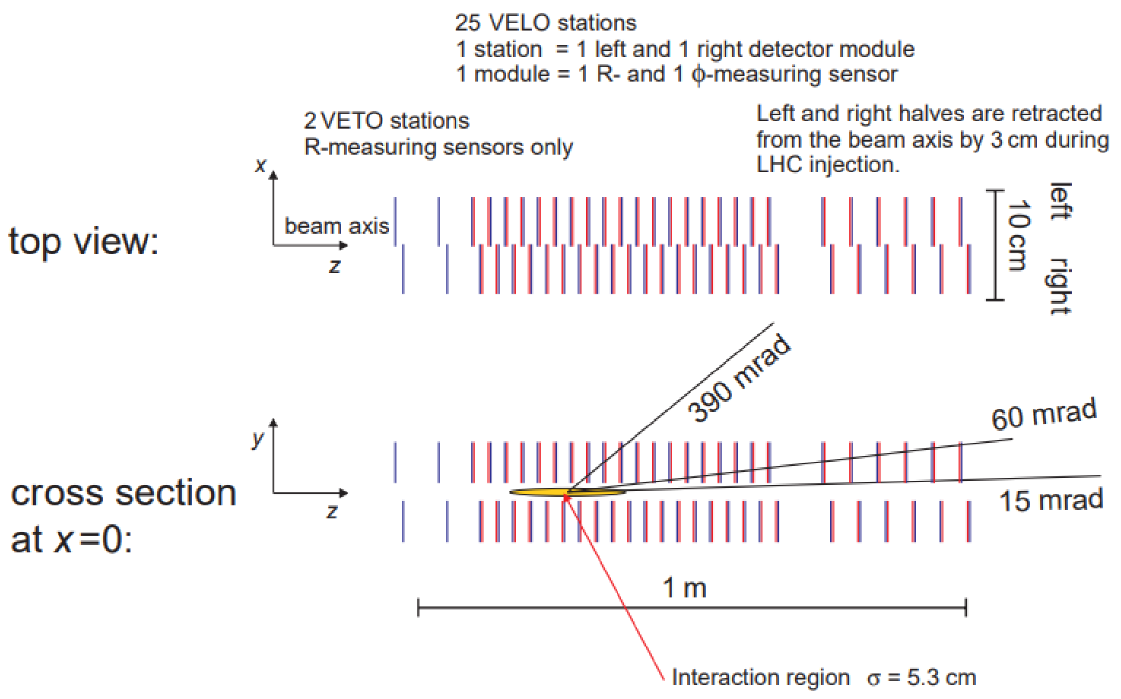

25 VELO stations
1 station = 1 left and 1 right detector module
1 module = 1 R- and 1 ϕ-measuring sensor
2 VETO stations
R-measuring sensors only
Left and right halves are retracted from the beam axis by 3 cm during LHC injection.
top view:
x
beam axis
z
10 cm
left
right
390 mrad
60 mrad
15 mrad
cross section at x=0:
y
z
1 m
Interaction region σ = 5.3 cm

Figure 1.3.3: The layout of the Velo detector. The top picture presents the Velo setup seen from the top, indicating the overlap between the left and right detector's halves. The bottom figure is a cross-section of the Velo at $x = 0$. The black lines indicate the maximum and minimum angular coverage of the Velo and the average angle of the tracks. The figure was taken from [21].

The first type of the Velo sensor is called R-type, and it is dedicated to measuring r-coordinate, i.e., the distance from the proton beam, thus, the strips have a semi-circular shape. The R-type sensors are divided into four sectors in the azimuthal angle to improve the pattern recognition phase of the track reconstruction. The strip pitch [3] increases from $38\mu m$ at the innermost region to $102\mu m$ at the far edge. The ϕ-type sensor is, in turn, divided into two regions, inner and outer, with different pitches to cope with high occupancies. The respective strip topologies are presented in Figure 1.3.4. Both R and ϕ sensors have a thickness of 300 μm. The Velo sensors and read-out electronics are cooled by the evaporated CO_2 system. This system keeps the sensors approximately at the temperature of $-8°$C during data taking.

The read-out of the data is performed by the Beetle front-end ASIC[4] [102]. These chips are placed on the outer edge of the sensor, see Figure 1.3.5. The Beetle chip integrates 128 channels with low-noise charge-sensitive preamplifiers and shapers, an analogue pipelined memory, and a multiplexer.

The primary vertex spatial resolution of about 13 μm in the transverse plane and close 70 μm along the

[3]Pitch is a distance between the centres of two adjacent strip implants
[4]ASIC stands from Application-specific integrated circuit

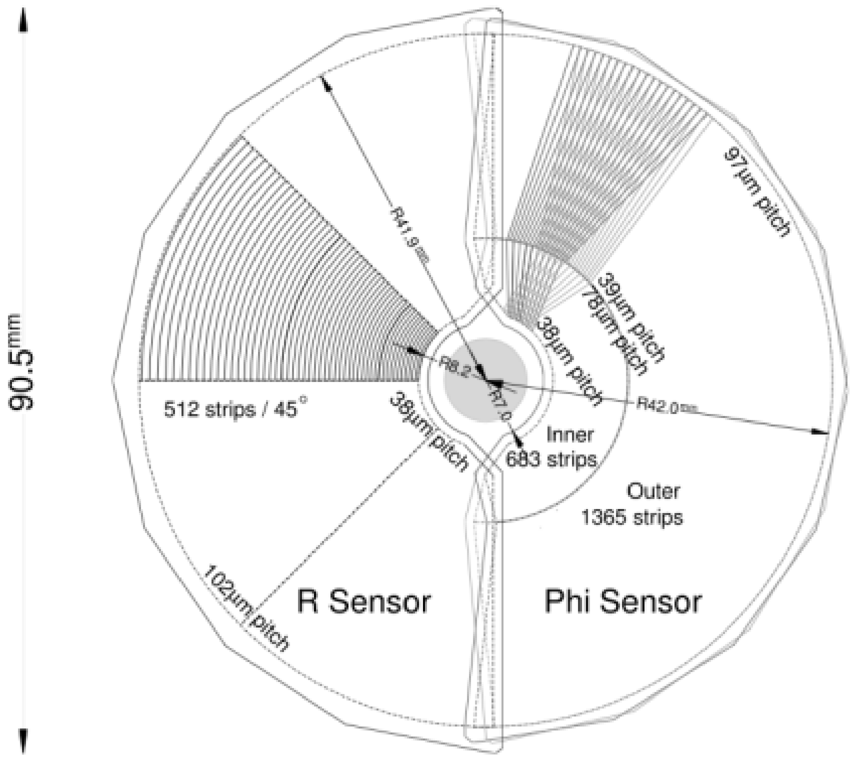


Figure 1.3.4: Geometry of the Velo R and ϕ sensors, with only a small portion of strips visible for clarity (left). Figure was taken from [17].

z axis allows for very precise decay-time measurements that are vital for the LHCb physics programme. The dependence of the primary vertex resolution versus number of tracks obtained using 2012 calibration data is shown in Figure 1.3.6. The resolution of the Impact Parameter, critical for detecting the displaced secondary heavy-flavour decay vertices, depends on multiple scattering, primary vertex resolution and single-hit resolution can be expressed as a function of transverse momentum p_t [113]:

$$\sigma_{IP} = \left(11.5 + \frac{24.5}{p_t[GeV/c^2]}\right) \tag{1.3}$$

1.3.1.2 Silicon Tracker

Silicon Tracker [28] sub-system consists of two detectors based on similar technology; the TT (Tracking Turicensis), upstream to the magnet, while the IT (Inner Tracker) is a part of the tracking stations (T1, T2, T3, see Figure 0.0.2) located downstream to the magnet.

The primary purpose of the TT detector is to reconstruct low-momentum tracks and decays of the long-lived particles, which decay outside of the Velo. The IT detector reconstructs tracks with momentum larger than $1.5GeV/c$, near the beam axis, that passed through the magnetic field. It covers approximately 2% of the T stations acceptance, which corresponds to 20% of all tracks that pass through this

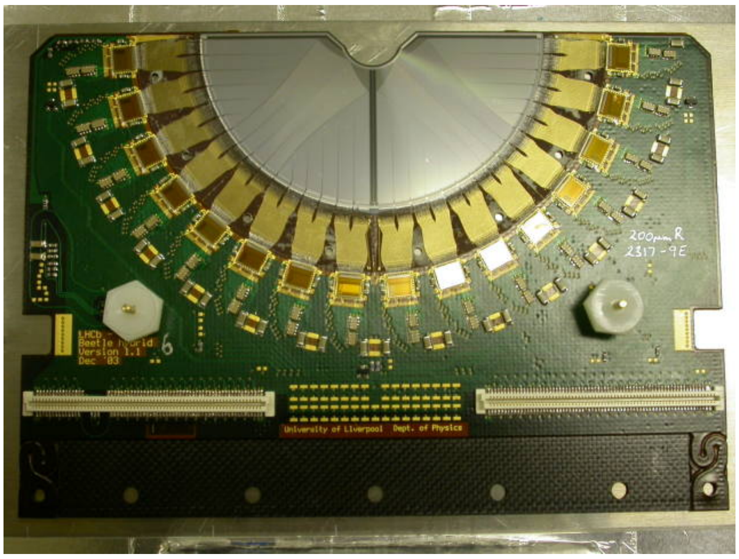


Figure 1.3.5: The photo of a Velo module with 16 readout Beetle chips (right). Figure taken from [17].

detector. On the other hand, TT is designed to cover the full LHCb acceptance region.

Both TT stations (TTa and TTb) are composed of two measuring planes capable of providing a 3d space point for each particle hit. By convention, the sensitive planes are denoted as $(TTaX, TTaU, TTbV, TTbX)$. The X coordinate is measured in the direction perpendicular to the direction of TT sensor vertical strips. Coordinates U and V are identical to the X but tilted by -5° and 5° respectively (see Figure 1.3.7). The distance between the two adjacent layers within each station is about 4 cm, and the distance between the TTa and TTb stations is 27 cm. The TT silicon sensors (p-on-n) are $500\mu m$ thick with a constant strip pitch of $183\mu m$.

One of the quantities that can be used to determine ST performance is the hit efficiency, which can be expressed as a ratio between the number of measured hits to the number of hits expected in a given region. This ratio was measured to be 99.7% for TT and 99.8% for the IT. This measurement was performed on the data collected during Run 1. Another important metric to determine ST performance is the hit spatial resolution. For 2012 the hit resolution was measured to be $53.4\mu m$ for the TT and $54.9\mu m$ for IT.

1.3.1.3 Outer Tracker

The Outer Tracker (OT) [20] is a complementary element to the IT detector, designed to cover the remaining LHCb acceptance region. Each of the OT modules is made of drift-time straw tubes filled with a gas mixture of 70% of Argon and 30% of CO_2. The Drift-time detector reconstructs the hit position by measuring the drift time of the ionization electron to the anode located at the centre of the tube. The distance between the wire and the particle's trajectory is determined by comparing the drift time with bunch crossing signals. The ionization electron is created when a charged particle interacts with a gas. OT achieves drift time less than $50ns$, which allows reconstructing hits with a spatial resolu-

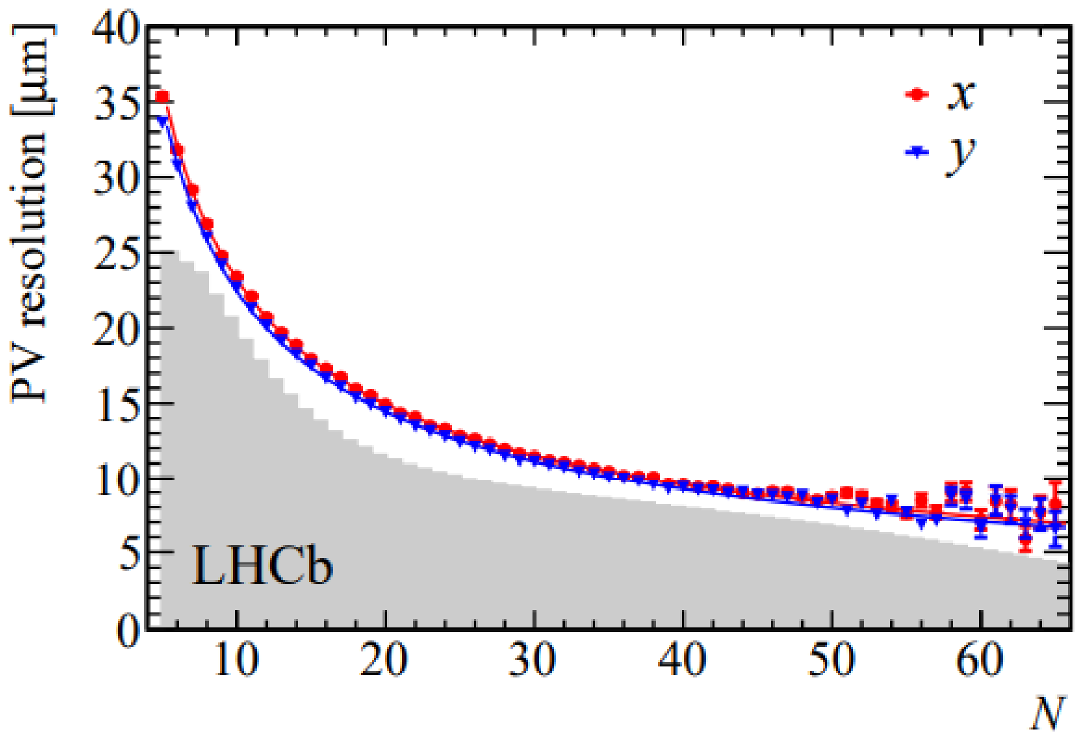


Figure 1.3.6: Primary vertex resolution as a function of a track multiplicity. The blue curve corresponds to x coordinate of the Primary Vertex, and the red one to the y coordinate. The gray histogram presents the number of tracks per reconstructed primary vertex. Presented results were obtained using 2012 calibration data with only one reconstructed primary vertex in the event. Figure taken from [113]

tion of $200\mu m$. OT has a consistent layout of the IT detector, which means it also has four modules in (X, U, V, X) orientation, which is shown in Figure 1.3.8.

1.3.1.4 Magnet

The LHCb Magnet plays a crucial role within the experiment. It bends the trajectory of the charged particles allowing to estimate its momentum. LHCb detector is equipped with a single warm dipole magnet. The magnet is situated between TT and T tracking stations. Figure 1.3.9 shows a photography of the Magnet. It is composed of two identical saddle-shaped coils. These coils are placed inside an iron yoke, that is compatible with the acceptance of the LHCb detector. The coils are made of AI-99.7 alloy with a 25 mm diameter central channel for water cooling.

The estimated momentum resolution depends on the proper measurement of the magnetic field. Therefore, a careful procedure to measure the magnetic field was conducted. As the outcome, the precision of the measurement of the magnetic field is quoted to be $4 \cdot 10\%{-}4\delta B/B$, and the maximal magnitude is 1.04 T, which is shown in Figure 1.3.9.

The systematic errors related to the track reconstruction procedure, which can play a dominant role in the precise measurement of CP asymmetries, can be decreased by operating the magnet at two polarities

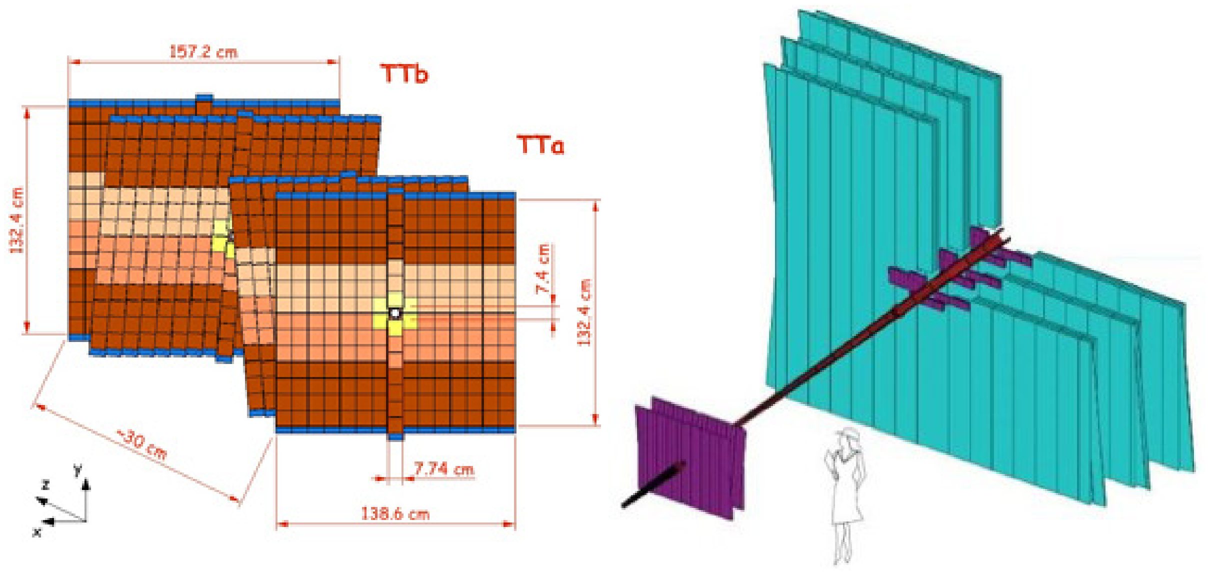


Figure 1.3.7: The layout of the TT station (left) and the schematic view of the whole ST system (plotted in magenta), the cartoon of the woman is shown to indicate the size of each detectors (right). Figure taken from [17].

(positive and negative curves in Figure 1.3.9). The amount of data collected is approximately equal for both polarities, and this split can be used to cross-check systematic reconstruction effects.

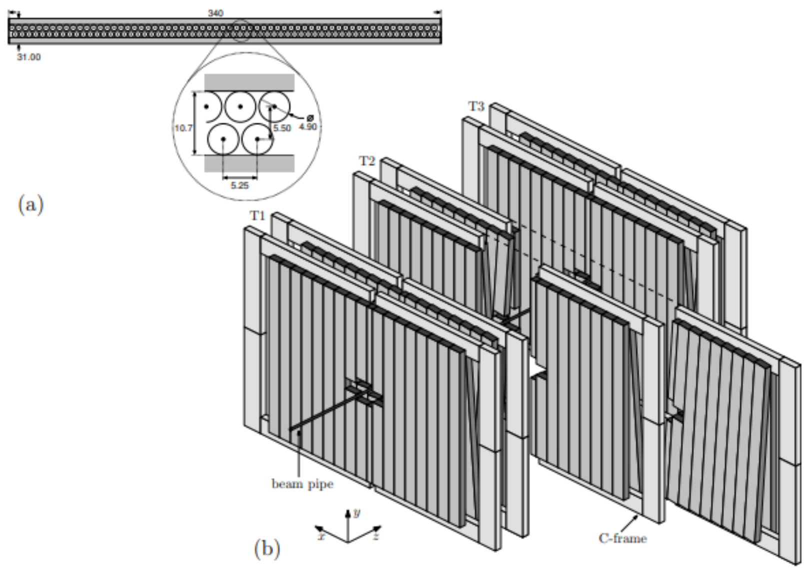
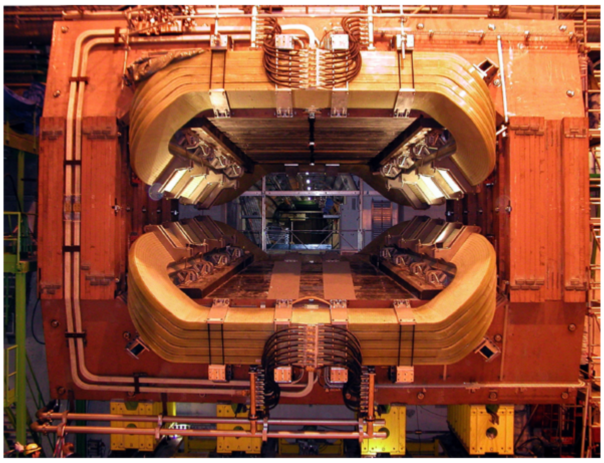


Figure 1.3.8: Schematic view of the OT stations. Figure (a) presents the cross-section of a single OT module, all distances are given in *mm*, and the arrangement of OT modules in layers and stations around the beam pipe (b). Figure taken from [17].



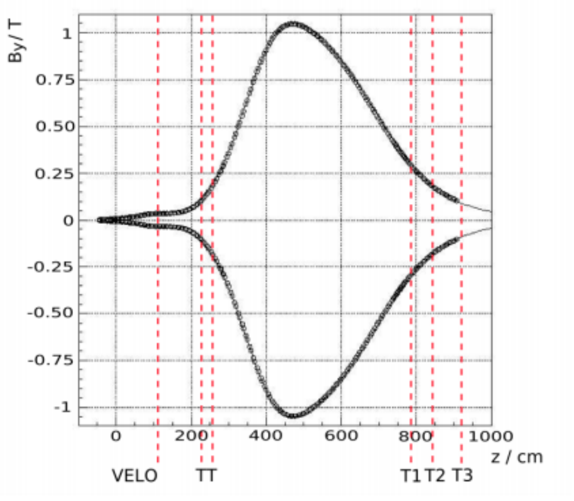


Figure 1.3.9: A photo of the LHCb Magnet taken after its completion in 2004 is shown on the left-hand side. The picture was taken towards the direction of the Velo detector before other sub-detector have been installed, the visible parts are coils (yellowish) and yoke (reddish). Magnetic field profile as a function of the z axis (direction along the beam pipe) is plotted on the right-hand side. The red dashed lines correspond to the location of the tracking sub-detectors (right). Figure taken from [17].

1.3.2 Particle identification

Particle identification (PID) is a vital step in any physics analysis. For instance, the ability to significantly reduce the background often relies on the correct separation of kaons and protons from pions. The LHCb PID system is complex and comprises of two ring-imaging Cherenkov detectors (RICH), a series of muon chambers, and a calorimeter system (ECAL and HCAL), see Figure 0.0.2.

The combined information from these sub-detectors allows distinguishing between various types of charged and neutral particles. Identification of the charged particles can be enhanced using tracking information. Calorimeters, apart from measuring the energy, can also provide information regarding the particle type for electrons, photons and hadrons. The muon system provides identification of muons with a high purity that is necessary for all CP-sensitive decay processes. Ability to distinguish pions and kaons/protons with high efficiency is done using the RICH detectors and is critical for the purely hadronic decays. It is important to note that for LHCb experiment the performance of the PID reconstruction is measured using data-driven techniques since the simulation poorly reproduces the PID variables. At the moment, two mutually exclusive approaches are used where simple low-level variables measured by respective sub-detectors are combined to provide more powerful selection variables. The first method, called $DLL_{X,\pi}$, relies on a linear combination of likelihood information produced by each detector which is then added to form the combined likelihood ratio (or difference of log-likelihoods) between particle X (where $X \in K, p, \mu, e$) and pion hypothesis. So, the general idea of applying the $DLL_{X,\pi}$ method is to evaluate log-likelihood difference between a given particle X and a pion hypothesis as follow:

$$\Delta log\mathcal{L}_{X,\pi} = log\mathcal{L}_X - log\mathcal{L}_\pi \tag{1.4}$$

Where: $\mathcal{L}_X$ and $\mathcal{L}_\pi$ represents log-likelihoods for particle X and a π. The classification is performed by simply comparing the $DLL_{X,\pi}$ to a tuned threshold. When the $DLL_{X,\pi}$ is greater than this threshold, it means that the particle is likely not to be a pion.

To enhance the PID performance the LHCb collaboration decided to apply also the multivariate analysis [59]. The model, called *ProbNNX*, that was proposed and deployed is a fully-connected neural network (that kind of model is described in section 3.3.4) with one hidden-layer implemented using the TMVA package [83]. In order to produce the *ProbNNX* output the tracking information, such as track momentum and pseudo-rapidity, is also used. In the end the output of the model is employed to distinguish a given particle specie X from any other (i.e., a multi class classification problem). Additional models are trained to identify neutral particles as well. In particular two models *isNotE* and *isNotH* are trained to separate photons from electrons and hadrons respectively. These baseline models proved that the application of Machine Learning could significantly improve the performance of Particle identification. Figure 1.3.10 shows performance comparisons using ROC curves [5] determined for respective models.

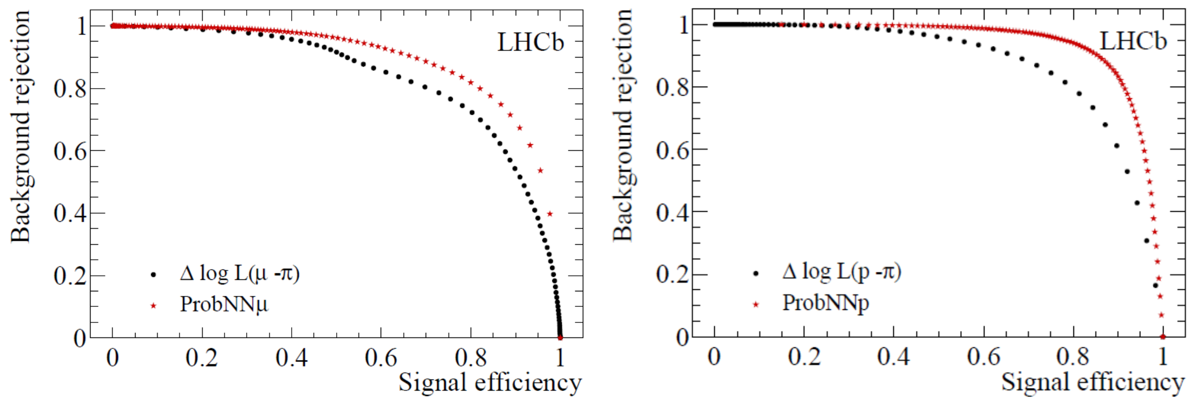


Figure 1.3.10: Background misidentification rates versus muon (left) and proton (right) identification efficiency. The variables $\Delta\mathcal{L}(X\pi)$ (black) and ProbNN (red), are compared for $5 - 10$ *GeVc* muons and $5 - 50$ *GeVc* protons, using data sidebands for backgrounds and simulated samples for the signal. The data sample used corresponds to 2012 sample collected at center-of-mass energy 8 GeV. Figure adapted from [59]

The overall Particle Identification performance can be summarize using the following figures of merit:

- Electrons: 90% identification efficiency with about 5% electron to hadron missidentification probability.

- Kaons: identification efficiency averaged over the momentum range of $2 - 100$ GeV/c is 95% with a nearly 5% pion to kaon missidentification rate.

[5]For a formal definition of ROC curve see section 3.2.0.2

- Muon: 97% identification efficiency with pion to muon missidentification rate in between 1 and 3%.

The remainder of this section is dedicated to present each component of the Particle Identification system.

1.3.2.1 RICH detector

The most critical component of the Particle Identification system is the Ring Imaging Cherenkov detector (RICH). LHCb has two of these detectors installed [13]. The first one, called RICH1, is placed just before TT detector and the second one (RICH2) after T stations, see 0.0.2. These detectors were designed to identify charged hadronic particles over a large momentum range of $2 - 100\ GeV/c$. To accomplish this task, RICH1 was filled C_4F_{10} gas radiator, which provides sensitivity for particles with momentum in range $2 - 10\ GeV/c$ (low-momentum particles) including those that are swept out of the detector acceptance by the magnetic field. In contrast, RICH2 is filled with CF_4, which can be used to cover the momentum range of $15 - 100\ GeV/c$. The geometry of both RICH1 and RICH2 detectors are presented in Figures 1.3.11 and 1.3.12, respectively.

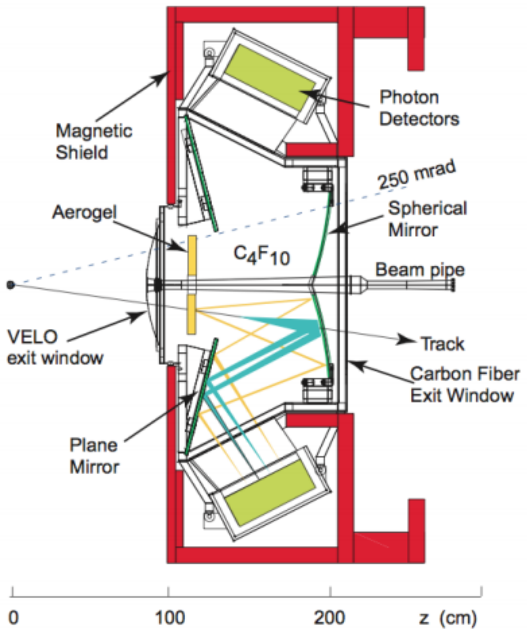
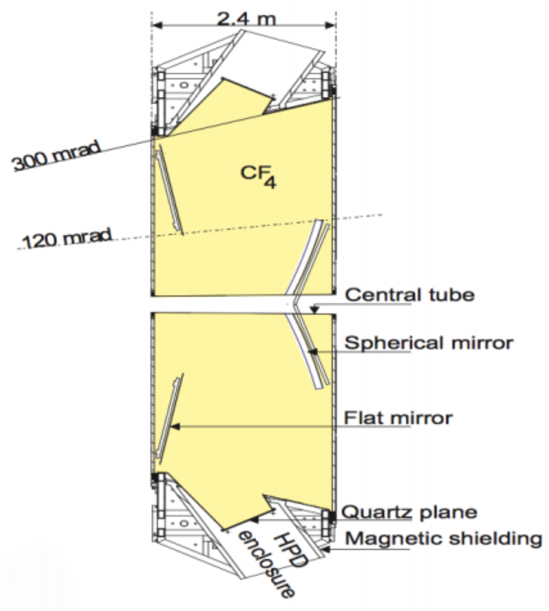


Figure 1.3.11: Geometry of the low momentum RICH detector (left), photo of the RICH1 detector (right). Figures taken from [17]

The fundamental principle of operation of the RICH detector is to measure Cherenkov radiation emitted when a charged particle traverses its active volume. Cherenkov radiation is always emitted when



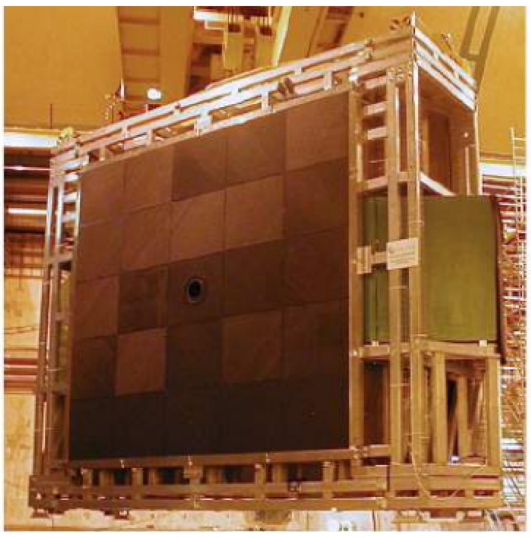


Figure 1.3.12: Geometry of the high momentum RICH detector (left), photo of the RICH2 detector (right). Figures taken from [17]

a charged particle moves through a medium at the velocity higher than the speed of light at this medium. The angle at which the Cherenkov photons are emitted (θ_c) depends directly on the particle velocity, and it is expressed by

$$cos\theta_c = \frac{1}{n\beta} \qquad (1.5)$$

Where: β is the velocity of the particle divided by the speed of light,
and $n = \frac{c}{v_{medium}}$ is the refractive index.

Once emitted, Cherenkov radiation is reflected via a combination of spherical and flat mirrors to hybrid photon detectors (HPD). The HPD has a photocathode that emits electrons when excited by the Cherenkov radiation. Electrons are accelerated by a potential of about 20kV towards a silicon detector, which allows identifying the location of the hit. The performance, quantified using the identification efficiency is shown in Figure 1.3.13.

1.3.2.2 Muon Stations

The proper muon identification is an essential requirement because muons are the final decay states of some of the most important heavy flavor decays such as $B_s^0 \to J/\psi\ \mu^+\mu^-$, $B_s^0 \to K^{*0}\ \mu^+\mu^-$ and they can be used as an initial flavor tag for measurements of B^0 and B_s^0 oscillations.

The muon system provides muon identification as a log-likelihood variable, which depends on the track momentum and the number of hits detected in muon stations and how close the hits are with respect to the extrapolated track position in the muon system. It also provides such information as x, y position in the muon station, which can be used for standalone-track reconstruction and finally p_T

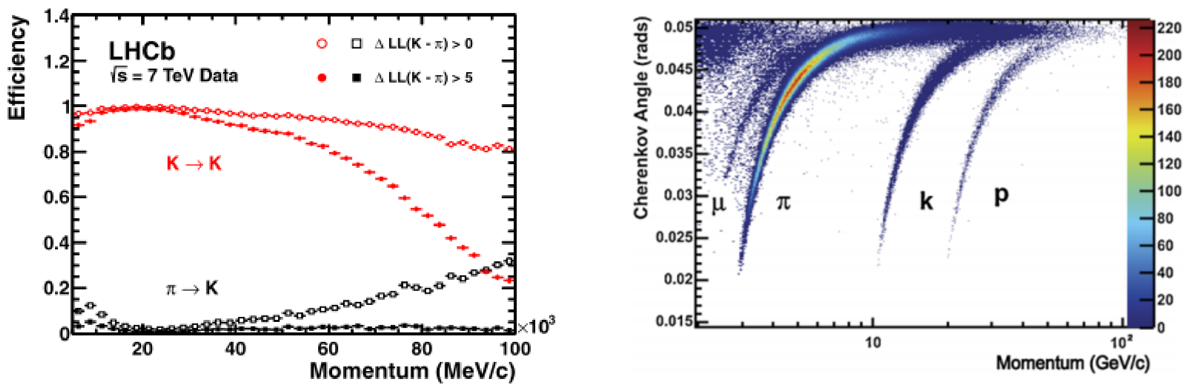


Figure 1.3.13: Kaon identification efficiency and pion misidentification rate measured using simulated events as a function of track momentum. Two different $\Delta log\mathcal{L}(K\pi)$ requirements (often called tight and loose) have been imposed on the samples (left), reconstructed Cherenkov angle as a function of track momentum in the C_4F_{10} radiator (right). Figures taken from [13]

information used by the L0 trigger system. More details refer to section 1.3.3. Because the information from the muon stations is used in the hardware part of the LHCb trigger system, it is read out at the frequency of $40Mhz$ (which is the LHC machine clock).

Muon stations are located the farthest from the interaction point. The placement of these detectors is dictated by the fact that muons interact very weakly with the material, have a high masses ($105\, MeV/c^2$), and long lifetime ($2.2 \cdot 10^{-6}s$) thus muons travels much farther than any other charged particles.

The muon system is composed of five stations, one situated just before the calorimeters and four downstream from them. Each of the stations is composed of two types of detectors. The first one is a multiwire proportional chamber located far from the beam pipe, and triple gas electron multiplier (GEM) detectors [6] placed in central quadrants close to the beam. Those detector use gas mixture consisting of Ar, CO_2 and CF_4. A cartoon of the muon station is shown in Figure 1.3.14. The efficiency of the muon identification is, on average, above 98% with pion and kaon misidentification rate below 1%, which is shown in Figure 1.3.15.

1.3.2.3 Calorimeters

The calorimeter system performs several functions. It is responsible for providing fast information for the hardware trigger level and allows identification of electrons, photons, and hadrons, jointly with a measurement of their energies and transverse positions.

The calorimeter system is designed to measure the energy of an interacting particle. This is achieved via measuring the energy of secondary electromagnetic and hadronic showers, which are created when a particle travels through the very dense absorber material (i.e., the material with a very low radiation length). The signal is formed using scintillator detectors (see text below). The measured energy is the total energy of all showers absorbed in the active materials, thus corresponds to the initial energy of the

[6] GEM detector is used due to the higher particle flux

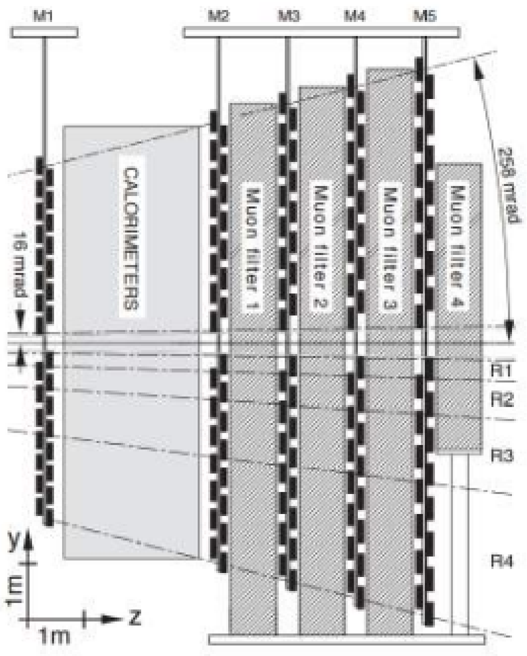


Figure 1.3.14: Side view of the muon detector (left) and a photo of M5 station. Figures taken from [17].

initial particle.

The calorimeter system consists of an Electromagnetic Calorimeter (ECAL) and Hadron Calorimeter. Both are placed between the first and second muon stations, see 0.0.2. ECAL subdetector is dedicated to identifying photons and electrons. It is equipped with two additional detectors, placed in front of it, a PreShower detector (PS) and a Scintillator Pad Detector (SPD), the layout and granularity of both are presented in Figure 1.3.16.

PS and the SPD are used by the low-level trigger to distinguish electrons from photons and pions. The information about the number of tracks per event obtained by SPD is also used by the trigger to drop events that are too busy. The ECAL is made of 2 mm lead plates followed by a 4 mm scintillator pad (shashlik like layout). Its granularity depends on the distance from the beam, see Figure 1.3.16. The energy resolution of the ECAL detector can be expressed:

$$\frac{\sigma_E}{E} = \frac{10\%}{\sqrt{E/GeV}} \oplus 1\% \tag{1.6}$$

where: $\oplus$ denote addition in quadrature which can be formulated as:

$$\Delta a \oplus \Delta b = \sqrt{\Delta a^2 + \Delta b^2} \tag{1.7}$$

The HCAL has an alternating structure of iron and scintillator tiles. The scintillator tiles are 4 mm tick and the iron ones are 16 mm. The HCAL energy resolution, obtained from the testbeam data can

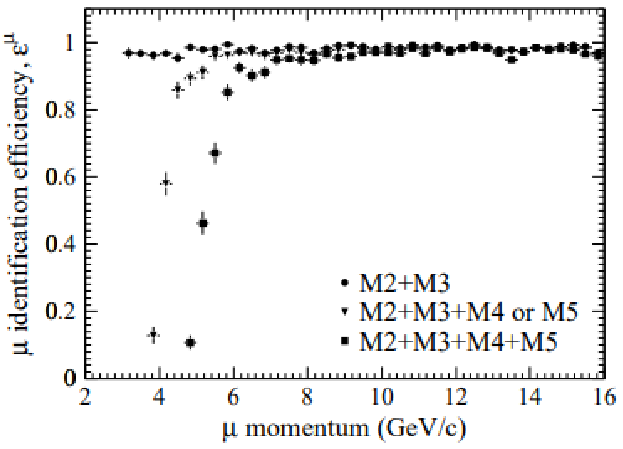


Figure 1.3.15: Muon identification efficiency as a function of momentum, for different requirements on the number of hits. Figure taken from [22].

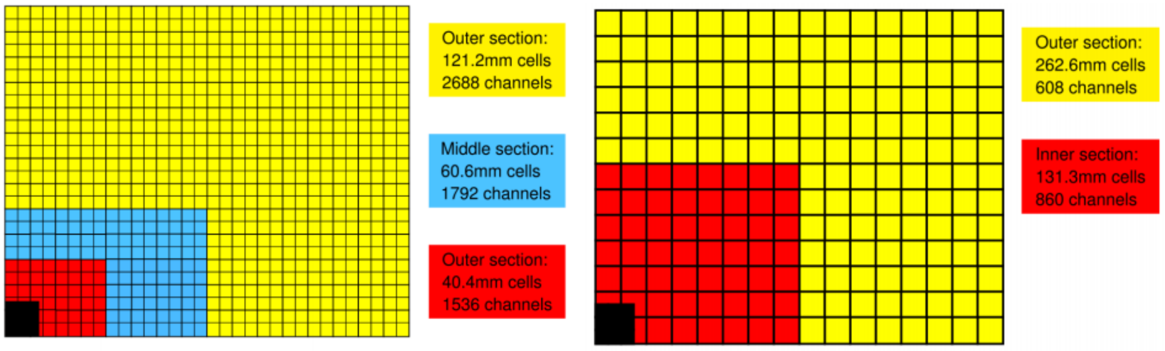


Figure 1.3.16: Granularity for the different detector regions of the SPD, PS, and ECAL (left) and of the HCAL (right). Figure taken from [17].

be expressed as:

$$\frac{\sigma_E}{E} = \frac{(69 \pm 5)\%}{\sqrt{E}} \oplus (9 \pm 2)\% \tag{1.8}$$

1.3.3 LHCb trigger

LHCb trigger is an example of the real-time system dedicated to compressing the input data stream. The raw data volume is far beyond the limit of the present storage technology, thus the necessity of employing such a system. The fundamental idea behind the large detector's trigger is to work out a decision whether a given event is interesting, from the point of view of the physics programme, or not. The LHCb trigger was designed to reduce the data rate from the initial bunch crossing rate of 40 MHz (i.e., one collision event each 25 ns) to about 12.5kHz of fully reconstructed events to be recorded on tapes. The data rate reduction is achieved by making a fast decision based on approximate measurement of particle transverse momentum and energy, muon identification, track displacement, and topological properties, which allows selecting some specific decays.

The LHCb trigger is built as a two-stage system. The first one, called Level-0 trigger (or L0 for brevity), is implemented as a hardware layer with the fixed response time of 6 *mus*. The processing power is provided by FPGA [7] chips. L0 trigger uses the information from calorimeters and muon stations to reduce the bunch-crossing rate to 1.1 MHz, which is the maximum input rate of the front-end ASICs used by other sub-systems. This partial information is then combined and process by dedicated electronics boards that give a final decision to process a given event further or drop it.

The L0 calorimeter trigger leverage the information from ECAL, HCAL, PS, and SPD detectors. Its decision is mostly based on transverse energy deposited in a cluster of 2×2 cells (the cells are presented in Figure 1.3.16) of the same size. The transverse energy, which is rather interesting quantity, is defined as:

$$E_T = \sum_{i=1}^{4} E_i \sin \theta_i \tag{1.9}$$

Where E_i is the energy deposited in the $i - th$ cell, and θ_i is the angle between the beam axis and the direction of the particle's flight path. This quantity is combined with the information on the number of hits in the PS and SPD to distinguish between hadron, photon and electron candidates.

Events accepted by the L0 trigger are sent to the Event Filter Farm, a computing cluster located at the LHCb pit, that consisted of approximately 29 000 and 50 000 CPU cores during Run 1 and Run 2 respectively. This computing farm is responsible for running High Level Trigger (HLT) application instances. The HLT software is written in C++ and consists of selection algorithms designed to identify specific decay processes, for instance, *b* or *c* hadron decays. The trigger strategy had changed over the course of years when LHCb detector collected the data. Figure 1.3.17 presents the triggering scheme for both Run 1 and Run 2.

Implementing the second stage of the LHCb trigger as a full software application has a great advantage of the flexibility that plays a paramount role in adapting to changing data taking conditions. On the other hand, it also requires constant monitoring of the trigger configuration, which is a highly non-trivial task. The machine provided beams that were tuned in the way that the average number of proton-proton interactions per one beam crossing was 1.6. The data taking periods, when protons were circulating in the machine, were called fills and they were, in turn, divided in runs. Each run could possibly be configured individually, taking into account differences in the Data Acquisition system, calibration and alignment conditions etc.

During the down period between Run 1 and Run 2 (called the Long Shutdown 1) a considerable amount of work has been done to optimise and improve the HLT software. This led to splitting the trigger into two logically exclusive parts called HLT1 and HLT2 (see text below for details). It allowed evaluating the online alignment and calibration, while the events were stored in the disk buffer. The online calibration procedures were vital for achieving high-quality reconstruction in HLT that is com-

[7] FPGA stands for Field Programmable Gate Arrays, which are devices based on a matrix of configurable logic blocks. Their design has a benefit compared to a general-purpose CPU that allows massive parallelism since FPGA programable blocks can work independently and simultaneously as streaming processors.

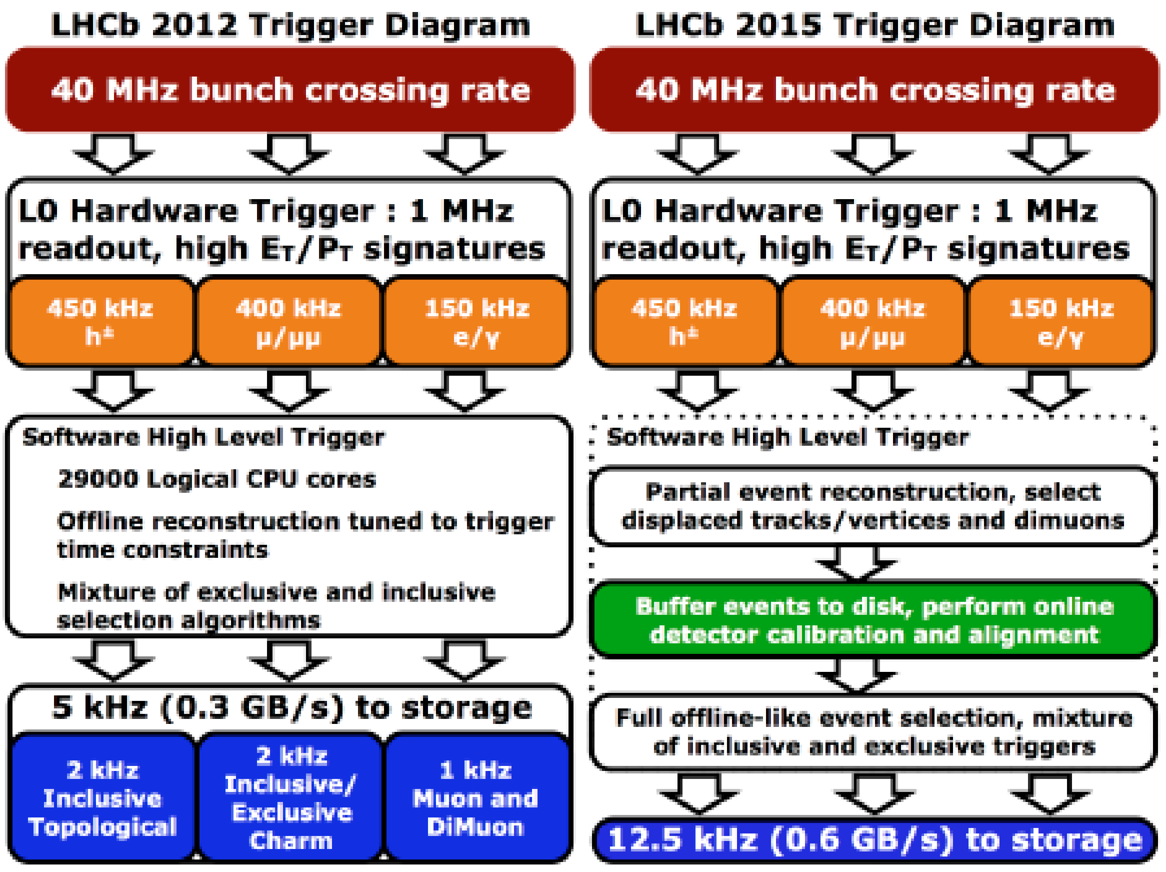

LHCb 2012 Trigger Diagram
40 MHz bunch crossing rate
L0 Hardware Trigger : 1 MHz readout, high ET/PT signatures
450 kHz h±
400 kHz μ/μμ
150 kHz e/γ
Software High Level Trigger
29000 Logical CPU cores
Offline reconstruction tuned to trigger time constraints
Mixture of exclusive and inclusive selection algorithms
5 kHz (0.3 GB/s) to storage
2 kHz Inclusive Topological
2 kHz Inclusive/ Exclusive Charm
1 kHz Muon and DiMuon
LHCb 2015 Trigger Diagram
40 MHz bunch crossing rate
L0 Hardware Trigger : 1 MHz readout, high ET/PT signatures
450 kHz h±
400 kHz μ/μμ
150 kHz e/γ
Software High Level Trigger
Partial event reconstruction, select displaced tracks/vertices and dimuons
Buffer events to disk, perform online detector calibration and alignment
Full offline-like event selection, mixture of inclusive and exclusive triggers
12.5 kHz (0.6 GB/s) to storage

Figure 1.3.17: LHCb trigger data flow during Run 1 (left) and Run 2 (right). Each graph illustrate a high-level trigger architecture and a typical event acceptance rate after each stage. Figure taken from [47].

parable to the offline one.

Within HLT1 the full detector information is used. The reconstruction process starts with the vertex detector. The track candidates are selected based on the probability that particular track originates from heavy flavor decay, which is achieved by determining their impact parameter. Selected tracks are then associated with track segments in the tracking stations, which allows estimation of the transverse momentum of the corresponding charged particle (so called forward tracking algorithm). This information, together with track's χ^2 and impact parameter χ^2_{IP}, is used to select interesting events. During Run 2, a small portion of data selected by HLT1 was used to calibration and alignment of the detectors. This process, which takes a few minutes, is performed to reduce the probability of misalignment on the tracker. Any misalignment would impact the momentum resolution affecting the quality of reconstruction in HLT2. No particle-identification is available at this stage (apart from a coarse muon identification). The final output rate of HLT1 is approximately 110 kHz.

The final selections are performed by HLT2 trigger using the full particle identification variables. HLT2 implements two types of trigger lines that select exclusive and inclusive processes, respectively . Exclusive algorithms are used to select specific decays. For instance, they required all decay products to be within the detector acceptance and reconstructed. Inclusive trigger selections, also called topological lines, trigger on partially-reconstructed b hadrons decays. Those lines are designed to detect all b hadrons decays with a displaced secondary vertex, and at least two charged particles in the final state. The output bandwidth is divided into three streams and undergoes constant, careful monitoring. About 40% of the total output rate is assigned to inclusive topological selections, another 40% is reserved to exclusive lines targeting the c-hadron decays and the rest is given to other exclusive processes. The quality of reconstruction in the HLT2 during Run 2 allowed LHCb to implement so-called ”Turbo” lines that return fully reconstructed analysis-ready data. Additionally, those lines allow saving space by discarding the raw data, keeping only the relevant information describing reconstructed events.

The trigger efficiency is estimated using the so-called TIS-TOS (Triggered Independent of Signal - Triggered on Signal) method, described in detail in [137]. TOS events are those where daughter particles that form a particular decay candidate passed the trigger selection criteria. In the case where other particles in the event passed the criteria, a given decay candidate is called TIS. Both categories are not exclusive. When estimating the trigger efficiency, detailed knowledge of which class a given event belongs to is very important. It should be stressed that the reconstruction algorithm that is the subject of this thesis (described in chapter 4) was executed as a part of the HLT2.

1.3.4 LHCb software

In order to produce the Monte Carlo simulated samples (see Figure 1.3.18) and analyze data collected by the LHCb spectrometer a dedicated software framework has been designed and implemented [54]. Most of the applications were written in C++, and they are based on two frameworks ROOT [34] and Gaudi [23]. The list below contains a brief description of selected packages:

- **Gauss** [25] was designed to generate the initial particles and simulates their transport through the LHCb detector. The Gauss application consists of two major independent processing phases.

The first one is a generation of the proton-proton interactions (primary vertices), at LHC energies, that result, in turn, in producing primary particles. This generation process is handled by PYTHIA [129], a general-purpose event generator, whist the decay and time evolution of the produced particles is delegated to EventGen [94]. The second phase of Gauss application performs detector response simulation via a customized Geant4 [14] based module.

- **Boole** [54] performs a final stage of the LHCb detector simulation. It applies the detector response to hits previously generated by the Gauss. This step, called digitization, includes simulation of the detector and read-out electronics response, together with L0 trigger hardware information. The output format of Boole corresponds exactly to the data coming from the real detector.

- **Brunel** [54] this package is responsible for the whole process of data reconstruction, which consist of retrieving all recorded hits in a detector, doing the pattern recognition to identify trajectories, finding primary vertices of proton-proton interactions, and assigning PID likelihoods. Brunel can process both simulated and the real collected by detector data in a completely agnostic way. The outcome of the Brunel consists of the high-level reconstructed objects (e.g tracks and vertices described in Chapter 4) that are saved in a Data Summary Tape (DST) format.

- **DaVinci** [54] was designed to process the Brunel output and, based on it, reconstruct decays of interest and apply selection criteria to reduce the background. The outcome of this step is a dataset containing decay candidates for the user-specific decay topologies, which are used as a starting point for further physics analyses.

1.4 LHCb upgrade

This section is divided into three parts. The first one is dedicated to present the general concepts of why LHCb collaboration decided to upgrade its detector. The second one describes the scope of the Upgrade by discussing which elements will be replaced. The final subsection focuses on the Upstream Tracker detector. It provides a very detailed description of this detector since the author was personally involved in its development.

1.4.1 Motivation

The data collected during both Run 1 and Run 2 allowed to perform and report several World best measurements of rare decays of b and c hadrons, which were used to set new limits on models describing New Physics. However, many measurements are still limited by the statistical uncertainties. Continuing data collection at the current rate would not allow us to decrease them to the level compatible with the theoretical predictions. In order to increase annual data yields, the LHCb detector must undergo a major upgrade during the Long Shutdown 2 (expected to be finished in early 2022). The changes will allow the detector to operate at the increased luminosity of $20 \times 10^{33} cm^{-2} s^{-1}$, which is five times higher

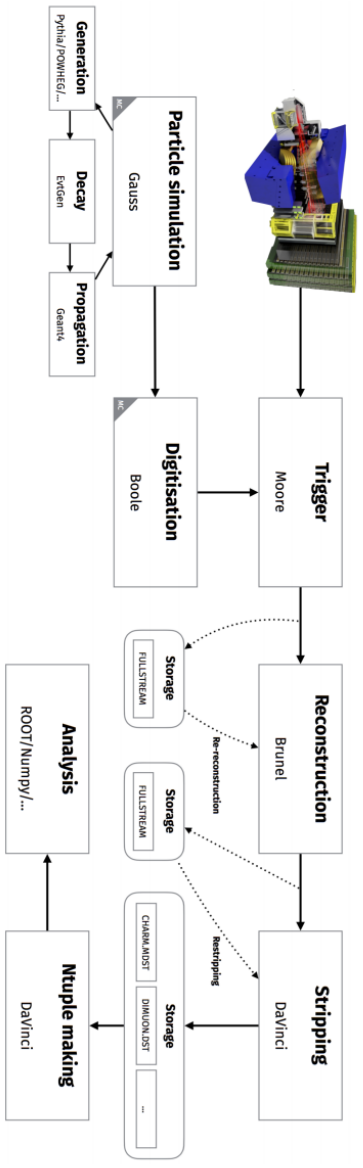


Figure 1.3.18: LHCb event simulation flowchart. Each rectangle with a sharp corners represents a separated LHCb package described in text. Figure taken from [136].

than the previous one. The new detector is expected to read data (full detector information) at the bunch crossing rate of 40 MHz, which allows collecting about 10 fb^{-1} per year while during both Run 1 and Run 2, LHCb collected approximately 8 fb^{-1}. Figure 1.4.1 present the luminosity plan. This figure also presents the prediction of the longer-term future of the LHCb experiment, which is out of the scope of this thesis.

1.4.2 General aspects of the LHCb Upgrade

One of the main limitations that drive the idea of the LHCb Upgrade was the limitation of the hardware L0 trigger and the readout electronics (which resulted in limited event input rate to HLT trigger). This limitation comes from the specific of the current read-out system, see section 1.3.1.1. To overcome this limitation, all tracking detectors and their read-out systems have to be replaced to be capable of reading

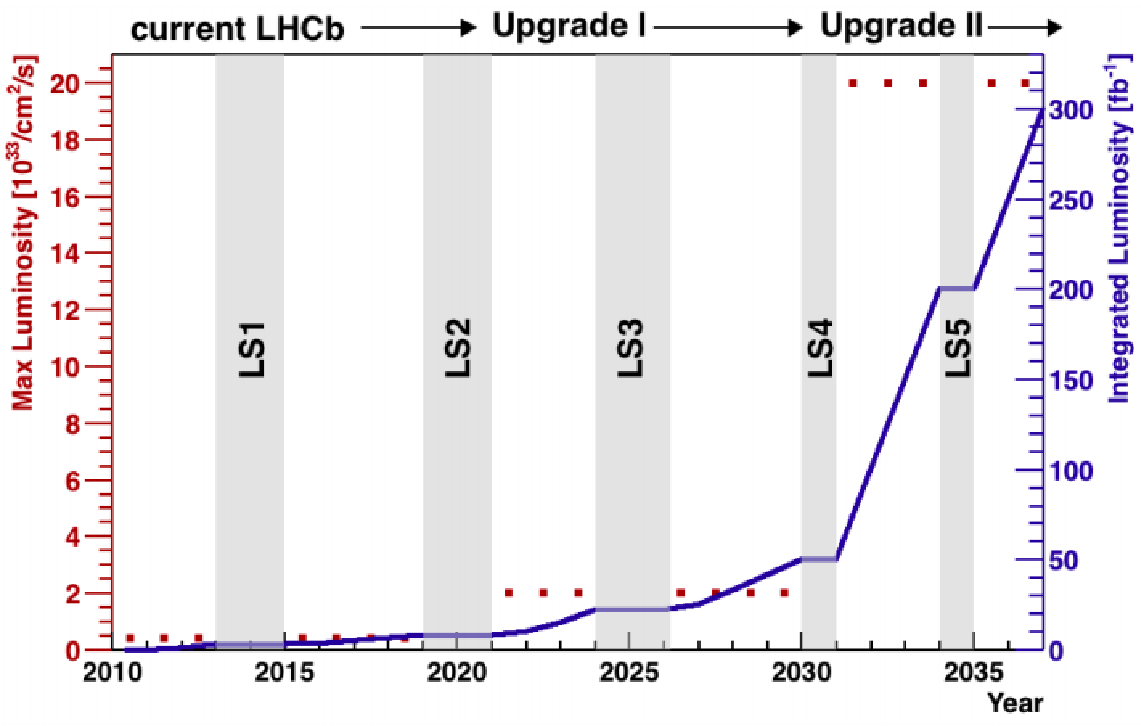


Figure 1.4.1: Actual and predicted integrated luminosity from 2010 to 2037 for the LHCb experiment. Red dots represent the value of measured or predicted instantaneous luminosity. Solid blue line represents the value of measured or predicted integrated luminosity.

out the full detector information at the rate of 40 *MHz*. New read-out electronics will allow removing the L0 trigger, keeping only the software flexible part. Figure 1.4.2 presents the new layout of the LHCb detector. When comparing this layout with the previous one, see Figure 0.0.2, it is clearly visible that the overall structure of the spectrometer stays as it was. All components of the tracking system (vertex detector, TT and T stations) will be replaced. The Cherenkov detectors will be heavily modified, and both the PS and SPD detectors will be removed from the spectrometer. Only the detectors that were previously contributing to the L0 trigger will not have major interventions.

1.4.3 Upgraded Velo

The upgraded Velo detector will be placed closer to the beam, its active area will reach the distance of $5.1mm$ from the beam axis, and it will have a finer granularity thanks to changing from micro-strip to pixel technology. The new Velo will operate at much higher particle flux due to increased luminosity, which causes an increase in the average number of visible proton-proton collisions from 1.6 to 5.2 (i.e., on average 5.2 primary vertices will be present at each beams crossing). Thus, the Velo group decided to use pixel sensors to reduce occupancy. The upgraded Velo detector will consist of 41 million $55 \times 55\mu m$ pixels, which will be read out by the custom-build VeloPix front-end ASIC, at the 40 MHz rate. The cooling is provided by evaporative CO_2 system. It will employ an innovative micro-channels embedded in the structure of the support silicon modules. A layout of the upgraded Velo module is shown in Figure 1.4.3. Both the expected performance of the track reconstruction efficiency and impact parameter

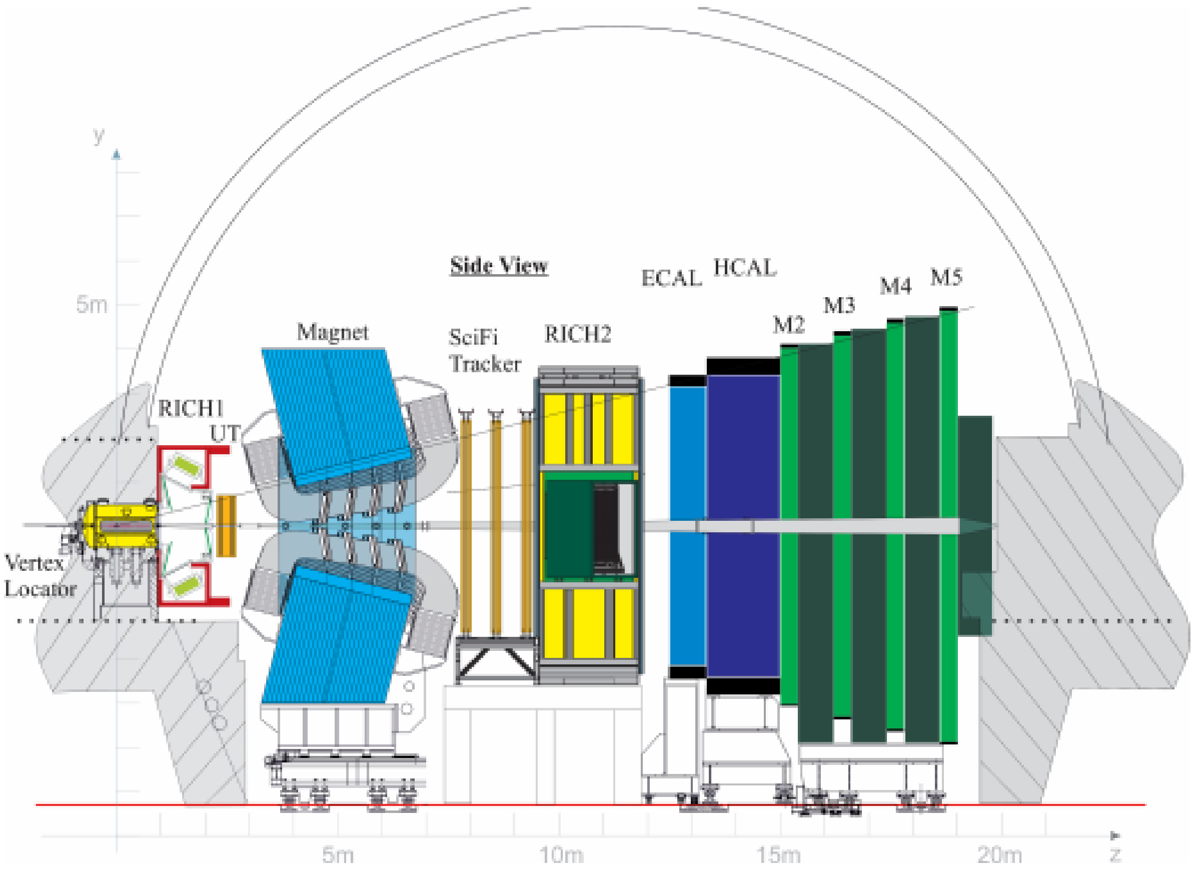


Figure 1.4.2: Layout of the upgraded LHCb detector. Figure taken from [49]

as a function of the inverse of the transverse momentum are shown in Figure 1.4.4.

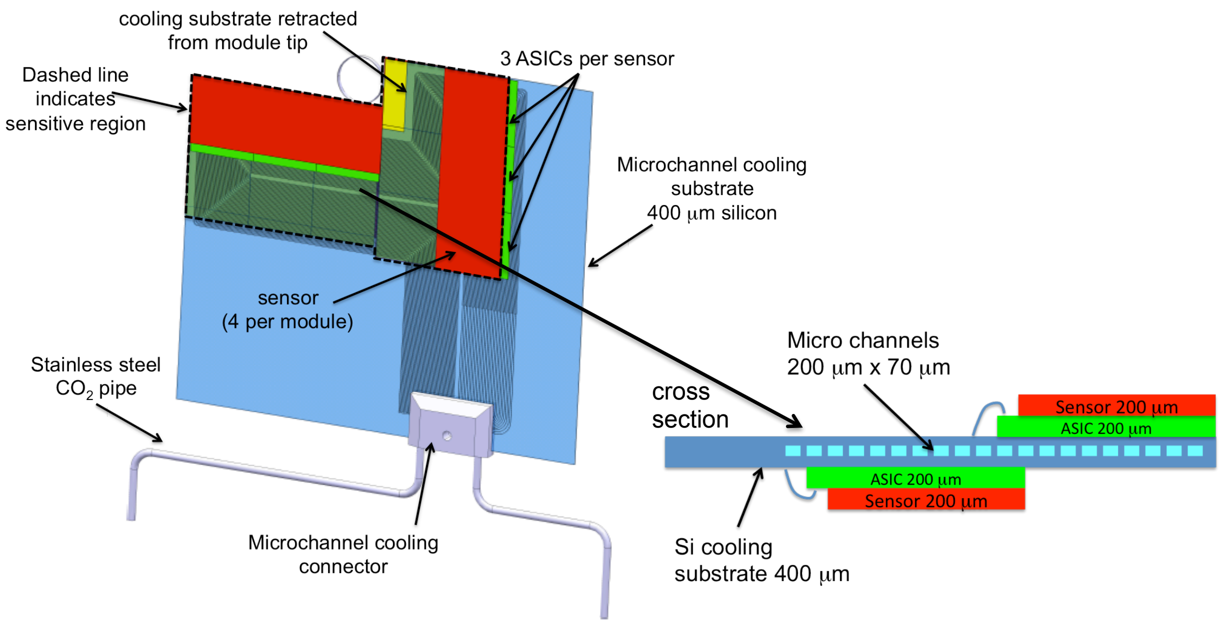

cooling substrate retracted from module tip
Dashed line indicates sensitive region
3 ASICs per sensor
Microchannel cooling substrate 400 μm silicon
sensor (4 per module)
Stainless steel CO$_2$ pipe
Micro channels 200 μm x 70 μm
cross section
Sensor 200 μm
ASIC 200 μm
ASIC 200 μm
Sensor 200 μm
Si cooling substrate 400 μm
Microchannel cooling connector

Figure 1.4.3: Layout of a module of the upgraded Velo detector, see detailed description in text. Figure taken from [48]

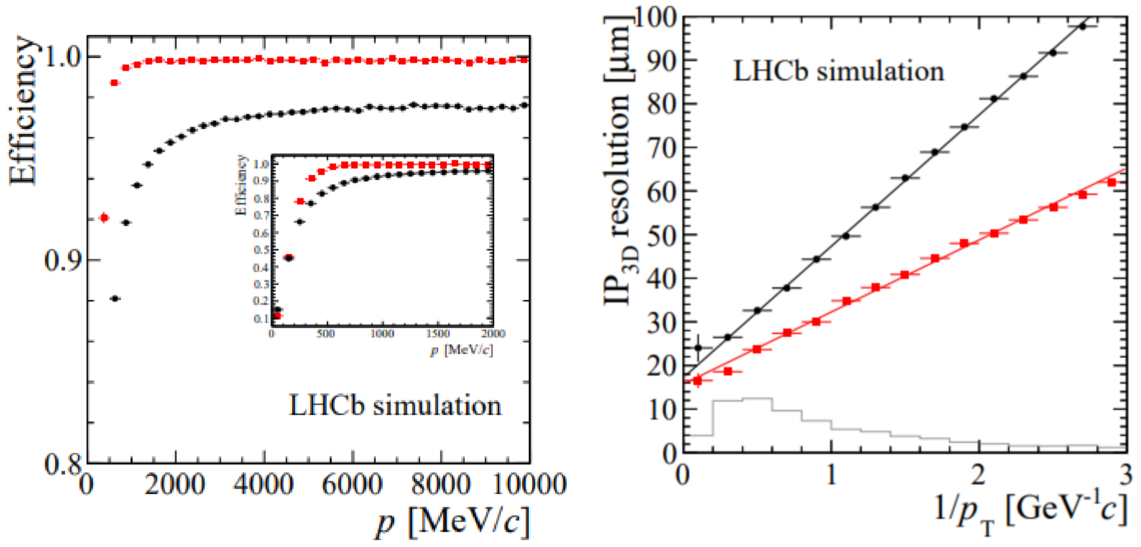


Figure 1.4.4: The left figure shows the reconstruction efficiency evaluated for particles which are reconstructible as Velo tracks as a function of momentum. The right plot shows the 3D resolution of the IP, the light gray histogram shows the relative population of b hadron daughter tracks in each $1/p_t$ bin. Figures taken from [48].

1.4.4 Scintillating Fibre Tracker

Scintillating Fibre Tracker (FT) [49] [8] was designed to replace the T tracking stations. Two factors drove this decision. Extensive simulations studies showed that the upgraded condition would be too harsh for a straw gas detector (like the previous OT gaseous system). The foreseen occupancy would be too high to provide reliable input to the tracking pattern recognition algorithm. Moreover, the readout electronics of both OT and IT detectors would not be capable of working as a part of a new data acquisition system. FT tracker covers the full LHCb detector acceptance downstream to the magnet and, by design, guarantees a spatial resolution close to $80\mu m$. This detector will consist of three stations, each of them composed of four detection planes organized, similar to IT, in a (X, U, V, X) orientation, see Figure 1.4.5. The module will consist of scintillating fibres with a radius of $125\mu m$, and a length of 2.5 m, which will be read out by Silicon Photo-Multipliers (SiPMs), located either at the top or bottom of the detector. The SiPMs are kept in a cold box, at $-40°C$, to reduce the dark count rate [9], and each of them consists of nearly 128 individual pixels.

The FT detection mechanism is based on measuring the photons emitted when a particle traverses the detector's active area. Those photons are propagated through the fibres and finally reach the silicon pixels located at the end of the fibres. A signal proportional to the number of photons detected within a given SiPM detector is used to determine the position of the particle, shown in Figure 1.4.6. Each pixel

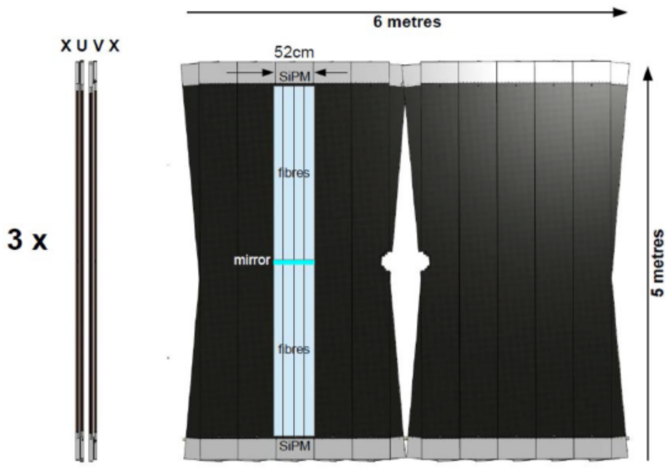


Figure 1.4.5: Layout of one tracking station of the SciFi detector. Figure taken from [49]

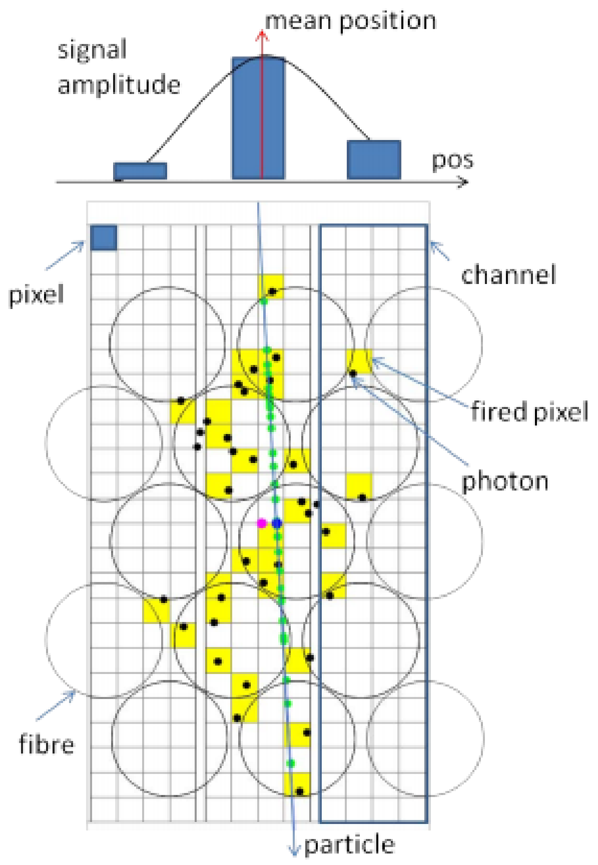


Figure 1.4.6: Simplified visualization of the detection mechanism of the FT. The squares show the pixels located at the end of the fibres, and the circles indicate the cross-section of the scintillating fibres. Note that the fibres are not aligned to the detector channels, and the photons can arrive at the detector outside the fibre area. Figure taken from [49]

can detect one photon at a time.

[8]The previous name of this sub-detector was SciFi and referred to the speculative fiction genre since many people did not believe that the construction of such a detector is feasible.

[9]The dark count rate, is the count rate that is measured in the absence of photons, is caused by thermally generated electron-hole pairs.

1.5 Upstream Tacker

The Upstream Tracker (UT) is a detector that was designed to replace TT tracker. Its structure and sensor technology is very similar to its predecessor. Thus it will be composed of four planes of silicon-strip detectors divided into two stations, as it is schematically shown in Figure 1.5.1. Similarly to TT, the detector plates will be arranged in a (X, U, V, X) orientation, with the second and third planes rotated at a stereo angle of $\pm 5^{\circ}$ with respect to the y-axis.

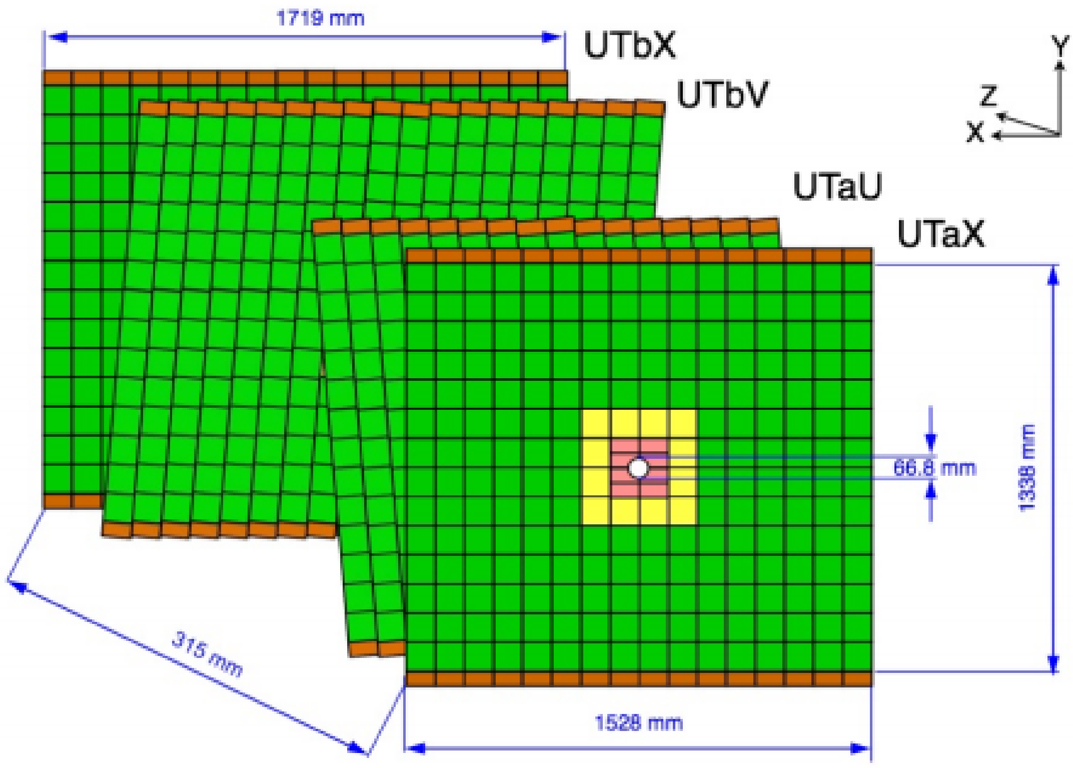


Figure 1.5.1: Layout of the four UT detector planes (looking downstream of the interaction point). Outermost, intermediate, and innermost sensors are shown in green, yellow, and reddish, respectively. Figure taken from [49]

The expected trigger efficiency enhancement with respect to the TT will originate from improved acceptance coverage at small polar angles and finer granularity. This will be achieved by designing the innermost sensor to have a circular cut-off around the beampipe, as shown in Figure 1.5.1 (reddish sensors). The UT sensors will have improved radiation hardness, which is critical since the UT sensors have to withstand an integrated luminosity of 50 fb^{-1}, which is a factor of 5 more than TT. Figure 1.5.2 presents the expected irradiation dose, estimated by a FLUKA simulation [37], after 50 fb^{-1} as a function of y coordinate, obtained for a detector station slice at $x = 0$. Those expected irradiation doses drove the decision on the sensor's technologies. The sensors that are situated closer to the beam will be n^{+}-on-p technology, which is more radiation hard than p^{+}-on-n.

Four different sensors geometries, denoted type-A, B, C, and D, are used in the UT detector depend-

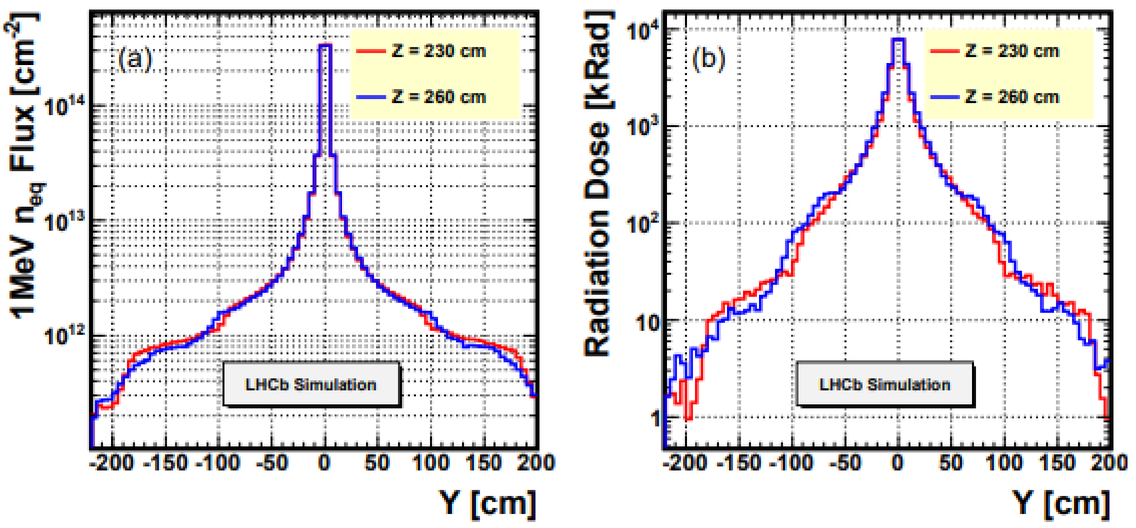


Figure 1.5.2: Expected fluence profile (left) and radiation dose profile (right) as a function of y-coordinate, for a slice of the detector that is positioned at $x = 0$. This slice represent the highest fluence region throughout the UT system. Figure taken from [49]

ing on proximity to the beam pipe, see Figure 1.5.3. To mitigate the effects of irradiation and reduce leakage current, the cooling system is designed to provide a maximum operating temperature of the silicon detectors of $-5 \circ C$.

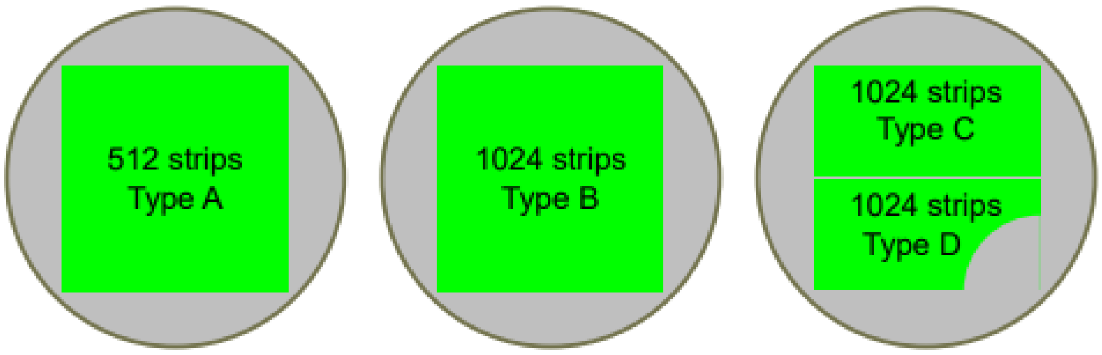


Figure 1.5.3: Sketch of the three mask designs for the UT upgrade. Sensors C and D are shorter and can be produced in a single 4-inch wafer, whereas sensors A and B require a full wafer. Figure taken from [49]

1.6 SALT ASIC

The design of the new strip readout chip, operating at bunch crossing frequency of 40 MHz, was a critical part of the LHCb Upgrade. The primary goal of this ASIC was to overcome hardware trigger limitation, which significantly limited the amount of collected data in Run 1 and Run 2, see section 1.3.3. The Silicon ASIC for LHCb Tracking (SALT) is a chip designed at AGH-UST Krakow that is capable of reading 128-channels. It was manufactured in radiation-hard TSMC CMOS[10] 130*nm* [11] technology and employed a novel architecture comprising an analogue front-end and an ultra-low power (< 0.5 mW) fast (40 MSps[12]) sampling 6-bit ADC in each channel. Thus, the chip is capable of performing quite involved data processing on-detector and significantly reducing the data volume sent to the trigger system.

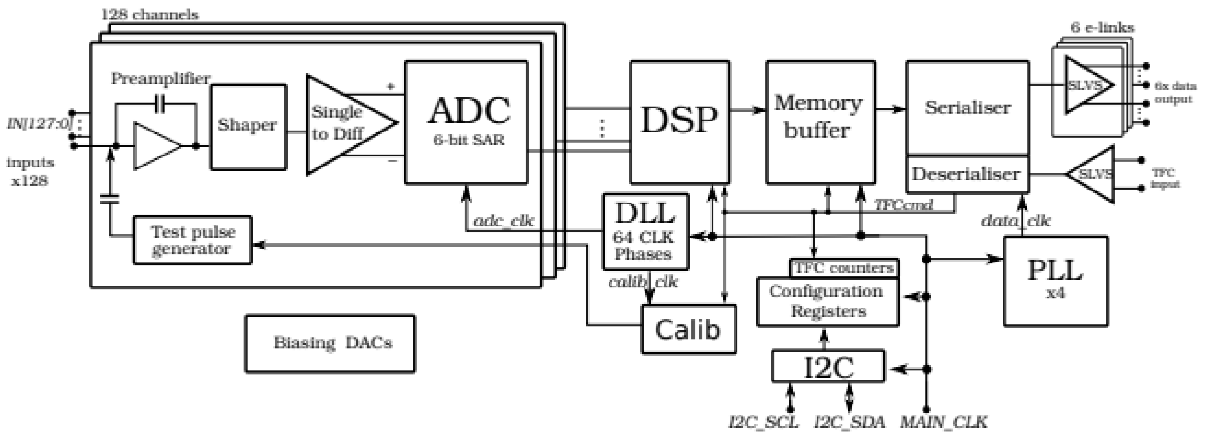


Figure 1.6.1: Block diagram of the SALT ASIC readout chip, see text for more details. Figure taken from [35].

The Salt consists of two main blocks analog and digital, see diagram 1.6.1. The analog block consists of a charge preamplifier and a fast shaper with a peaking time of about 25*ns*, and 25 ns after the peaking time of no more than 5% of the peak value. Those parameters are required to distinguish between consecutive LHC bunch crossings. The SALT was designed to operate with both types (p^+-on-n and n^+-on-p) of strip sensors with capacitance in range $5 - 20\,pF$. The shaper is followed by a single-to-differential block that converts a single-ended signalling to a differential one [13]. The last component of the analog block is a fully differential 6-bit SAR ADC, which converts the analog signal to the digital domain.

The digital ADC output is processed by a Digital Signal Processing (DSP) block, which performs the following algorithms:

- **Channel masking**. Noisy or dead channels can be masked and removed from further processing;
- **Pedestal subtraction**. This subtraction is performed in each channel independently using a different value;

[10]CMOS stands from Complementary metal–oxide–semiconductor and it's a technology that uses complementary and symmetrical pairs of p-type and n-type MOSFETs (Metal–Oxide–Semiconductor Field-Effect Transistor) for logic functions.

[11]130*nm* refers to the minimum gate length that can be manufactured using a particular technology.

[12]milion of samples per second

[13]Single-ended signaling is a method of signal transmission where one wire carries a changing voltage (signal) and the second wire is connected to a reference voltage (ground), while differential signaling sends a signal via employing two complementary voltage signals to transmit one information signal [6]

- **Mean Common Mode Subtraction**. Removal of the mean value of calculated on top of channels with signal below the certain hit threshold;
- **Zero Suppression**; Only the data registered in channels where the signal is above hit threshold are sent out for further processing in the trigger system.

Since high level emulation of these algorithms is part of the topic of this thesis, the detailed explanation of each can be found in chapter 5.

After the DSP, the data is serialized and sent out through, so-called, e-links using SLVS interface [36] operating at $320MBit/s$ data rate. Each ASIC is equipped with e-links, but only some of them are active, depending on the expected hit occupancy on the sensor. The configuration and monitoring of the SALT ASIC can be adjusted through the Inter-Integrated Circuit (I^2C) interface [111].

Not only is the Universe stranger than we think, it is stranger than we can think

Werner Heisenberg

2 Physics behind the LHCb experiment

This chapter is dedicated to providing a brief introduction to the physics of LHCb. It starts by presenting the fundamental concept of symmetries in physics, including the particular type of discrete symmetries and its consequence, then the Standard Model of particle physics is briefly described.

2.1 Symmetries in physics

Until the 20th-century, principles of symmetry played a minor role in theoretical physics. The ancient Greeks were fascinated by the symmetries of objects and believed that these should be mirrored in the structure of nature. Still, they did not manage to associate those symmetries with any deterministic law of physics. Instead, symmetry was a critical component that inspires several architects designing the most stunning and recognizable buildings, and even recently, psychologies proved that symmetrical faces are more attractive [99]. In one of the most important book of all time "Philosophiæ Naturalis Principia Mathematica" [108] Newton postulated that laws of mechanics incorporate symmetry principles, notably the principle of equivalence of inertial frames, or Galilean invariance. These symmetries implied conservation laws. However, these conservation laws were seen as consequences of the dynamical laws of nature rather than as consequences of the underlaying symmetries.

This situation had dramatically changed at the beginning of the 20th century when Emmy Noether proved her famous theorem relating continuous symmetries and conservation laws. This theorem states that as a direct consequence of the invariance of physics laws under continuous spatial transformations, such as spatial translation, spatial rotation and time translation, momentum angular momentum and energy are conserved respectively. The less intuitive case is the symmetry of the wave function under the change of its phase that leads to the conservation of the electric charge. Precisely speaking, a symmetry is a mathematical operation that leaves the physical system invariant. From the moment of this paper publication onwards, physics can be defined as the science that studies fundamental symmetries and the mechanism of the symmetries breaking.

In Quantum Mechanics, symmetries are mostly discrete and satisfied if an operator commutes with the Hamiltonian (representing the time transformations). If a discrete symmetry holds, it leads to a conserved quantum number and a selection rule for the transitions between different states. Within the framework of particle physics, there are three fundamental symmetries *P*, *C*, and *T*; those symmetries are shown in Figure 2.1.1

A parity transformation *P* can be represented by inversion of the spatial coordinates of a coordinate system and changing the helicity of the particle. Namely, the eigenfunction of the parity operator $\hat{P}$ satisfy the condition:

$$\hat{P}\,|\Upsilon,\vec{r}\rangle = |\Upsilon,-\vec{r}\rangle = p|\Upsilon,\vec{r}\rangle \tag{2.1}$$

Where p is an eigenvalue of the $\hat{P}$ operator and $|\Upsilon\rangle$ its eigenstate. A second application of the $\hat{P}$ transforms the $|\Upsilon\rangle$ to its initial state, therefore the p must be equal to ± 1, and by convention for fermions $p = 1$.

Time parity *T* reverse the time direction of a process, and the charge conjugation *C* changes the sign of the additive quantum numbers, transforming particles into antiparticles keeping their helicity. The eigenvalues of the charge conjugation operator $\hat{C}$ can be expressed in the following form:

$$\hat{C}\,|particle\rangle = |anti\,particle\rangle = c|particle\rangle \tag{2.2}$$

Where c is an eigenvalue of the $\hat{C}$ operator, and similarly to the $\hat{P}$ operator's eigenvalues can have one of ± 1 value. It is interesting to note that only particles that are their own antiparticles can be eigenstates of the charge parity operator.

The strong and electromagnetic interactions are invariant under each of *C*, *P*, and *T* symmetries. The combination of *CPT* is an exact symmetry of any interaction described by the Lorenz invariant quantum field theory. This phenomenon is described as the *CPT* theorem [123] (thus far we did not find any experimental results that would negate this fundamental theorem). It is observed that *C* and *P* are not exact symmetries and they are both broken by the weak interactions: the weak charged currents couple exclusively to left-handed fermions and right-handed anti-fermions, and hence maximally violate *C* and *P* symmetries individually. The combined *CP* operation transforms a left-handed fermion into a right-handed anti-fermion. Therefore it could be expected that the *CP* symmetry is the one that is held by the weak interactions. This statement was valid until 1964 when the group , led by James Cronin and Val Fitch, working on neutral kaons decays discovered that this is not a case, [46].

2.1.1 Group Theory

This subsection is dedicated to providing a brief introduction to Group Theory, which is a branch of mathematics that was developed to studying symmetries. This section reviews some of the properties

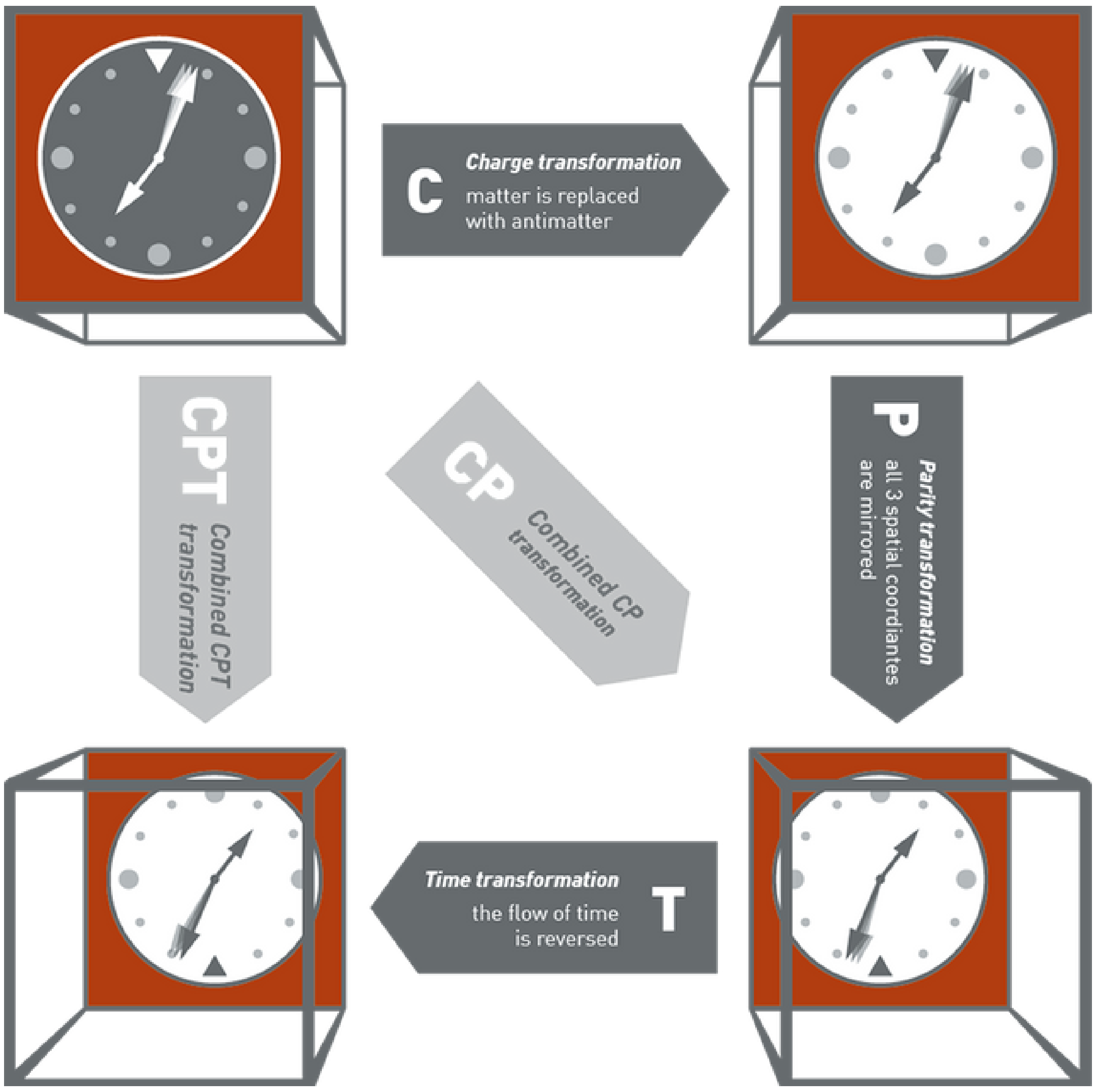

C
Charge transformation
matter is replaced with antimatter
P
Parity transformation
all 3 spatial coordiantes are mirrored
T
Time transformation
the flow of time is reversed
CP
Combined CP transformation
CPT
Combined CPT transformation

Figure 2.1.1: Symmetries in particle physics. Each arrow represents one of the C, P, T transformations or their combinations. With the discovery that the weak interactions maximally violate both charge and spatial parity symmetries, it was postulated the true symmetry between matter and antimatter is the combined charge and spatial parity transformation. As it turned out, this one is also broken by the weak interaction, but in that case, the violation effects are relatively small. The combined CPT transformation is treated as the fundamental property of our Universe and should hold for all processes. Figure taken from [1]

of them.

A Group G is an abstract set of elements, which can be finite or infinite, with defined operator $(\cdot)$ on it, which obeys:

- Closure: $\forall u, v \in G, u \cdot v \in G$
- Associativity: $\forall u, v, w \in G u \cdot (v \cdot w) = (u \cdot v) \cdot w$
- Neutral element: $\exists I \in G, u \cdot I = I \cdot u = u, \forall u \in G$
- Inverse element: $\exists u \in G, \exists u^{-1} \in G, u \cdot u^{-1} = I$

Group G is called Abelian or commutative if also the following axiom is satisfied:

- Commutativity $\forall u, v \in G u \cdot v = v \cdot u$

In elementary particle physics, the most common groups are of the type of $U(n)$, which can be represented as a collection of all unitary $n \times n$ matrices [1]. The second type of group used to construct the Standard Model is a Super Unitary group $SU(n)$. One of the fundamental properties of the unitary matrices is that their determinant is equal to 1, this plays a vital role in quantum theory that requires probability conservation. A group of matrices can represent any group, thus for every abstract element u, there is a corresponding matrix M_u, which needs to fulfil all axioms listed above.

As a concrete example let consider a group of real, orthogonal [2] $n \times n$ matrices with the determinant equal to 1, this group called $SO(n)$ and can represent all rotation is space of n dimensions. For $n = 3$, group $SO(3)$ describes rotational symmetry of our word, which according to the Noether's theorem is equivalent to the conservation of angular momentum [75].

2.2 Standard Model of elementary particles

The Standard Model (SM) is a relativistic Quantum Field Theory that describes properties of elementary particles and the electromagnetic, weak, and strong interactions. It has been experimentally validated on numerous occasions, making very accurate predictions. However, it is an incomplete theory since it does not include the theory of gravity and does not explain the dark matter and dark energy, the neutrino masses, nor the matter-antimatter asymmetry of the Universe.

This theory was built to model the particle interactions observed in nature. Using this framework, one can obtain predictions for physical phenomena by calculating transition probability from an initial state $\langle i, i'|$ to a given final state$|k, k'\rangle$. These calculations are performed using Quantum Field Theory tools such as expansions of a path integral into a power series, which can be visualized using Feynman diagrams (one diagram per term). Each term in those series can be interpreted as a particular interaction process.

[1] A unitary matrix is one whose inverse is equal to its transpose conjugate $U^{-1} = U^{\dagger}$

[2] An orthogonal matrix is a matrix whose inverse is equal to its transpose: $O^{-1} = O^{T}$

According to the Standard Model, the whole space is filled with different types of fields, and the excitation of those fields are what can be interpreted as particles, the visualization of properties of elementary particles are shown in Figure 2.2.1 . The matter is built by twelve particles called fermions [3], which has a half-integer spin, and they have to obey the Pauli exclusion principle. The interactions between fermions are perceived as an exchange of integer spin particles called bosons. None of them is known to have any underlying substructure. Thus they are called fundamental particles. All fermions can be further split into two groups: quarks and leptons. This distinction is driven by the interaction in which a particular fermion can participate.

The mathematical structure of the Standard Model can be described by the following symmetry group:

$$SU(3)_C \times SU(2)_L \times U(1)_Y \tag{2.3}$$

The group $SU(3)_C$ represents symmetry transformations in an internal colour space that are used to describe properties of the strong interactions. The dimension of the group corresponds to three types of colour charge. The second $SU(2)_L \times U(1)_Y$ symmetry group describes electro-weak interactions and takes into account the spontaneous symmetry breaking leading to massive intermediate bosons. The $SU(2)_L$ group describes rotations in an abstract weak-isospin space and the subscript L is used to express the fact that the fundamental representations of the weak interactions are left-handed dublets and right-handed singlets. The remaining $U(1)_Y$ group describes the electromagnetic interactions that should conserve weak hypercharge Y. In this case, the symmetry transformations can be interpreted as phase shifts of the wave function. The core idea behind the significance of this combined symmetry group is that the Lagrangian describing any process occurring in Nature must be invariant with respect to any transformation that belongs to it.

The strong interactions, which are mediated by eight massless gluons carrying a quantum number called colour and occurs between quarks, those interactions are described by a theory called Quantum Chromodynamic (QCT). Among all fermions, only quarks can carry colour charge, which allows them to interact via the strong interaction. There are six types of quarks known as flavors: up u, down d, charm c, strange s, top t and beauty b. Due to the unique nature of the strong interaction, in which the mediators of the force, the gluons, carry the same colour charge that they mediate, quarks are [4] "glued" to quarks and in Nature they always form colourless composite particles called hadrons.

Depending on the number of quarks they are made of, hadrons are classified as mesons (composed of $q\bar{q}$ pair), baryons (with three quarks combining all the colors) and exotic hadrons composed of four and five quarks, which has been recently discovered and reported by the LHCb collaboration [51].

[3]And their corresponding antiparticles

[4]This happens for all quarks besides the top quark, which lifetime is too short to combine with other quarks to form hadrons

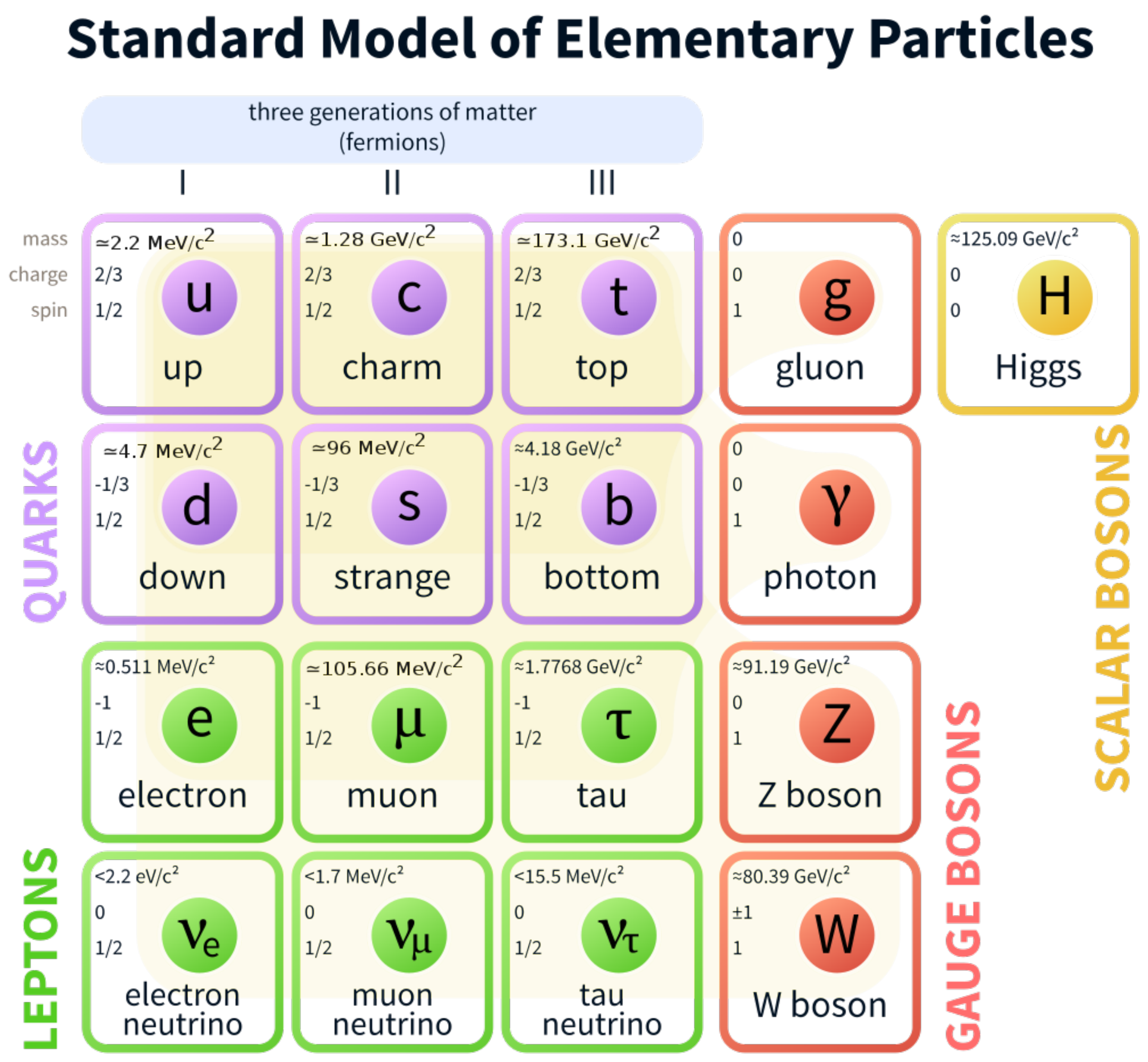


Figure 2.2.1: Illustration of the fundamental particles in the Standard Model and their properties. The fermions are organized in three generations (denoted by I, II, and III) comprised of quarks preseted as the purple squares and leptons as the green squares. The gauge bosons are shown as red circles, and the yellow square indicates the scalar Higgs boson.

2.3 Weak interactions and CKM matrix

As explained in the previous section, the $SU(2)_L \times U(1)_Y$ gauge group describes the properties of the electro-weak interactions. Those interactions are carried by four vector (i.e., spin 1) bosons, namely $W^{\pm}$, Z, and the γ. The first three of them mediate the weak interactions, and γ is responsible for carrying electromagnetic interactions. One of the key features of the weak interactions is the experimental observation that they couple to the left-handed particles only, which is a direct manifestation of the maximal violation of the charge and spatial parity symmetries. Another unique feature of the weak forces is the mixing between different quarks families, which leads in consequence to the experimental observation that the quark mass eigenstates are not the same as the weak eigenstates. The Cabibbo-Kobayashi-Maskawa (CKM) matrix relates the weak eigenstates, $(d\prime,s\prime,b\prime)$, with the mass eigenstates, (d,s,b), and is written as:

$$\begin{bmatrix} d' \\ s' \\ b' \end{bmatrix} = \begin{bmatrix} V_{ud} & V_{us} & V_{ub} \\ V_{cd} & V_{cs} & V_{cb} \\ V_{td} & V_{ts} & V_{tb} \end{bmatrix} \begin{bmatrix} d \\ s \\ b \end{bmatrix} \tag{2.4}$$

where the 3×3 unitary matrix V_{CKM} is known as the Cabibbo-Kobayashi-Maskawa (CKM) quark mixing matrix [39] [92]. The magnitude of each element in V_{CKM} represents the coupling strength of the quarks in the subscript to the weak field $W^{\pm}$. One of the properties of such matrix is the possibility to fully describe its transformation properties by three Euler angles and one complex phase. This phase parameter is vital for the theory since its non-zero value can explain, within the theoretical framework of the SM, the violation of Charge-Parity CP symmetry. Using the relation between the unitary matrices and rotation matrices the CKM matrix can be decomposed as follow:

$$\begin{aligned} V_{CKM} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_{23} & s_{23} \\ 0 & -s_{23} & c_{23} \end{bmatrix} \begin{bmatrix} c_{13} & 0 & s_{13}e^{-i\delta_{13}} \\ 0 & 1 & 0 \\ -s_{13}e^{i\delta_{13}} & 0 & c_{13} \end{bmatrix} \begin{bmatrix} c_{12} & s_{12} & 0 \\ -s_{12} & c_{12} & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} c_{12}c_{13} & s_{12}c_{13} & s_{13}e^{-i\delta_{13}} \\ -s_{12}c_{23} - c_{12}s_{23}s_{13}e^{i\delta_{13}} & c_{12}c_{23} - s_{12}s_{23}s_{13}e^{i\delta_{13}} & s_{23}c_{13} \\ s_{12}s_{23} - c_{12}c_{23}s_{13}e^{i\delta_{13}} & -c_{12}s_{23} - s_{12}c_{23}s_{13}e^{i\delta_{13}} & c_{23}c_{13} \end{bmatrix} \end{aligned} \tag{2.5}$$

where $c_{ij} = \cos(\theta_{ij})$, $s_{ij} = \sin(\theta_{ij})$, θ_{ij} are the respective quark mixing angles, and δ_{13} is a complex phase. Angle θ_{12} is called Cabbibo angle, which was introduced in 1963 when only two generation of quarks were known [38].

One of the customary parametrizations of the V_{CKM}, that is convenient to present the hierarchical structure of its parameters is called Wolfenstein parametrization [140]:

$$V_{CKM} = \begin{bmatrix} 1 - \frac{1}{2}\lambda^2 & \lambda & A\lambda^3(\rho - i\eta) \\ -\lambda & 1 - \frac{1}{2}\lambda^2 & A\lambda^2 \\ A\lambda^3(1 - \rho - i\eta) & -A\lambda^2 & 1 \end{bmatrix} + O(\lambda^4). \tag{2.6}$$

Experimentally, it is found that the mixing between mass and weak eigenstates is relatively small in the quark sector. Based on current knowledge [43], the values of V_{CKM} parameters are:

- $\lambda = 0.224837^{+0.000251}_{-0.000060}$,
- $A = 0.8235^{+0.0056}_{-0.0145]}$,
- $\rho = 0.1569^{+0.0102}_{-0.0061}$,
- $\eta = 0.3499^{+0.0079}_{-0.0065}$.

Therefore each of the mixing angles, represented as an off-diagonal element 2.4, is small, and the CKM matrix is approximately diagonal, see Figure 2.3.1. Therefore, processes that involve off-diagonal elements of the CKM matrix, those that change the generation of the quarks are Cabibbo-suppressed with respect to those on the diagonal, which are referred to as Cabibbo-favoured.

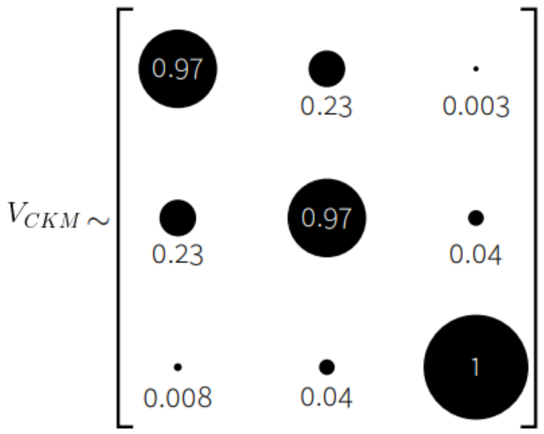


Figure 2.3.1: Hierarchy of the V_{CKM} matrix elements. The numbers are approximate to illustrate the relative magnitudes of the elements.

The V_{CKM} matrix is unitary one what leads to a series of relationships between different elements that can be experimentally probed. These constraints impose the following relations:

$$\begin{aligned} V^*_{1j}V_{1k} + V^*_{2j}V_{2k} + V^*_{3j}V_{3k} &= \delta_{jk} \\ V^*_{j1}V_{k1} + V^*_{j2}V_{k2} + V^*_{j3}V_{k3} &= \delta_{jk} \end{aligned} \tag{2.7}$$

Those six relations, summarized by Equation 2.7, can be visualized as, so called, unitarity triangles in the complex plane. The most popular to study is the triangle with $j = b$ and $k = d$. It drew attention due to having all sides of the similar sizes. This triangle can be completely constructed by defining that one of the sides lies on the real axis and then by defining its apex as:

$$(\bar{\rho}, \bar{\eta}) = -\frac{V^*_{ub}V_{ud}}{V^*_{cb}V_{cd}} \tag{2.8}$$

while the remaining apexes are $(0,0)$ and $(0,1)$. The angles of this unitary triangle denoted as α, β, γ are defined as follow:

$$\alpha = arg\left(\frac{V_{tb}^{*}V_{td}}{V_{ub}^{*}V_{ud}}\right), \qquad \beta = arg\left(\frac{V_{cb}^{*}V_{cd}}{V_{tb}^{*}V_{td}}\right), \qquad \gamma = arg\left(\frac{V_{ub}^{*}V_{ud}}{V_{cb}^{*}V_{cd}}\right)$$

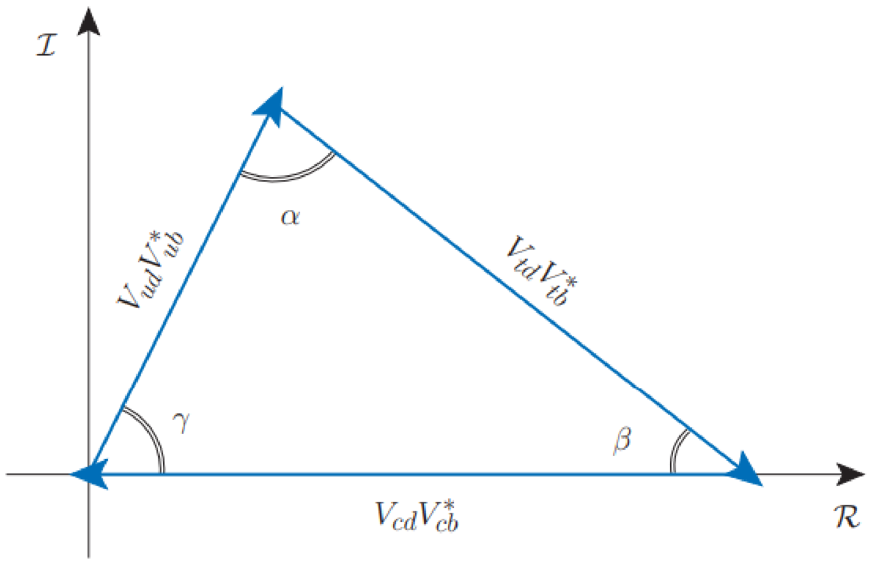


Figure 2.3.2: Visualization of one of the unitary triangles of the V_{CKM} matrix. Definition of the angles α, β, γ can be found in text.

The values of those angles cannot be predicted using the Standard Model framework. Instead, they must be measured experimentally. Any possible discrepancy in relation 2.7 may indicate a contribution from physics beyond the Standard Model. Figure 2.3.3 shows the overall status of the CKM unitary triangle in the $\overline{\rho}, \overline{\eta}$ plane, with a global fit to the apex. Within the current experimental uncertainties, there are no significant deviations from the Standard Model predictions.

2.4 Neutral Meson Mixing and *CP* violation

One of the most astonishing phenomena in physics is neutral meson mixing, i.e., the ability to change into its antiparticle spontaneously. Such transitions are related with the corresponding flavour quantum number violation (strangeness for K^0 mesons, charm for D^0 mesons and beauty for B^0, and B_s^0 mesons) and they can only be induced by the weak interactions. Mixing, also called flavour oscillation, is also an essential source of *CP* violation in the SM. Those mixing processes can be described by the so-called box diagram presented in Figure 2.4.1.

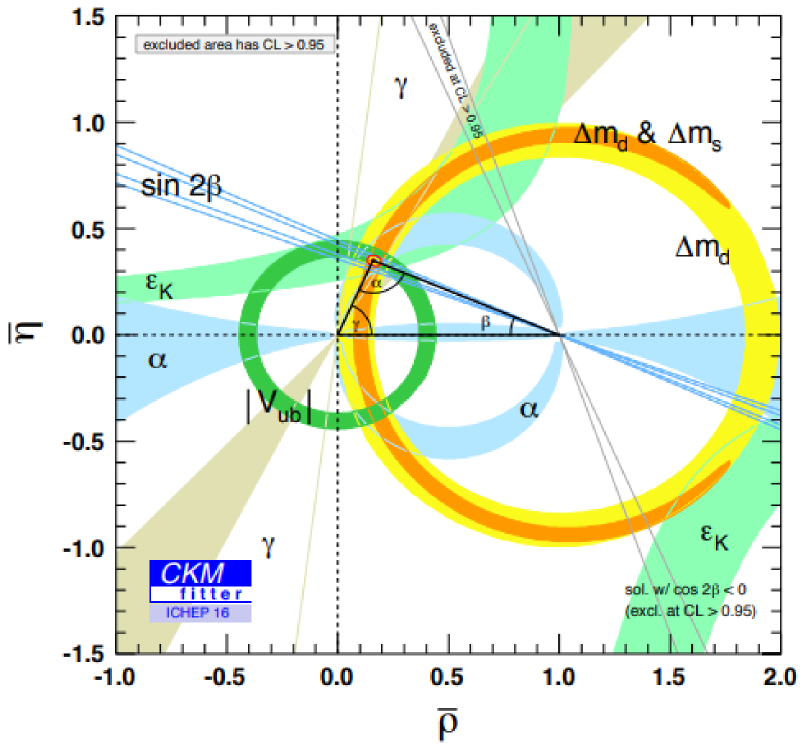


Figure 2.3.3: Overlapping constraints of the CKM unitary triangle. The red dashed area indicates global fit of the CKM apex with 68% confidence interval. Figure adopted from [43].

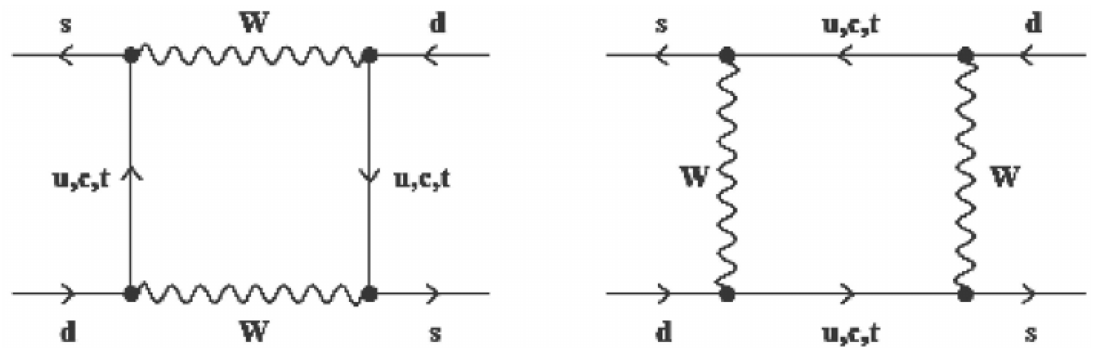


Figure 2.4.1: Box diagrams illustrating the mixing of K^0 mesons. In both cases virtual W bosons and quarks connect the four vertices. Figure adopted from [106].

The mass eigenstates, that propagate in space, are written as a superposition of flavor eigenstates:

$$\begin{aligned} |M_L\rangle &= p|M\rangle + q|\overline{M}\rangle \\ |M_H\rangle &= p|M\rangle - q|\overline{M}\rangle \end{aligned} \tag{2.9}$$

where p and q are complex numbers satisfying the normalization condition $|p|^2 + |q|^2 = 1$. The evolution of this system is described by the non-hermitian Hamiltonian, given as:

$$\mathcal{H} = M - \frac{i}{2}\Gamma \tag{2.10}$$

where M and Γ are the mass and decay matrices, respectively, defined as:

$$M = \begin{pmatrix} M_{11} & M_{12} \\ M_{12}^* & M_{22} \end{pmatrix}, \qquad \Gamma = \begin{pmatrix} \Gamma_{11} & \Gamma_{12} \\ \Gamma_{12}^* & \Gamma_{22} \end{pmatrix} \tag{2.11}$$

The Hamiltonian of the process, Equation 2.10, is not hermitian, thus ensures that the neutral meson may eventually decay. The diagonal elements of both M and Γ matrices are the same due to CPT theorem, and if the off-diagonal elements are equal to zero the meson states are degenerated and the mixing would not occur. Moreover, both M and Γ are hermitian operators. It is customary to define the average mass and average decay widths as follow:

$$M \equiv \frac{M_H + M_L}{2}, \quad \Gamma \equiv \frac{\Gamma_H + \Gamma_L}{2} \tag{2.12}$$

Next, two parameters x and y can be defined using the mass and decay width differences respectively:

$$x \equiv \frac{M_H - M_L}{\Gamma} = \frac{\Delta M}{\Gamma}, \quad y \equiv \frac{\Gamma_H - \Gamma_L}{2\Gamma} = \frac{\Delta\Gamma}{2\Gamma} \tag{2.13}$$

The mass difference, ΔM, can be related to the mixing frequency and together with the difference of decay width, $\Delta\Gamma$, can be measured experimentally.

The time evolution of the flavor eigenstates obeys the time-dependent Schrödinger equation:

$$i\frac{d}{dt}\begin{pmatrix} |M(t)\rangle \\ |\overline{M}(t)\rangle \end{pmatrix} = \mathcal{H}\begin{pmatrix} |M(t)\rangle \\ |\overline{M}(t)\rangle \end{pmatrix} \tag{2.14}$$

$$i\frac{d}{dt}\begin{pmatrix} |M(t)\rangle \\ |\overline{M}(t)\rangle \end{pmatrix} = \begin{pmatrix} M_{11} - i\frac{1}{2}\Gamma_{11} & M_{12} - i\frac{1}{2}\Gamma_{11} \\ M_{12}^* - i\frac{1}{2}\Gamma_{12}^* & M_22 - i\frac{1}{2}\Gamma_{22} \end{pmatrix}\begin{pmatrix} |M(t)\rangle \\ |\overline{M}(t)\rangle \end{pmatrix} \tag{2.15}$$

The solution of Equation 2.14 is a pair of eigenvalues given by:

$$v_1 = \begin{pmatrix} p \\ q \end{pmatrix}, \qquad v_2 = \begin{pmatrix} p \\ -q \end{pmatrix} \tag{2.16}$$

Those eigenvalues allows to diagonalize the hamiltonian:

$$QHQ^{-1} = P = \begin{pmatrix} M_L - \frac{i}{2}\Gamma_L & 0 \\ 0 & M_H - \frac{i}{2}\Gamma_H \end{pmatrix} \tag{2.17}$$

where Q is a matrix composed of vertically stacked v_1 and v_2, and P is an auxiliary matrix introduced to simplify the notation. Using mentioned notation the time evolution of the flavor states can be written as:

$$\begin{pmatrix} |M(t)\rangle \\ |\overline{M}(t)\rangle \end{pmatrix} = QPQ^{-1} \begin{pmatrix} |M\rangle \\ |M\rangle \end{pmatrix} = \begin{pmatrix} g_+(t) & \frac{q}{p}g_-(t) \\ \frac{p}{q}g_-(t) & g_+(t) \end{pmatrix} \begin{pmatrix} |M\rangle \\ |M\rangle \end{pmatrix} \tag{2.18}$$

where the parameters $g_\pm$ are given as:

$$g_+ = e^{-imt}e^{\Gamma\frac{t}{2}} \left[\cosh(\frac{\Delta\Gamma t}{4})\cos(\frac{\Delta M t}{2}) - i\sinh(\frac{\Delta\Gamma t}{4})\sin(\frac{\Delta M t}{2})\right] \tag{2.19}$$

$$g_- = e^{-imt}e^{\Gamma\frac{t}{2}} \left[\sinh(\frac{\Delta\Gamma t}{4})\cos(\frac{\Delta M t}{2}) - i\cosh(\frac{\Delta\Gamma t}{4})\sin(\frac{\Delta M t}{2})\right] \tag{2.20}$$

and $m = \frac{1}{2}(M_L + M_H)$, $\Gamma = \frac{1}{2}(\Gamma_L + \Gamma_H)$, $\Delta M = M_H - M_L$, $\Delta\Gamma = (\Gamma_L - \Gamma_H)$.

Equation 2.18 allows expressing the probability of mixing from particle to its antiparticle and vise-versa. Those probabilities are given by

$$|\langle\overline{M}|\mathcal{H}|M\rangle|^2 = \left|\frac{q}{p}\right|^2 |g_-(t)| = \frac{e^{\Gamma t}}{2}\left|\frac{q}{p}\right|^2 \left[\cosh(\frac{\Delta\Gamma t}{2})\cos(\Delta M t)\right] \tag{2.21}$$

$$|\langle M|\mathcal{H}|\overline{M}\rangle|^2 = \left|\frac{p}{q}\right|^2 |g_-(t)| = \frac{e^{\Gamma t}}{2}\left|\frac{p}{q}\right|^2 \left[\cosh(\frac{\Delta\Gamma t}{2})\cos(\Delta M t)\right] \tag{2.22}$$

Equation 2.21 clearly indicates that the mass difference between light and heavy states drives the mixing frequency. It is also important to analyze ratio $\frac{p}{q}$. If it differs from 1 then the mixing process violate *CP* symmetry. The word average of this mixing parameter for B^0 is $\frac{p}{q} = 1.0009 \pm 0.0013$ and for B_s $\frac{p}{q} = 1.0003 \pm 0.0014$ [135], which means the results are consistent with conservation of *CP* symmetry.

2.5 Baryon Asymmetry of the Universe and Sakharov conditions

The previous sections described the combined *CP* symmetry and discussed possible channels to study its violation. Here comes a vital question. Why do the researchers study the *CP* violation? Why is it so crucial that even it is mentioned inside of the logo of the LHCb experiment?

One of the answers to this question is the problem of the asymmetry of baryons with respect to anti-baryons. The current observed Universe is filled with baryons. This observation is contradicted to the cosmological measurement, which strongly indicates that in the era of the early Universe, the matter and antimatter should have been created in equal amounts. Therefore, there must have been some process that had created the matter-antimatter asymmetry. In 1967 Russian physicist Sakharov postulated three conditions that have to be fulfilled to make baryogenesis, or in other words existence of known Universe, possible [122]:

1. **Baryon number violation**. All known perturbative processes in the Standard Model result in equal numbers of baryons and anti-baryons. However, there are non-perturbative electroweak processes that can produce baryons without anti baryons [134].

2. *C* **and** *CP* **violation**. Violation of these symmetries are required even if there are processes, see the first condition, that could generate more baryon that anti-baryon an opposing process would generate an excess of anti-baryons, so the net baryon number of the system would still remain the same.

3. **Departure from thermal equilibrium**. Baryogenesis cannot occur at thermal equilibrium; otherwise, the inverse of this process will occur at the same rate, and a net asymmetry will not be generated.

The first condition is fulfilled by one of the default property of the Standard Model that requires the baryon minus lepton number to be conserved. Still, it does not have any restriction on the conservation of each of these numbers individually. The mechanism that would allow satisfying the third condition is the electroweak phase transition. Although, due to the mass of the Higgs boson $m_H \approx 126GeV/c^2$, the phase transition would only be weakly first order and not provide a strong enough departure from thermal equilibrium [93].

Within the Standard Model framework, the only source of the violation of *CP* symmetry is the weak interactions in the quark sector. Although the known *CP* violation in the quark sector is orders of magnitude too small to explain the baryon - anti-baryon asymmetry in the Universe, and therefore it is likely that this additional CP-violation originates in physics beyond the Standard Model. Therefore, precise comparisons of CP-violating observables and the Standard Model's predictions provide an invaluable probe of the New Physics.

The Artificial Intelligence is the New Electricity.

Andrew Ng

3 Principles of Machine Learning

This chapter provides a formal introduction to Machine Learning. It begins by presenting the concept of supervised learning. Next section is dedicated to discussing each of the models, which were taken into consideration during the author's research. It contains a brief mathematical description of each model and the overall approach to training them. The next section introduces the methodology of how the performance of each model can be measured and discusses the problem of the model's prediction interpretation. In other words, it tries to answer the vital question, "why should I trust the model that I built?". The final section covers the idea of bonsai Boosted Decision Trees, which are the binned version of the Boosted Decision Tree classifier. It explains the concept of discretization, its implementation, the reason this approach was chosen, and the issues it addresses.

3.1 What is Machine Learning?

This introductory section is dedicated to providing answers to the following questions:

- What is Machine Learning?
- Are there any difference between Machine Learning and classical algorithms?
- What types of Machine Learning models can be distinguished?
- Why is Machine Learning becoming more and more popular?
- What kind of problems can Machine Learning solve?

The classical approach to building software contains the following steps: collect system and software requirements, design the software architecture, then there is a coding and testing part, and the software finally goes to the maintenance phase. The presented model is the simplest one called the Waterfall[1]. The most important part is the coding phase, in which the developer or group of developers need to convert the system requirements, written in the Natural Language, into the code, which is understandable by a computer. They have to create a step-by-step set of instructions to process an individual input to determine the exact output. This strategy is not sufficient for many problems such as spam detection, image processing, or Natural Language understanding. For instance, the number of possible scenarios, e.g., sentences to translate from English to Polish, is so enormous that it is infeasible to create a dedicated logic for each of them.

In such cases, Machine Learning comes to the rescue. Instead of writing explicit instructions, a researcher can provide examples and train the model using the data to extract the hidden patterns. The only remaining problem is to collect a sufficient amount of good quality data. However, is it a problem at all?

Figure 3.1.1 shows the quantity of data generated per minute by the most frequently used online services. Facebook recently reported that the system that is designed to collect the logs generates about 2.5 PB of data per second [5], this is the approximate amount of the filtered data collected by the LHCb experiment during one year of nominal data taking. Another example of the online service that generates a massive amount of data is Twitter, according to its technical report, Twitter's users compose messages that required about 8 PB/day of data storage. One of the most challenging problems that the industry needs to solve is to understand the data and make a profitable conclusion based on it.

Machine Learning as a discipline can be divided into three main areas: **supervised**, **unsupervised** and **reinforcement learning**. Figure 3.1.2 presents all mentioned types of Machine Learning approaches together with a description of problems that can be solved by them.

The unsupervised learning describes problems, where the data has no manually assigned target value. One of the most interesting unsupervised problems is synthetical data generation. It is is usually performed using the idea of Generative Adversarial Networks (GANs) [73]. Another type of unsupervised problem is a dimensionality reduction, which reduces the number of random variables under consideration by selecting a set of principal variables. One of the methods that are widely used to achieve this goal is the PCA or Principal Component Analysis. PCA is a statistical procedure that reduces the dimensionality of the dataset by creating new uncorrelated variables that successively maximize variance. Finding such new variables, called the principal components, reduces to solving an eigenvalue/eigenvector problem [86].

Reinforcement learning employs an agent that interacts with an external environment and tries to learn the optimal behaviour strategy. This approach may lead to creating the program, that is capable to archive superhuman performance in such games as chess or Go [128], or can solve problem of protein folding [125]. The theory and application of the reinforcement learning is beyond the scope of this thesis, however author recommend the book [133], which the best resource to study this topic, and what is even more important it is available for free.

[1]The other approach to the software project management is called *agile*, which is a practice that promotes continuous iteration of development and testing throughout the software development life-cycle of the project. Both development and

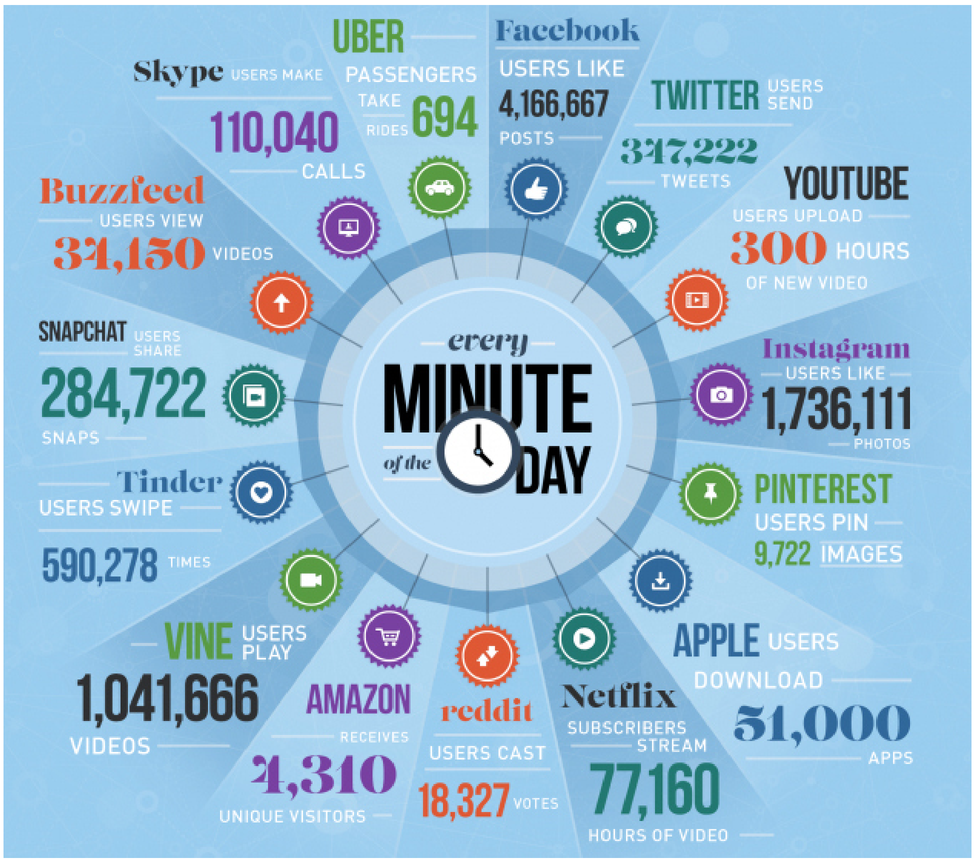


Figure 3.1.1: The amount of data generated per minute by the most popular online services.

Finally, supervised learning is a group of problems that are characterized by the presence of label that is assigned to each of the data examples. Supervised learning can be further split into classification and regressions tasks. Regression is a method of modelling a continuous target value (label) based on independent variables (predictors). This method is mostly used for forecasting [2] and finding out the relationship between variables. From the perspective of this thesis, the classification is the most important type of problem. Therefore the remainder of this section will focus on it. In this task, the model tries to find the mapping between the input space and one of the k labelled output.

As a concrete example, let us consider a problem of a real track segment (or seed) selection, which is a topic of chapter 4. This is a supervised classification problem because the input to the model was a pair $(track_seed, label)$, which were obtained using Monte Carlo simulation, and the task is to predict whether a particular seed is suitable for further processing.

To give a more precise and formal definition of the classification task, let us consider a given set of N training examples of the form $\{(x_0, y_0), \cdots, (x_{N-1}, y_{N-1})\}$, where $x_i \in R^M$ is a M-dimensional feature vector and $y_i \in \mathbb{N}_{>1}$[3] is a label. During the training phase, an algorithm tries to find the function $f: X \rightarrow Y$, where X is an input space and Y is the output space. The fitted function f is an element of some

testing activities are concurrent.

[2]One of the examples that are usually used when the regression problem is introduced is finding a house price when such features as house area or the number of bedrooms are given.

[3]Within the context of this thesis, the one class classification problem, e.g. an outlier detection, is not not taken into consideration as a classification one

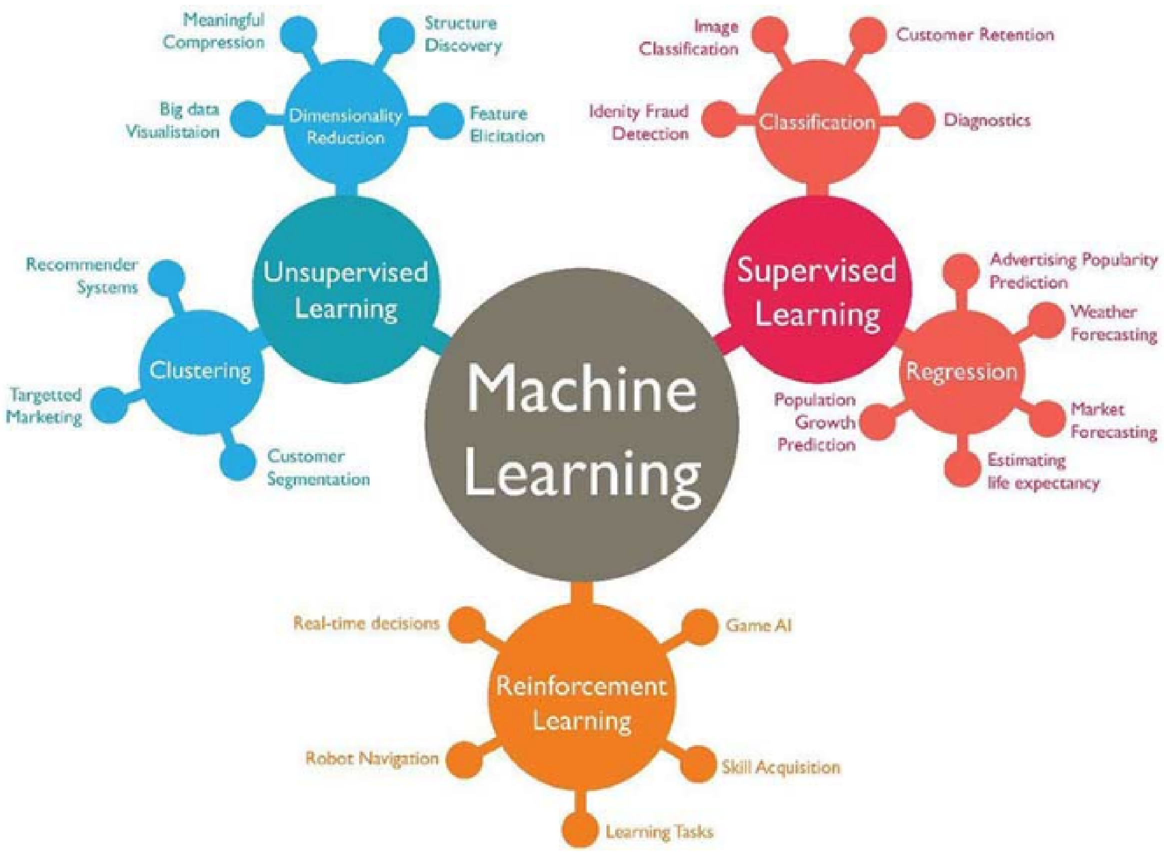


Figure 3.1.2: Graph showing the hierarchy of types of Machine Learning problems.

space of all possible mapping functions, which by convention, is called hypothesis space and denoted as $\mathcal{H}$. To measure how well the function f fits to the training data, a loss function $\mathcal{L}(y, f(x))$ needs to be defined [74]. This function reduces all the various good and bad aspects of a possibly complex system down to a single number. This scalar value allows candidate solutions to be ranked and compared to others.

When a loss function is selected the training procedure aims to minimize it with respect to the fitted function's parameters (these are the parameters of the selected model). This statement can be formalized in a following way:

$$f^*(x) = \underset{f \in \mathcal{H}}{\operatorname{argmin}} \mathcal{L}(y, f(x)) \tag{3.1}$$

where $f^*(x)$ is an optimal, from the perspective of minimizing the loss function with respect to the training data, function belonging to $\mathcal{H}$, and $\underset{x}{\operatorname{argmin}}$ is an operator when applied to a given function picks out the point in the function's domain at which its takes its minimum value (assuming that the point is unique), which can be formalized:

$$\underset{\zeta \in S \subseteq Z}{\arg\min}\, \Phi(\zeta) := \{\zeta \mid \zeta \in S \land \forall \xi \in S : \Phi(\xi) > \Phi(\zeta)\} \tag{3.2}$$

where $\Phi : Z \to Y$, and the S is a subset where the function Φ is defined.

3.1.0.0.1 Information Theory

To get a better understanding of a common choice for the cost function, the remainder of this section provides a brief introduction to the Information Theory. Back in 1964, Claude Shannon and Warren Weaver published the paper "The mathematical theory of communication" [126], where they introduced the concept of information entropy. This quantity was used to determine the optimal, with respect to the expected length, message encoding. The basic intuition behind the entropy is that more likely events are less informative than the rare ones. To be more precise, the entropy needs to fulfill the following requirements:

1. Rare events should have high information content.
2. In opposition, likely events should have very low information content, and in the case of a certain event, the information content should be 0.
3. The information content of two independent events should be equal to the sum of it.

To satisfy above properties, the Shannon entropy is defined in the following manner:

$$H(P) = \mathbf{E}_{x\sim P}\left[\frac{1}{logP(x)}\right] = -\mathbf{E}_{x\sim P}[logP(x)] = -\sum_{i=1}^{n} P(x_i)logP(x_i) \tag{3.3}$$

Where: $P(x)$ is the probability distribution of an event x to occur, $\mathbf{E}$ is the expectation value operator, and $x \sim P$ signifies that random variable x comes from distribution P. To summarize, the Shannon entropy measures the expected amount of information drawn from distribution P. It provides a lower bound on the length of encoding needed, on average, to encode a symbol taken from distribution P, which is visualized in Figure 3.1.3.

As an example, let us consider a variable m that has four possible states (a, b, c, d) for which the respective probabilities are given by $(\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8})$. The naive approach, where the creator of the encoding does not leverage the information about respective state probabilities, would require to transmit a message of length 2 bits. Although, taking into consideration the nonuniform probability of each state, the entropy can be expressed in the following manner:

$$H(m) = \frac{1}{2}log(\frac{1}{2}) + \frac{1}{4}log(\frac{1}{4}) + 2\frac{1}{8}log(\frac{1}{8}) = 1.75 \tag{3.4}$$

Which indicates that providing a better encoding the sender can save, on average, 0.25 bit per message. The example, encoding that gives a shorter code to the more likely events, using for instance the following encoding $\{a : 0, b : 10, c : 110, d : 111, \}$. For such a encoding the average message length is equal to:

$$\text{average code length} = \frac{1}{2}\cdot 1 + \frac{1}{4}\cdot 2 + \frac{1}{8}\cdot 3 = 1.75\text{bits} \tag{3.5}$$

Equation 3.5 shows a very interesting relation between entropy and optimal coding. In general, the entropy is a lower bound on the number of bits needed to transmit the state of a random variable. There is no way to get an average message length smaller than the entropy value.

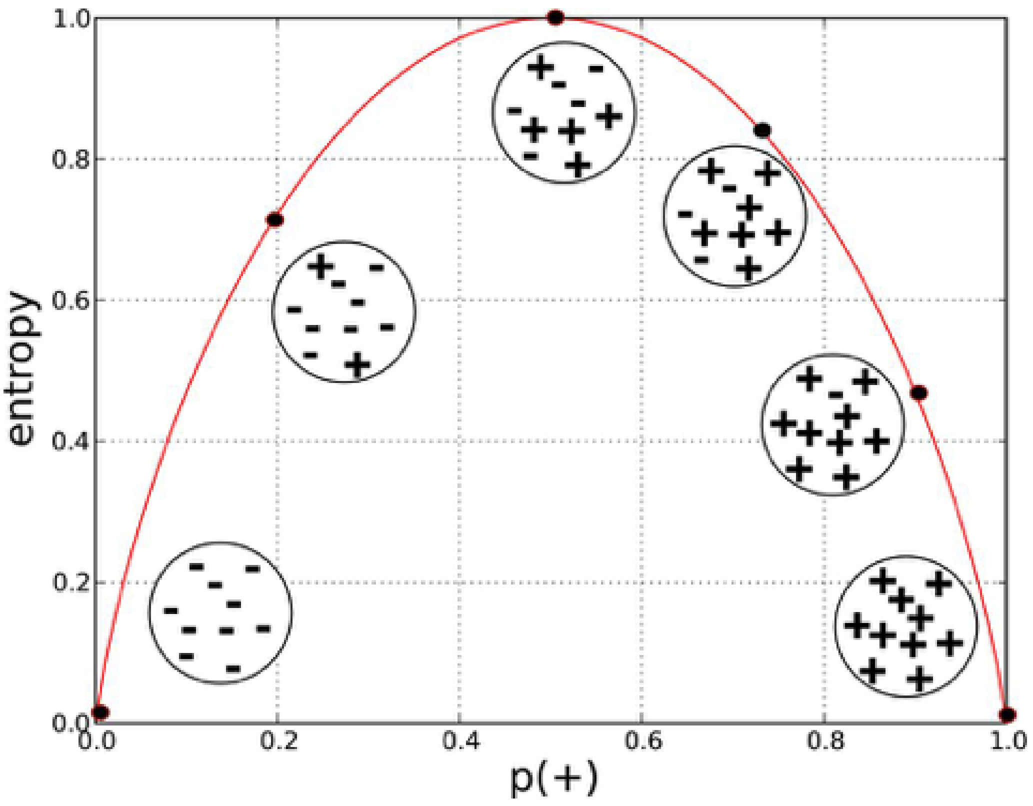


Figure 3.1.3: Visualization of the information entropy. The x-axis measures the proportion of data points belonging to the positive class, and the y-axis axis measures their respective entropies. Entropy is lowest at the extremes when the data either contains no positive instances or only positive instances. That is, when the data set is pure, the disorder is 0. Entropy is highest in the middle where the data is evenly split between positive and negative instances. Extreme disorder, because there is no majority. Figure adapted from [114]

The concept of entropy can be extended to the case of two separate probability distributions $P(x)$ and $Q(x)$ over the same random variable x. In such a case, the quantity called **Cross Entropy** measures the amount of information needed to send a message containing symbols drawn from distribution $P(x)$, when using encoding designed to minimize the length of messages coming from the probability distribution $Q(x)$. The cross-entropy is defined in a following way:

$$H(P, Q) = -\mathbf{E}_{x \sim P}[logQ(x)] = -\sum_{i=1}^{n} P(x_i)logQ(x_i) \tag{3.6}$$

One of the key properties of the cross-entropy is the fact that it is always non-negative. It shall take the minimum value of 0 only when $P(x)$ and $Q(x)$ are the same distributions. Therefore, the cross-entropy can be interpreted as kind of a distance metric, that measure similarity between two distributions. For the binary classification, this is the case where the number of classes to predict $k = 2$, the cross-entropy simplifies into the following form:

$$\mathcal{L} = -y \ln(f(x)) - (1-y)\ln(1-f(x)) \tag{3.7}$$

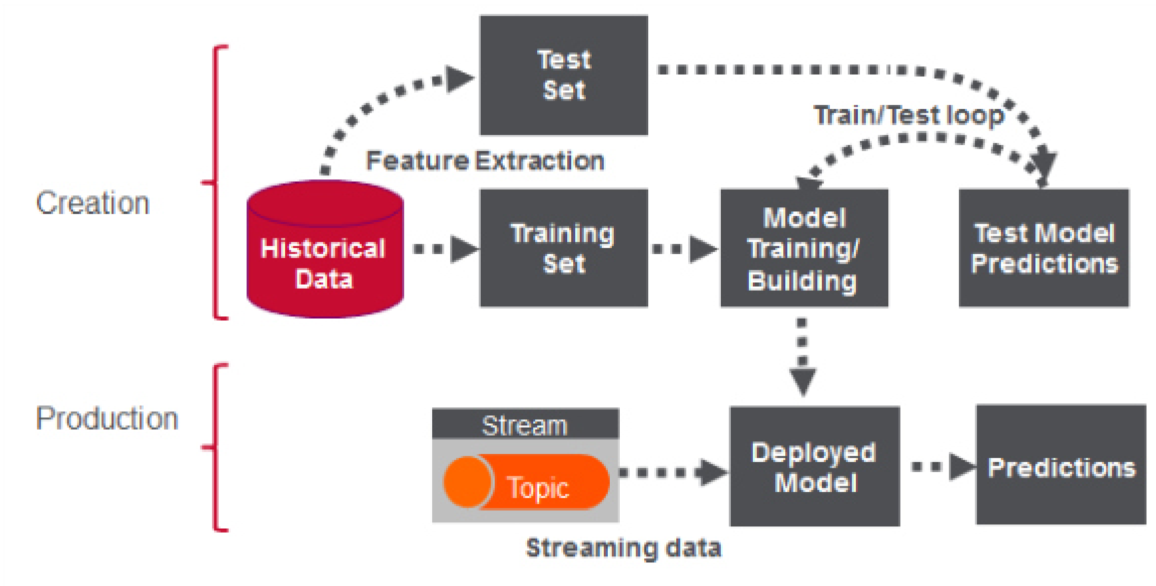


Figure 3.1.4: Simplified Machine Learning flowchart

Figure 3.1.4 presents a typical methodology to build a Machine Learning system in the form of a flowchart. Within this process, we can distinguish two phases: Creation and Production. The creation phase starts by collecting the data, and then the entire data is split into two subsets of training and testing samples. The former is usually divided into two disjoint sets training and validation sets.

The training set is used to select the hypothesis function, and the test set is used to verify the model's performance. The process of training the model is usually a very complicated iterative procedure, which consists of such steps as selecting the hypothesis space, finding an optimal set of model's hyper-parameters, etc. The way to robustly and properly measure the goodness of the model fitting is very often based on metrics calculated using cross-entropy.

3.2 Classification metrics overview

3.2.0.1 Confusion matrix

The starting point in every discussion of the classifier performance measurement is a confusion matrix. Karl Pearson invented the idea of the confusion matrix in 1904. For a binary classification problem, the outcome of the classification can have four possible values. If the particular example has a positive label [4] and the classification result is positive the outcome is called **true positive**; if it is classified as negative, it is counted as **false negative**. In case the true label was negative, and the classification estimated negative, it is counted as **true-negative**. Otherwise, the outcome is called **false-positive**. The false-positive and false-negative are also known as, using statistical language, type I and type II error, respectively. A two-by-two matrix can be constructed for better understanding and visualization of the classification outcome using the set of examples, usually called the test set. Figure 3.2.1 shows a confusion matrix

[4] In the field of High Energy Physics the positive example is usually called the signal, and the negative one the background

along with equations of selected standard metrics that can be derived from it. The elements from the diagonal represent the correct decision made by the classifier. In opposition, the off-diagonal numbers correspond to the miss-classified examples.

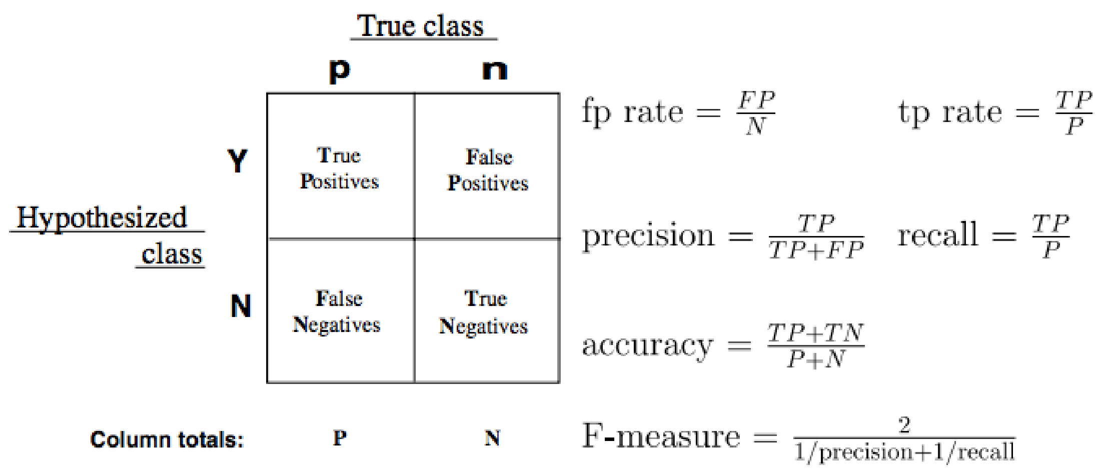


Figure 3.2.1: Confusion matrix and common performance metrics derived from it along with their respective formulas.

Using the information that the Confusion Matrix is built on, one can calculate the following quality metrics:

- **accuracy** $= \frac{TP+TN}{P+N}$, defines the number of examples that were correctly classified. Note, this quantity may be misleading. Consider the problem of an imbalanced dataset where the number of positive examples is 98% of the entire dataset. The dummy classifier, which always returns 1, achieves a 98% accuracy score.

- **precission** $= \frac{TP}{TP+FP}$; measure what proportion of events, that were correctly classified as a given class to all events that were classified as this class. Precision is a good measure to determine when the costs of False Positive is high.

- **recall** $= \frac{TP}{TP+FN}$ actually expresses how many of the actual positives model correctly classified. Recall shall be the model metric in use to select the best model when there is a high cost associated with False Negative. Recall can be interpreted as a measure of the model's sensitivity.

- **F1** $= \frac{2}{\frac{1}{precision}+\frac{1}{recall}}$ should be use, when one seeks for a balance between Precision and Recall and there is an uneven class distribution.

3.2.0.2 Receiver operating characteristics

The Receiver Operating Characteristics graphs are two-dimensional plots where the true positive rate (Y-axis) is plotted against the false positive rate (X-axis). A ROC graph depicts relative trade-off between benefits (true positives) and costs (false positives).

Figure 3.2.2 presents a sample of a ROC graph comparing different discrete classifiers. That kind of classifiers for a given input sample returns a single number as a decision, in the case of binary classification it is Yes or No decision. The origin, point (0,0), represents the classifier, which never issuing a positive classification outcome, which means its prediction regardless of the input vector is always 0. Such a classifier commits no false-positive errors but also gains no true positives. On the contrary, point (1,1) represents classifier, which always returns a true value. Generally, all points that lay on a straight line that satisfies equation $y = x$ correspond to the random guessing policy, and the corresponding classifiers are useless, and they have not extracted any patterns from the data.

Point D $(0, 1)$ corresponds to the perfect classification, which means all negative examples are suppressed, and all positive are preserved. In general, the classifier is better than another if its corresponding point in a ROC space is to the northwest[5] of the first.

Classifiers appearing on the upper right-hand side of a ROC graph may be considered as a *liberal*. They make positive classifications with weak evidence, so they classify nearly all positives correctly, but they often have high false-positive rates. Therefore the point A is more conservative than point B. The point E corresponds to the strategy, which is worst than the random guessing; this indicates some issue with data labelling, Usually, there are no such points in practical applications.

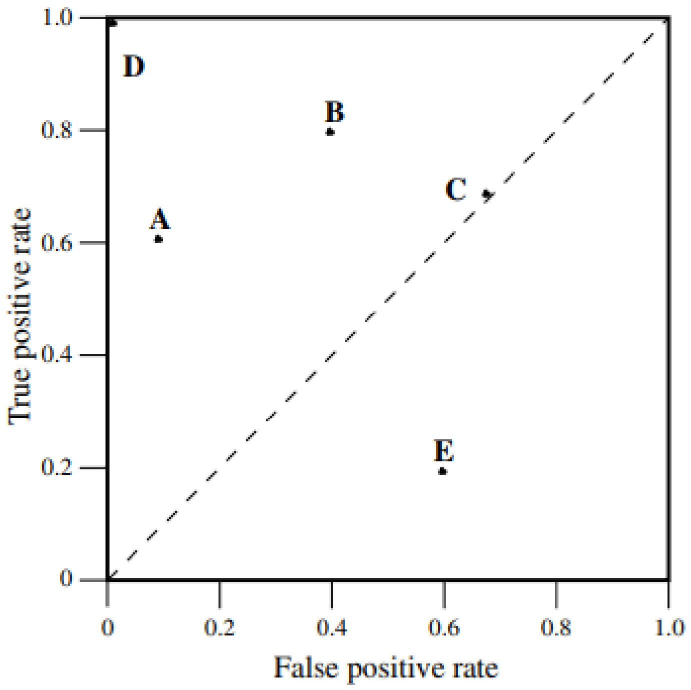


Figure 3.2.2: A exemplary Receiver Operating Characteristics graph indicating the performance of five discrete classifiers.

Most of the classifiers, instead of returning a discrete value, return a probability measure also called the score. These probabilities can be compared to thresholds to produce the binary classification. Each threshold value produces a different point in the ROC space, and a collection of those points creates the

[5] oriented according to the compass rose

ROC curve. The ROC curve is a very convenient way to find a classifier's operating point. The curve provides a convenient diagnostic tool to investigate one classifier with different threshold values and the effect on the True Positive Rate and False Positive Rate. One might choose a threshold in order to bias the predictive behaviour of a classification model. For instance, in the case of the track seed classification problem, the classifier has to preserve almost all true tracks while keeping the background as low as possible. Therefore, the classification threshold should be selected so that the true positive rate would be equal to 0.99.

ROC curve is a popular diagnostic tool for classifiers on balanced and imbalanced binary prediction problems alike because it is not biased to the majority or minority class. ROC analysis, in opposition to the study based on accuracy metrics, does not have any bias toward models that perform well on the majority class at the expense of the minority one, which is a desirable property that is quite attractive when dealing with imbalanced data [78].

The ROC curve is a two-dimensional representation of classifier performance. This representation may not be convenient when comparing the performance of two or more classifiers. Thus, it is customary to reduce the ROC curve to a single scalar value representing the expected classifier performance while keeping its properties. The common practice is to calculate the Area Under the ROC Curve (ROC AUC). From the statistical point of view, the ROC AUC can be interpreted as a probability that the classifier will rank a randomly chosen positive example higher than a randomly chosen negative sample.

3.3 Model selection

This section provides a detailed description of the model selection and validation procedure. Within the scope of this project, four different models were tested. Each of the subsection starts by presenting the mathematical formalism of a particular model. The next part focuses on applied methodology to find the optimal, in the sense of maximizing the selected metric score, set of model's hyper-parameters, see section 3.3.6. The first two presented models were constructed using the sklearn [112] and the numpy [76] libraries.

3.3.1 *k*-Nearest Neighbors classifier

The *k*-NN was selected to play the role of a study baseline. The *k*-NN is a **non-parametric** model, which means it takes no assumption on the underlying probability distribution of data [55]. Instead, the model's fundamental assumption is that similar inputs should produce similar outputs. To make a prediction, the model looks at the K points in the training set nearest to the input x_i and counts how many members of each class are in this set and finally returns the empirical fraction as the estimate. That statement can be formalized in terms of probability:

$$p(y = k, \mathcal{D}, K) = \frac{1}{K} \sum_{i \in N_K(x, \mathcal{D})} 1(y_i = k) \tag{3.8}$$

where: $N_K(x, \mathcal{D})$ are the K nearest points to x in $\mathcal{D}$ and $1(y_i = k)$ is the indicator function defined as follow:

$$1(e) = \begin{cases} 1 & \text{when e is true} \\ 0 & \text{when e is false} \end{cases} \tag{3.9}$$

In order to make a prediction, the algorithm needs to evaluate the following steps:

```
Data: Training data $x_i^{train}, y_i$, and test data $x^{test}$
Result: Predicted test output $\hat{y}$
Load the training and test data;

Find the k training data points $x_i$, which has the shortest distance $\mathcal{D}(x_i^{train}, x^{test})$;
Decide $\hat{y}$ with a majority vote among those k nearest neighbours;
```

Algorithm 3.1: k-nearest neighbour, *k*-NN

The number of neighbours *k* and distance function are the only two hyper-parameters that need to be selected apriori by a machine learning practitioner. In the case where the choice is not intuitive, usually, an extensive scan of the parameter space should be performed as a part of the verification studies. Figure 3.3.1 visualize the influence of the different values of *k* on the classifier decision boundaries. The decision boundary is a disjoint region where each of the points is classified into the same class. It is clearly visible that considering more neighbours produce a smoother decision boundary.

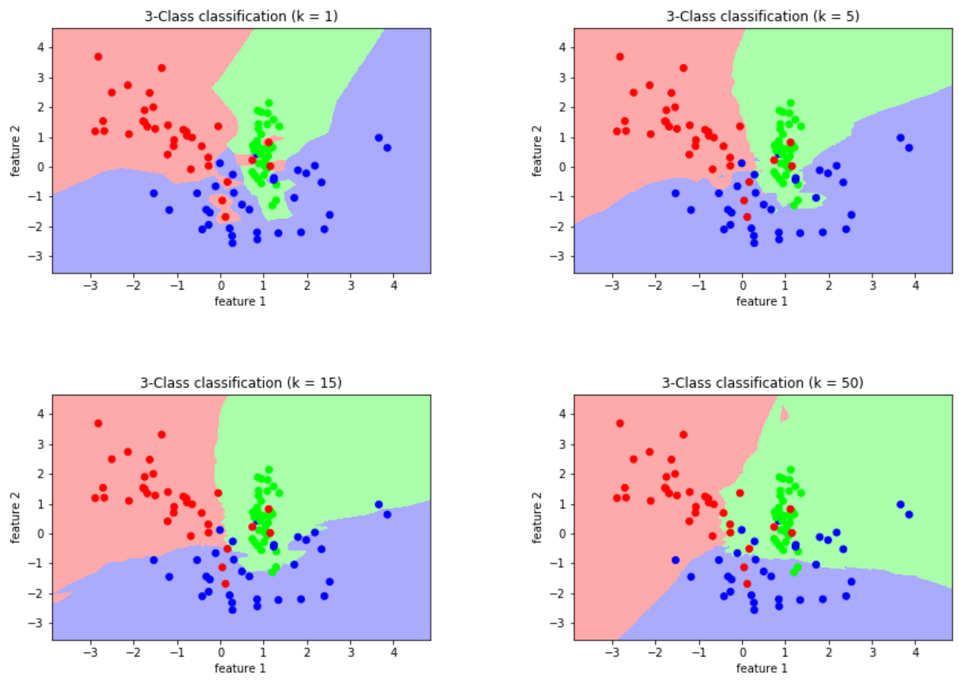


Figure 3.3.1: *k*-NN decision boundaries. Each plot was generated for a different value of the nearest neighbours. The data (100 points) and the labeling function used to generate these plots were selected randomly.

K-NN model has two limitations. First of all the K-NN's prediction time and space grow linearly with the number of training examples[6], which make this model infeasible when the amount of the data samples are very large (i.e., $10^5 - 10^6$ events). The second limitation of the k-NN models is the fact that this model suffers from the **curse of dimensionality**. In low dimensional spaces, data may seem very similar, but the higher the dimension, the further these data points may seem to be. To illustrate this phenomenon let consider a collection of points x_i that are sampled uniformly within the unit d dimensional hypercube $\forall i, x_i \in [0, 1]^d$, and the model that is looking for $k = 10$ neighbours of a particular test point. Let l be the edge of the smaller hypercube that contains all k-nearest neighbour of a given test point, then

$$\begin{aligned} l^d &\approx \frac{k}{n} \\ l &\approx (\frac{k}{n})^{\frac{1}{d}} \end{aligned} \tag{3.10}$$

If the training dataset contains $n = 1000$ entries each of dimensionality $d = 20$, then $l \approx 80\%$, which means almost the entire space is needed to find the 10-NN. This breaks the k-NN assumption because the k neighbors are not particularly close to each other, and therefore similar, to any other data points in the training set. If we take into consideration that dimensionality of the track classification problem, it is not expected to get promising results. Although training the k-NN is pretty fast, so seems to be a reasonable choice for a baseline model.

3.3.2 Logistic Regression

Another family of models that is usually trained in order to get a study baseline is Logistic Regression, which is one of the simplest parametric classification models. The name of this model may be confusing. Despite having term "regression" inside its name, it is a classification model. It is based on the assumption that the output is a linear function of the inputs:

$$y(x) = \sigma(w^T x + b) = \sigma\left(\sum_{i=1}^{D} w_i x_i + b\right) \tag{3.11}$$

where: $w^T x$ is a dot product between the input vector x and the model's weights w, and b is a bias term. In this context, the bias term means the model's output when no input signal is given. Do not confuse with the statistical "bias", which represents the difference between true parameter value and the estimator's expected value. The σ is a **sigmoid** function. It is defined as:

$$\sigma(x) = \frac{1}{1 + \exp(-x)} \tag{3.12}$$

The term "sigmoid" means S-shaped, which is shown in Figure 3.3.2. This function is used to map the whole real axis into a finite interval. In the case of classification, it squashes the regression output (term

[6]There are a couple of methods like KD-trees that reduce K-NN evaluation complexity. Still, it does not affect the space complexity of the model

$w^T x$) into the interval $\{0, 1\}$, which can be interpreted as a probability. Positive input numbers become high probabilities, and the negative values become low ones. The first derivative of this function can be expressed in the following way:

$$\frac{d\sigma(x)}{dx} = \sigma(x) \cdot (1 - \sigma(x)) \tag{3.13}$$

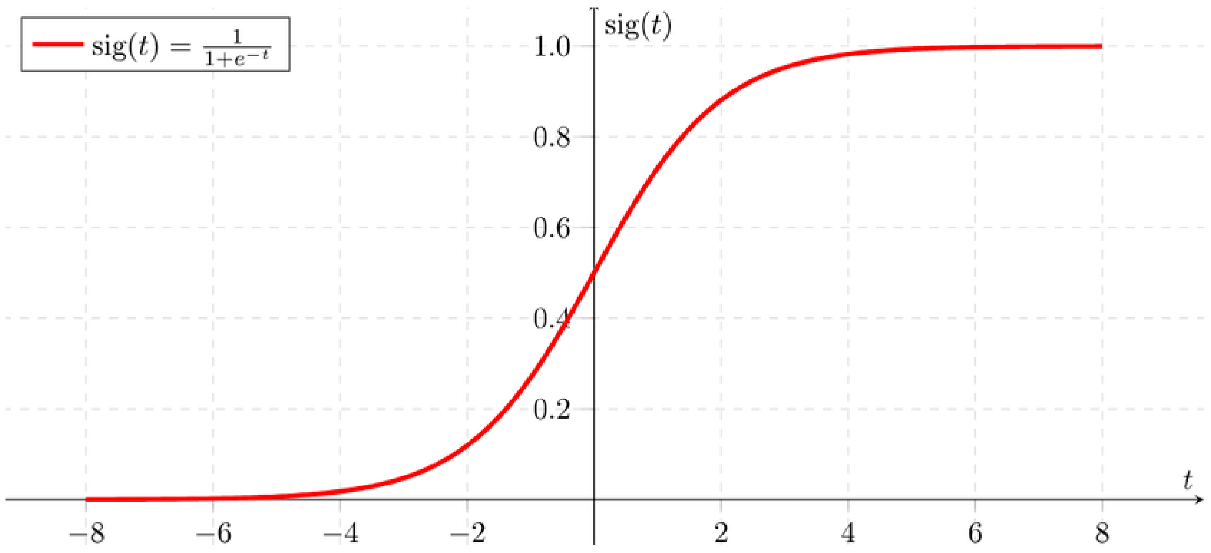


Figure 3.3.2: Plot of the sigmoid function.

In the case of binary classification, $y \in \{0, 1\}$, the model 3.11 can be rewritten in the following probabilistic form:

$$p(y|x, w) = B(y|\sigma(w^T x + b)) \tag{3.14}$$

where: $B(z|\theta)$ is a Bernoulli distribution defined as:

$$B(z|\theta) = \begin{cases} \theta & \text{if z = 1} \\ 1 - \theta & \text{if z = 0} \end{cases} \tag{3.15}$$

the θ is a probability of a success. The Bernoulli distribution is a special case of a binomial distribution, where $n = 1$.

The model fitting procedure can be derived using Maximum Likelihood Estimation (MLE) technique [7] [30]. The likelihood function is a quantity similar in its nature to the probability. The difference is that the probability is used to describe the future outcome of the random process, and the likelihood tells what is the probability that observed data were sampled from the process described by some specific model and is a function of the model's parameters.

For a dataset $D := \{x_n, y_n\}$ where $y_n \in \{0, 1\}$ and $n = 1 \cdots N$ the likelihood of the entire dataset can be written as:

[7]MLE is used to find $\tilde{\theta} = \underset{\theta}{\text{argmin}} logp(D|\theta)$, where D is a data

$$L(w) = \prod_{i=1}^{N} B(y_i|x_i, w)$$
$$= \prod_{i=1}^{N} \sigma(w^T x_i)^{y_i} \cdot \left[1 - \sigma(w^T x_i)\right]^{1-y_i} \tag{3.16}$$

Equation 3.16 assumes that the training examples are independent and identically distributed (iid). In the practical application instead of maximizing the likelihood 3.16 it is more convenient to minimize negative logarithm of likelihood according to the following formula:

$$NLL(w) = \sum_{i=1}^{N} y_i log\left(\sigma(w^T x_i)\right) + (1 - y_i)log(1 - \sigma(w^T x_i))\Big) \tag{3.17}$$

Formula 3.17 is identical to the cross-entropy 3.7 for a case when number of classes is two. This provides an additional justification for using cross-entropy as a cost function.

The conventional approach to finding a minimum of the cost function and setting the derivatives of this function with respect to the parameters w fails due to the lack of closed-form for the minimum. Instead, the iterative optimization algorithm has to be applied. Within the scope of this thesis, the Gradient Descent algorithm will be discussed.

3.3.2.0.1 Gradient Descent Optimization

The Gradient Descent is an optimization algorithm that attempts to find a minimum of a function by taking small steps in the direction of the gradient vector and ends at the local minimum. In the case of a linear model, it can be proved that it will always be a global minimum [30]. The proof is based on analysis of the Hessian matrix, which for the case of linear model [8] is always positive defined[9]. That kind of optimization is called convex, and it is the reason why generalized linear models still play an important role in classical Machine Learning. During a single optimization step the new, updated value of the parameter w_i is calculated according to the following formula:

$$w_i^{new} = w_i^{old} - \eta \cdot \frac{\partial \mathcal{L}}{\partial w_i^{old}}$$
$$= w_i^{old} - \eta \cdot \sum_{j=0}^{N} \left(\sigma((w^{old})^T x_j) - y_j\right) x_j \tag{3.18}$$

where: η is a step size, also called learning rate, which has to be set before the training procedure and $\mathcal{L}$ denotes the loss function. The wrong choice of the learning rate value can pose a number of problem for the training algorithm, which is shown in Figure 3.3.3. Setting it to a value that is too low may lead

[8]Linear term in a linear model refers to the parameters but not necessarily to the features, for instance, $f(x) = w_1 x + w_2 x^2 + w_3 \sin(x)$ is considered as a linear model

[9]$n \times n$ matrix M is positive defined if for all vector $\vec{x} \in R^n$ $\vec{x}^T M \vec{x} > 0$. Intuitively, a positive defined matrix is a matrix generalization of a positive number, which does not change the sign of the number

to a very long training process that may not be practical. On the other hand, setting too high may lead, in turn, to rapid oscillations in the loss function and usually, the algorithm will fail.

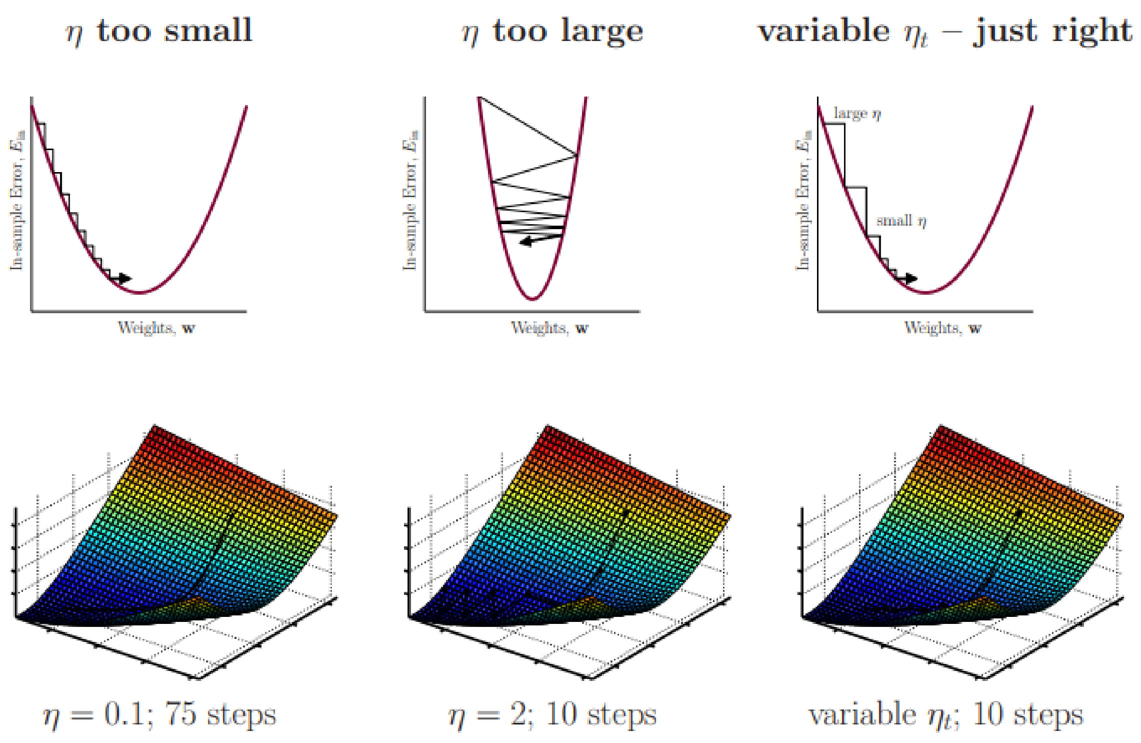


Figure 3.3.3: Comparison of the learning rate values and its influence on a final model performance. Figure taken from [12]

The second parameter of the gradient descent algorithm 3.18 is a number of examples N, that are used to calculate the new weights. There are three versions of this optimization method. The first one also called Stochastic Gradient Descent (SGD), update weights after each training example, which is equivalent to set $N = 1$. For $N > 1$ it is called Mini-batch Gradient Descent. The final method uses N equal to all examples, and it is usually called Batch Gradient Descent. Those tree methods and the corresponding weight updates are visualized in Figure 3.3.4. The SGD frequently updates the model, which provides immediate insight into its performance and improvement rate. However, this method's downside is noisy gradient information, which may affect the model parameters to oscillate around the minimum value, which may reveal a higher variance over training epochs. On the other hands, the Batch version of the Gradient Descent require a lot of memory and can be very slow for large datasets, and more stable error gradient may result in coverage to less optimal local minimum.

Several enhanced versions of GD algorithm were proposed in the literature, the summary of them can be found in [119]. From the perspective of deep learning, which is a topic of section 3.3.4, the ADAM optimization algorithm is the most recommended one [91].

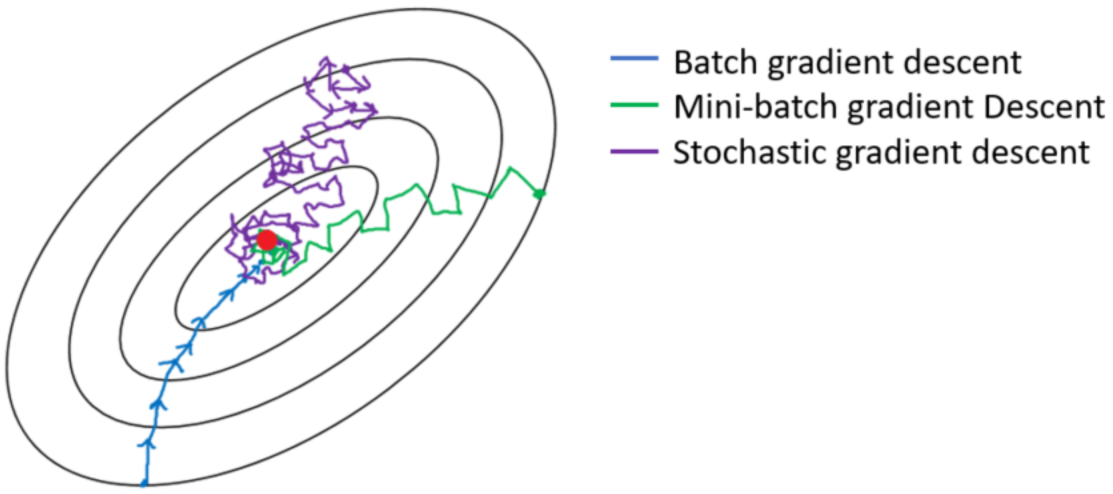


Figure 3.3.4: Comparison of the different variants of the gradient descent algorithms and its influence on weights update path toward minimum.

3.3.2.0.2 Regularization When training any classification model, the general task is to make it perform well not only on a training set but also on a new, previously unseen data sample. This objective is called generalization and can be quantified by a test error. One of the strategies to reduce the gap between training and test errors is called regularization. There are many forms of regularization. Within this thesis's scope, the modification of the cost function [10] in order to discourage the parameters from reaching a large value. A model with one or a few large parameters means that model is using only a small subset of features to make a decision. To avoid that kind of situation, the additional penalty term can be added to the loss function:

$$\mathcal{L}_{reg}(w) = \mathcal{L}(w) + \Omega(w) \tag{3.19}$$

where: $\mathcal{L}$ is the previous, non-regularized loss function, and the $\Omega(w)$ is a regularization term that is a function of a model's weights.

The simplest and most popular choice of the $\Omega(w)$ is a sum of squares of the the model's coefficients[11]. In such a case, the cross-entropy loss function 3.7 is expressed by the following formula:

$$\mathcal{L} = \sum_{i=1}^{N} y_i log\left(\sigma(w^T x_i)\right) + (1 - y_i) log(1 - \sigma(w^T x_i)) + \frac{\alpha}{2} w^T w \tag{3.20}$$

where: α is a hyper-parameter that weight a relative contribution of a squared norm penalty. That kind of regularization is also known as weight decay. It can be shown that the addition of the L^2 regularization term modifies the learning rule to shrink the weight vector on each optimization step.

[10]The other popular regularization method is data augmentation, which is used to generate a new training example by applying some transformation function on old examples. The proper choice of a transformation function in the case of high-dimensional tabular data is not obvious; that why it was not implemented.

[11]that kind of regularization is also called Ridge Regression or L^2

The second popular way of expressing $\Omega(w)$ is called a lasso regression:

$$\Omega_{l_1}(w) = \frac{\alpha}{2}||w|| = \frac{\alpha}{2}\sum_{i=1}^{M}|w_i| \tag{3.21}$$

Adding the lasso regularization term is equivalent to a feature selection mechanism. The proof of this statement is based on an analysis of the lasso regularization gradient, which is formulated by

$$\frac{d\Omega_{L_1}(w)}{dw} = sign(w) = \begin{cases} 1 & \text{when } w > 0 \\ 0 & \text{when } w = 0 \\ -1 & \text{when } w < 0 \end{cases} \tag{3.22}$$

Formula 3.22 indicates that L_1-regularization will move any weight towards 0 with the same step size, regardless the weight's value. In other words, if the α hyper-parameter is sufficiently large, some of the weights are driven to zero, causing the optimization solution to be sparer. In contrast, the L_2 gradient is expressed as

$$\frac{d\Omega_{L_2}(w)}{dw} = w \tag{3.23}$$

Formula 3.23 shows that the weight's update, see Equation 3.18, linearly decrease towards zero as the weight goes towards zero. Therefore, L_2-regularization will also move any weight towards zero, but it will take smaller and smaller steps as a weight approaches zero. Therefore, the model never reaches a weight of zero, regardless of how many steps it takes. Figure 3.3.5 shows the difference between these two regularization methods.

The statistical justification of using both $L1$ and $L2$ regularization comes from Maximum A posterior Estimation (MAP), which enhance the maximum likelihood method by adding prior factor. The MAP framework is one of the tools of the Bayesian statistic, which allows adding a probability distribution over the model's parameters (external heuristics). From a conceptual standpoint, the interpretation is that one has some prior knowledge about the possible values of the unknown parameters (for instance a physics law such as momentum conservation).

3.3.2.0.3 Logistic Regression and data linear separability

The logistic regression model has zero training error rate only when the data is linearly separable, which means that there is an n-dimensional hyperplane that can separate data into classes. Figure 3.3.6 presents example of such a data.

The linear separability of the data is a strong assumption and, in practical applications, holds rarely. To produce the nonlinear decision boundaries, two methods can be used. The first one is to handcraft a function $\varphi(x)$ that transforms data into the form that makes it linearly separable. This task is very challenging, taking into consideration the high dimensionality of the practical classification problems. The second idea, discussed in the next sections, is to build a classifier that can generate nonlinear classifica-

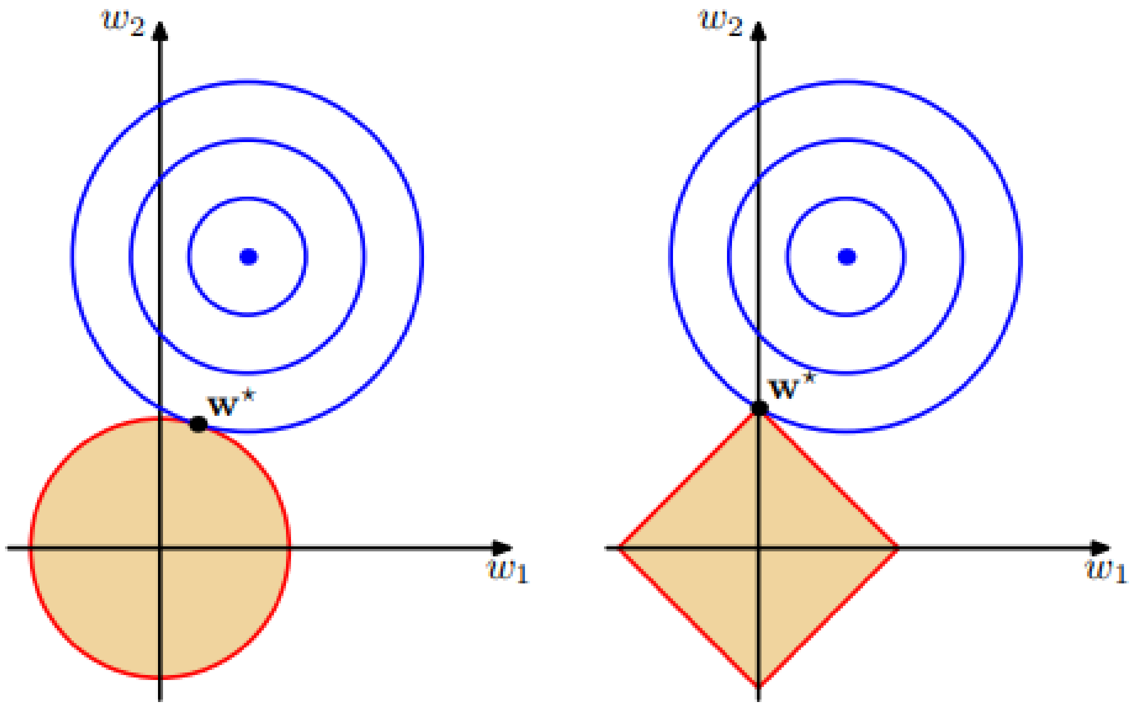


Figure 3.3.5: Visualization of the impact of the regularization term on the optimization solution. The blue contours plot represents regularized cost function along with a $L2$ (left) and $L1$ (right) regularization terms. The optimum value of the parameters is denoted by w^*. The lasso regularization gives a spare solution in which $w_1^* = 0$. Figure taken from [30]

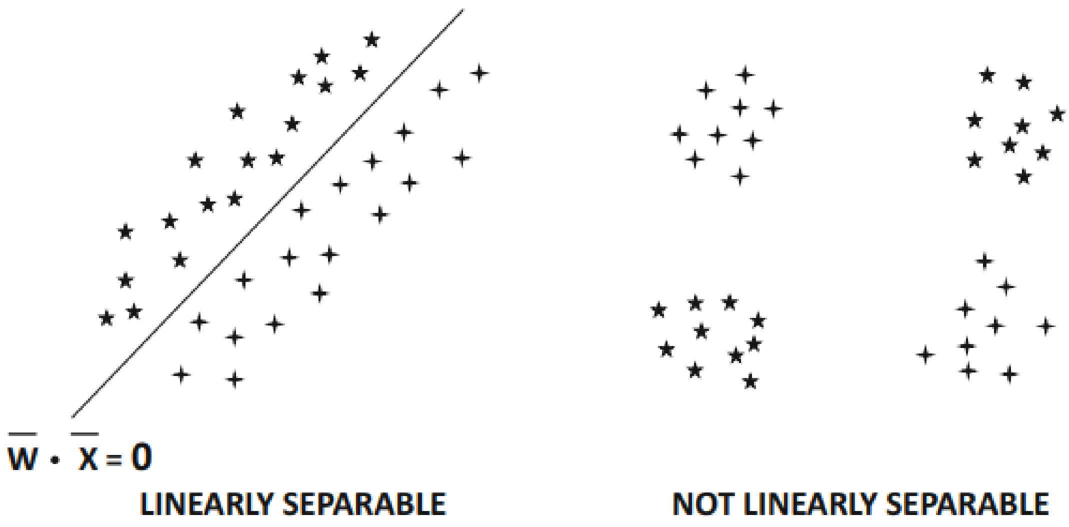
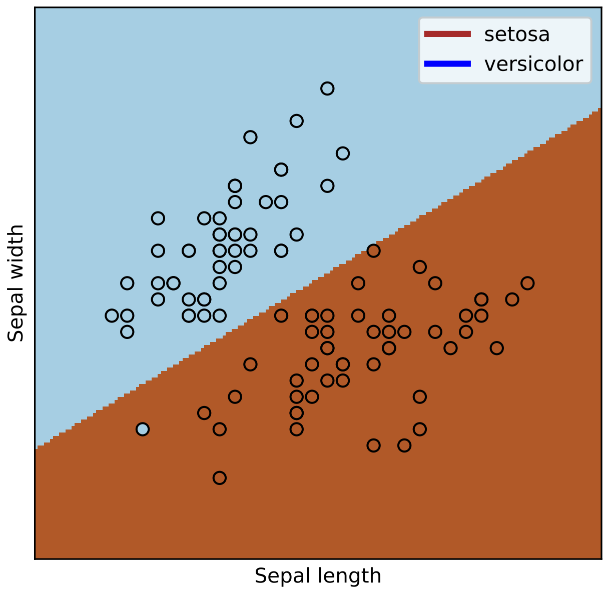


Figure 3.3.6: Visualization of the linear separability of a data.

tion rules.



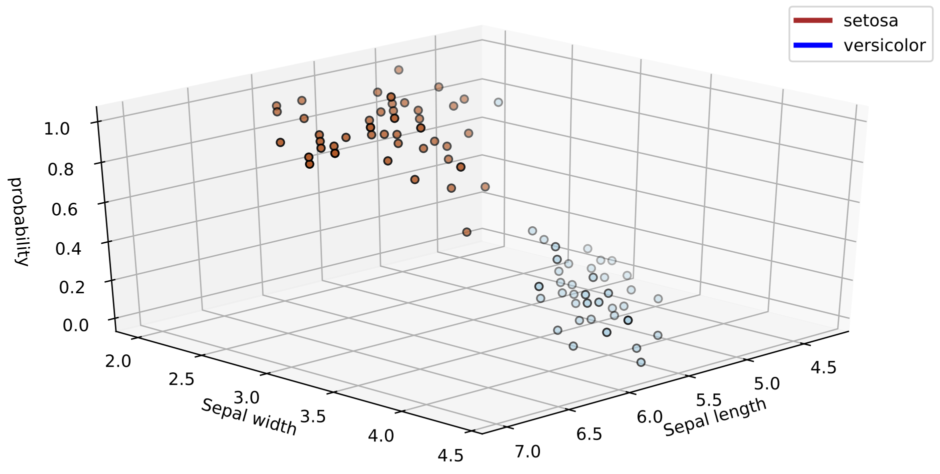


Figure 3.3.7: Logistic regression decision boundaries. The bottom figure presents a 3D plot, which visualize the classifier output (horizontal line) versus feature values.

3.3.3 XGboost Classifier

The XGboost stands for Extreme Gradient Boosting [45]. It is the state-of-the-art implementation of the Gradient Boosted Classifier. The Gradient Boosted Classifier is a model that uses k additive functions to predict the output. Each of these functions is called a weak learner. The next section discusses one of the usual choice of week learner - Decision Tree Classifier.

3.3.3.1 Decision Tree Classifier

The Decision Tree Classifier is a parametric model implemented by recursively partitioning the input space, and defining a local model in each resulting region of input space. This can be represented by a so-called tree object, with one leaf per region. Analytically, the model can be expressed in the following form:

$$f(x) = \sum_{l=1}^{L} \gamma_l 1(x \in \mathcal{R}_l) \tag{3.24}$$

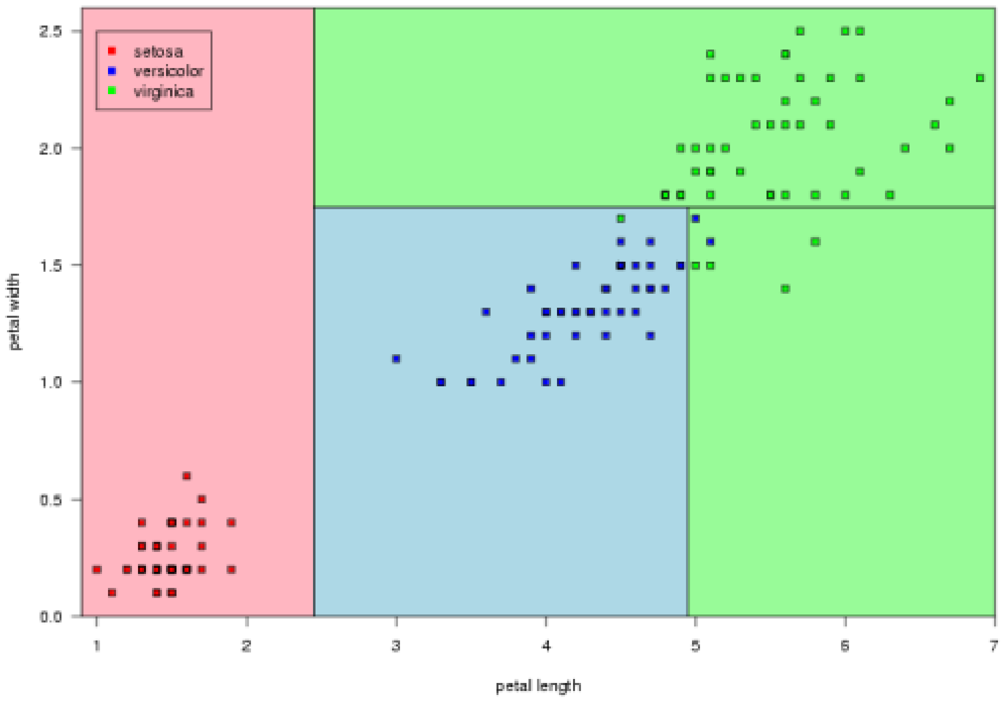
where: L is the number of disjoint regions $R_1, R2, \ldots, R_L$ of the entire input space, γ_l is a local response for region R_l, and x is a input N-dim vector. The decision whether to make an additional split is based on the information gain metric. The information gain is closely related to the information entropy 3.3:

$$IG(P, a) = H(P) - H(P|a) \tag{3.25}$$

where $H(P|a)$ is the conditional entropy of P given the value of the attribute is a. The information gain can be interpreted as a change in the entropy when attribute a is observed. The process of the partitioning of the input space is performed by the procedure, in which greedy selects the attribute that gives the highest information gain. From the practical perspective, Decision Tree (DT) algorithm is implemented in the following way:

```
Data : Training attribute-value dataset $D$
Tree = {}
if all instance in $D$ has the same class $c$ then:
    label(Tree) $\leftarrow c$
    terminate
else if attributes set $= \emptyset$ or no attributes has positive information gain then:
    label(Tree) $\leftarrow$ most common class in $D$
    terminate
for all attributes $a \in D$ do:
    $a_best$ = Best attributes according to the information gain
    Tree = create decision tree node with $a_best$ as a root
    $D_a$ = Induced sub-dataset from $D$ based on $a_best$
    for each value of attributes in $D_a$ :
        $Tree_a$ = BuildDecisionTree($D_a$)
        attach $Tree_a$ to the corresponding branch Tree
end for
return Tree
```

Algorithm 3.2: Building Decision Tree (psudocode)



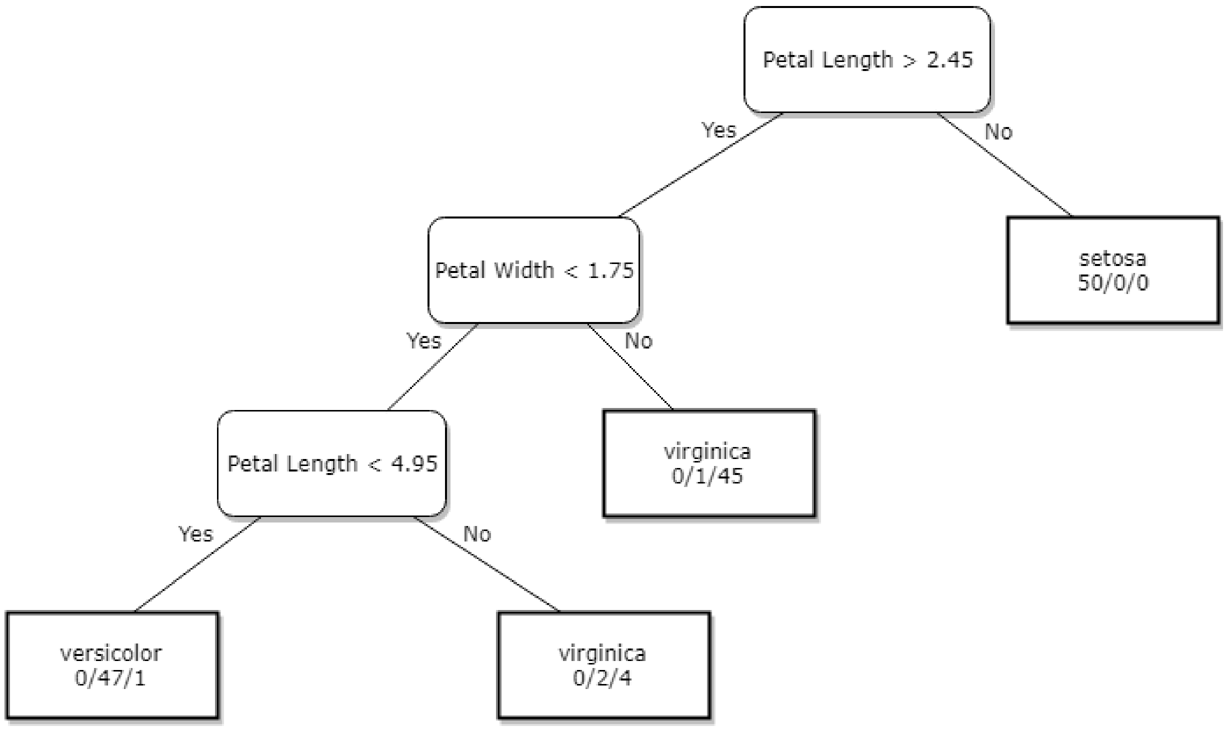


Figure 3.3.8: Example of the result of training a DT Classifier based on the Iris dataset [63] [18]. This dataset consists of three different species of iris plant. The upper figure present the decision boundaries that are produced by the classifier, each of the color represents different species of iris. The graph below, shows the nodes where the decisions are evaluated by the classifier. The decision boundaries plot was taken from [31].

Figure 3.3.8 presents the exemplary DT Classifier, that was trained in order to classify different spices of iris plant [63] [18]. It is visible that each node of the tree is composed of a single "yes or no" type question, and the prediction is calculated by recursively answering those questions (starting from the root node) using the information decoded within the input vector. One of the most critical features of this model is that a human easily interprets its prediction. One can get the intuition of why the model made a particular decision by looking at each question-answer pair and comparing them with an initial problem's domain knowledge. The DT models are very popular in the field of High Energy Physics.

3.3.3.2 Gradient Boosting

Gradient Boosting is an idea to enhance a single weak learner prediction by creating an ensemble of those models. The gradient boosted method creates a strong learner model by learning from the errors of the weak learners. Typically, the weak learner is selected to be a shallow DT or Logistic Regression model. Although all classification models can be used, the only regiment is that the weak learner needs to perform better than chance when trying to label the data.

Gradient Boosting constructs additive model by sub-sequentially fitting week learners to current pseudo residual at each iteration[12]. Mathematically, the Gradient Boosting algorithms minimize the cost function by approximating the optimal solution $f^*(x)$ using an additive expansion of the form:

$$f(\mathbf{x}_i) = \sum_{k=0}^{K} f_k(\mathbf{x}_i; w, q, T) \tag{3.26}$$

Where $f_k(\mathbf{x}; w, q, T)$ is a weak learner belonging to the class of regression trees, q represents the structure of each tree that maps from input vector $\mathbf{x}$ into the corresponding leaf index, T is a number of leaves in each tree and $w \in \mathcal{R}^T$ is a leaf weight. To learn the parameters of each function that constitutes the xgboost model, the following cost function is minimized:

$$\begin{aligned} \mathcal{L}_{xgboost}(f) &= \sum_{i=0}^{N} \mathcal{L}(y_i, f(\mathbf{x}_i)) + \sum_{k=0}^{K} \Omega(f_k) \\ &= \sum_{i=0}^{N} \mathcal{L}(y_i, f(\mathbf{x}_i)) + \gamma T + \frac{1}{2}\lambda ||w||^2 \end{aligned} \tag{3.27}$$

where $\mathcal{L}$ is a is a differentiable convex loss function [13] that measures the difference between the prediction $f(\vec{x_i})$ and the target y_i. The term Ω penalizes the complexity of the model and depends on two hyper-parameters. The first one γ is dedicated to reducing the number of leaves, and λ is $L2$ regularization parameter.

The optimization of the formula 3.27 is infeasible because it introduces functions as parameters, and that why using numerical methods such as Stochastic Gradient Descent is not an option since they op-

[12] one **iteration** is a process of adding one weak learner to the model

[13] Within the scope of this thesis only Cross-Entropy was tested, although section 4.15.2 provides description of an alternative metric.

erate on numerical vectors, not trees. Instead, the model is trained in an additive manner, each time a single decision tree is added to the model. Let $\hat{y}_i^{(t)}$ be the model's prediction of the i-th training instance at the t-th interaction, then this procedure can unfold:

$$\begin{aligned}
\hat{y}_i^{(0)} &= 0 \\
\hat{y}_i^{(1)} &= f_1(\mathbf{x}_i) = \hat{y}_i^{(0)} + f_1(\mathbf{x}_i) \\
\hat{y}_i^{(0)} &= f_1(\mathbf{x}_i) + f_2(\mathbf{x}_i) = \hat{y}_i^{(1)} + f_2(\mathbf{x}_i) \\
&\vdots \\
\hat{y}_i^{(t)} &= \sum_{k=1}^{t} f_k(\mathbf{x}_i) = \hat{y}_i^{(t-1)} + f_{(t-1)}(\mathbf{x}_i)
\end{aligned} \tag{3.28}$$

The cost function 3.27 can be expressed as:

$$\mathcal{L}_{xgboost}^{(t)} = \sum_{i=0}^{N} \mathcal{L}(y_i, \hat{y}_i^{(t-1)} + f_t(\mathbf{x}_i)) + \Omega(f_t) \tag{3.29}$$

Using the Taylor expansion formula for the objective, the loss function can be written as:

$$\begin{aligned}
\mathcal{L}_{xgboost}^{(t)} &\simeq \sum_{i=0}^{N} \left[\mathcal{L}(y_i, \hat{y}^{(t-1)}) + g_i f_t(\mathbf{x}_i) + \frac{1}{2} h_i f_t^2(\mathbf{x}_i) \right] + \Omega(f_t) \\
&\simeq \sum_{i=0}^{N} \left[g_i f_t(\mathbf{x}_i) + \frac{1}{2} h_i f_t^2(\mathbf{x}_i) \right] + \Omega(f_t)
\end{aligned} \tag{3.30}$$

where g_i and h_i are gradient and hessian defined as follow:

$$g_i = \left[\frac{\partial \mathcal{L}(y_i, \hat{y}^{(t-1)})}{\partial \hat{y}^{(t-1)}} \right] \tag{3.31}$$

$$h_i = \left[\frac{\partial^2 \mathcal{L}(y_i, \hat{y}^{(t-1)})}{\partial \hat{y}^{(t-1)}} \right] \tag{3.32}$$

Learning objective written in the form of 3.30 is very convenient from the perspective of the code implementation. To add customized loss function, one needs to provide the implementation of the gradient and hessian. The procedure, that minimize 3.30 is visualized in Figure 3.3.9.

XGBoost is a perfect combination of software and hardware optimization techniques to yield superior results using less computing resources in the shortest amount of time. The main features of this package are listed below:

- **Split finding approximate algorithm** To find the best split over a continuous feature space, data need to be sorted and fit entirely into memory. This may be a problem in the case of large datasets. An approximate algorithm is used for this. Candidate split points are proposed based on the

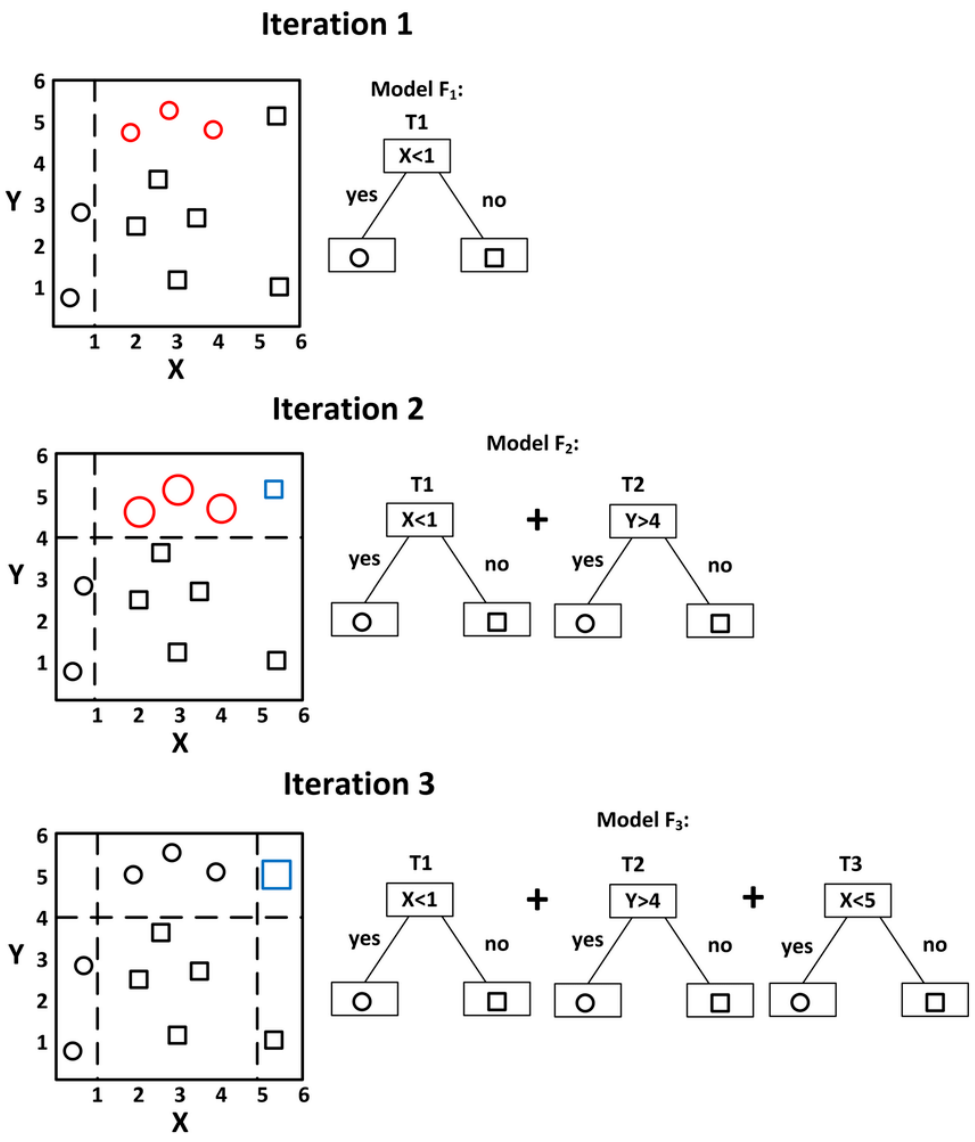


Figure 3.3.9: Gradient boosting algorithms visualization. Each iteration corresponds to adding one tree, which allows creating more sophisticated non-linear decision boundaries. This graph was taken from [141]

percentiles of the feature distributions.

- **Sparsity-aware algorithm**. The input may be sparse due to such reasons as one-hot encoding, missing values, or zero entries. XGBoost is aware of the sparsity pattern in the data and visits only the default direction (non-missing entries) in each node.

- **Out-of-core computation** Data that do not fit into main memory is divided into multiple blocks, and each block is stored on the disk. A dedicated algorithm can compress each block by columns and then decompress it on the fly.

- **Regularized Learning Objective** The default loss function is composed of two components

$$\mathcal{L}_{xgboost}(w) = \mathcal{L}(w) + \Omega(w) \tag{3.33}$$

  where $\Omega(w)$ is a regularization term and depends on the complexity of the model. Most of the implementations, including TMVA [83], miss this term and thus they are more prone to overtraining.

3.3.4 Deep Neural Network

This section provides a basic intuition on how the Deep Neural Network works. The one who wants to know more should refer to [74].

The critical limitation of linear models described in section 3.3.2 is a lack of ability to generate nonlinear decision boundaries. To overcome this limitation, a mapping $\phi(\vec{x})$ that transforms the input vectors into the representation that makes those vectors linearly separable has to be found. The most robust strategy of searching for such transformations is to construct a model that will be capable of learning $\phi(\vec{x})$. One of such a model is a neural network, which parametrizes $\phi(\vec{x};\theta)$ and uses the gradient-based optimization algorithm to find the optimal set of θ that corresponds to the desired representation. The idea of such a model comes from the brain's computation mechanism [105], which consists of computation units called neurons. The schematic view of such a model is presented in Figure 3.3.10.

A neuron is an entity that multiplies each input by its weight and sums them, afterwards applies a non-linear function to the result. The non-linear function is a crucial feature of the whole neural network idea, which is indicated in Figure 3.3.10 by the sigmoid shaped symbol. Its absence would not allow creating a more complex model that the logistic regression. The neurons are connected, forming a network, in which the output of a neuron is feed into the inputs following neurons. Neural network models arrange the neurons into layers, which reflect the flow of information. Within the scope of this thesis, only models built on top of fully-connected layers were investigated. Such a layer applies an affine transformation to the input vector [14]. Therefore, the output of the first layer can be written as:

$$h^{(1)} = \sigma^{(1)}(w^{(1)T}\vec{x} + b^{(1)}) \tag{3.34}$$

[14]Typically, that kind of layer is called linear, which can be misleading since it applies an affine transformation, not a linear one

and the output of the second layer is given by:

$$h^{(2)} = \sigma^{(2)}(w^{(2)T}h^{(1)} + b^{(2)}) \tag{3.35}$$

where $\sigma^{(i)}$ is an i-th activation function, $\vec{x} \in \mathcal{R}^{in}$, $w^{(i)} \in \mathcal{R}^{d_{i-1} \times d_i}$ are weights matrix, $b^{(i)} \in \mathcal{R}^{d_i}$ is a bias term and d_i is the number of neurons in the i-th layer.

Propagation of the input vector $\vec{x}$ through the network by iteratively calculating the output of each hidden layer to produce the prediction $\hat{y}$ is called **forward propagation**, see algorithm 3.3 for implementation details. Beside of providing the prediction, the full version of the forward propagation contains loss calculation step.

As indicated in Figure 3.3.10, the bottom layer, which has no incoming arrows, is the input to the model, where each neuron represents one component of the input vector $\vec{x}$. The top-most layer has no outgoing arrows. Thus, it is the output of the network. The layers situated between the input and the output are called **hidden**.

```
Require: Training data $x_i^{train}, y_i$;
Require: Network of depth $l$;
Require: $W^{(i)}$and$b^{(i)}, i \in \{1, ... l\}$ the weight matrices and bias vectors of the model
$h^0 = x_i^{train}$
for $k = 1, ..., l$ do
    $z^{(k)} = W^{(k)}h^{k-1} + b^{(k)}$
    $h^{(k)} = \sigma^{(k)}(z^{(k)})$
end for
$\hat{y} = h^l$
$\mathcal{L} = L(y_i, \hat{y}) + \lambda\omega(w, b)$
```

Algorithm 3.3: Forward propagation of feed-forward neural network

The non-linear function σ can be implemented in a various ways. The design of it is an extremely active research area. The two most popular types of activation functions are sigmoid, described in section 3.3.2, and rectified linear unit (ReLU) (both were used to train the model within the presented studies).

ReLU [71] it defined as follow:

$$ReLU(x) = max(0, x) = \begin{cases} x & x > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.36}$$

This activation function is the default activation function recommended for use inside hidden layer for most feed-forward neural networks [74]. Applying this function to the output of a linear transformation yields a non-linear transformation, which remains very close to linear. The only difference between a linear unit and a rectified linear unit is that a ReLU unit outputs zero across half its domain.

One of the essential features of this activation function is the fact that its derivative is not computationally expensive and remains large whenever the unit is active. On the other hand, the drawback of the ReLU units is that they cannot learn via gradient-based methods on examples for which their activation is zero. This issue was addressed by Leaky ReLU [103] and PReLU [79].

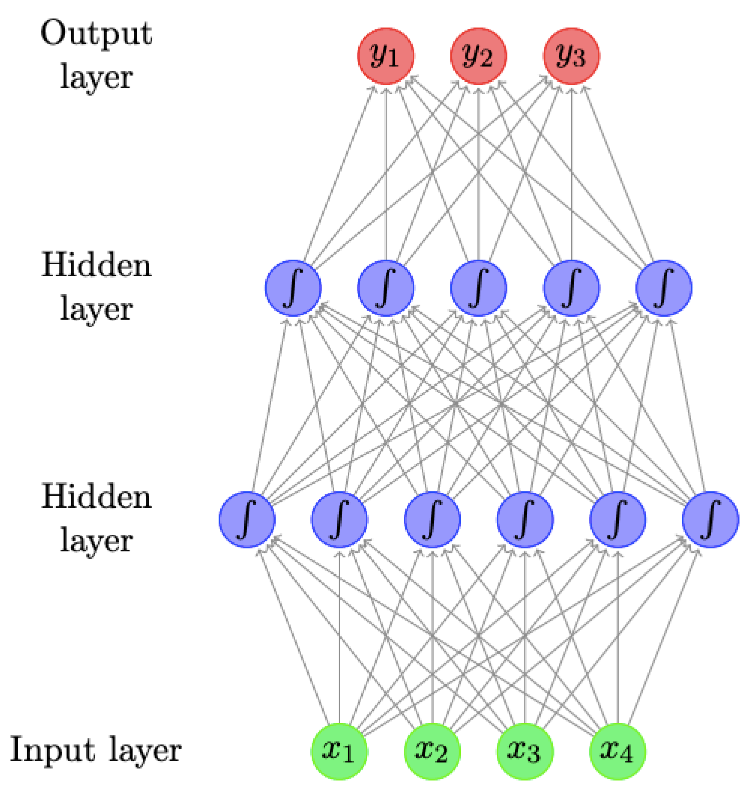


Figure 3.3.10: Feed-forward neural network with two hidden layers. Each circle represents a neuron, with incoming arrows being the neuron's inputs and outgoing arrows being the neuron's outputs. Each arrow carries a weight, reflecting its importance (not shown). Figure taken from [72].

3.3.5 Universal Approximation Theorem

One of the most important concepts of the neural networks with at least one hidden layer is the capability of approximating any Borel measurable function [15]. This observation is called the universal approximation theorem. The proof of this theorem is beyond the scope of this thesis and can be found in [84] and [56]. The intuition behind this theorem is shown in Figure 3.3.11. The theorem does not say how to construct the network; therefore the layer may be infeasibly large and may fail to learn and generalize correctly. From a practical perspective, it is recommended to use deeper (having more layers) rather than wider networks [107].

3.3.5.1 Back-propagation and computational graphs

The remainder of this section is dedicated to present the algorithm that is widely used to train the neural network model. It contains three steps:

- Forward propagation with loss calculation;
- Calculation of the loss function with respect to the model's weights;

[15]Within the scope of this thesis, a function is Borel measurable when it is continuous on a closed and bounded subset of $\mathcal{R}^n$

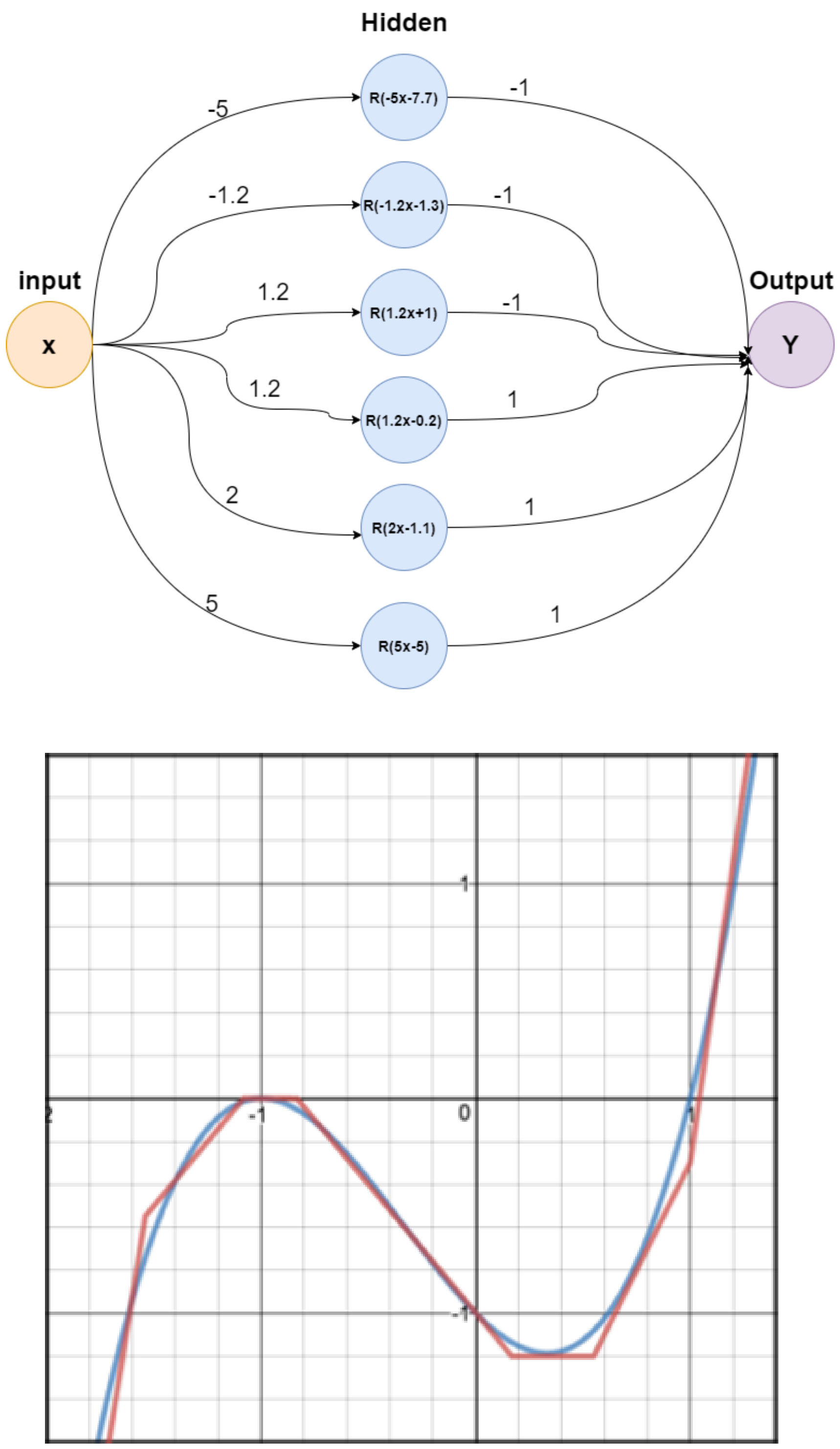


Figure 3.3.11: Visualization of the Universal Approximation Theorem. The objective was to build a network with one hidden layer that is able to represent $f(x) = x^3 + x^2 - x - 1$. This network is presented in the left plot. The right plot present the $f(x)$ and weighted sum of six ReLU functions. To reduce the approximation error more ReLU functions has to be added. Figure adapted from [65]

- Weight update via Gradient Descent-like algorithm using the previously calculated gradient.

This procedure was designed to iteratively adjust the weights of connected neurons using the cross-entropy between the network's output vector and the desired one. As a result of this procedure, the hidden units produce a new feature's representations, which are more convenient from the downstream task perspective. To estimate the change of weights for each layer, their gradient has to be calculated first. This algorithm is called **backpropagation** or backprop and was proposed in 1986 by G. Hinton [121]. The final round of evaluating a single step in training deep neural networks is to use the previously calculated gradients to update the model's weights using Gradient Descent-like algorithm.

```
Require: Training data $x_i^{train}, y_i$;
Require: Network of depth $l$;
Require: $W^{(i)}$ and $b^{(i)}, i \in \{1,\dots l\}$ the weight matrices and bias vectors of the model

run forward propagation for a given example $x_i^{train}$
compute the gradient on the output layer:
$g \leftarrow \nabla_{\hat{y}}\mathcal{L} = \nabla_{\hat{y}}\mathcal{L}(y, \hat{y})$
for $k = l-1, \dots, 1$ do
    convert the gradient on the $k$-th layer output into a gradient on the pre-$\sigma$ activation:
    $g \leftarrow_{z^{(k)}} \mathcal{L} = g \odot \sigma'(z^{(k)})$
    compute gradients on a weight and biases:
    $\nabla_{b^k}\mathcal{L} = g + \lambda\nabla_{b^k}\Omega(W, b)$
    $\nabla_{W^k}\mathcal{L} = gz^{(k-1)T} + \lambda\nabla_{W^k}\Omega(W, b)$
    propagate the gradients w.r.t the next lower-level hidden layer activation:
    $g \leftarrow_{z^{(k-1)}} \mathcal{L} = W^{(k)T}g$
end for
```

Algorithm 3.4: Backward propagation of feed-forward neural network

where $\odot$ is a Hadamard Product also known as element-wise product defined as:

$$(A \odot B)_{ij} = (A \odot B)_{ij} = (A)_{ij}(B)_{ij} \tag{3.37}$$

The backprop, which is demonstrated by the algorithm 3.4 is specified to the feed-forward neural network only. However, the modern deep learning framework, such as TensorFlow [8] or PyTorch [110], is based on a general form of backpropagation, which is implemented using computational graphs. A computational graph is a graph in which each node represents a variable. The variable may be a scalar, matrix, tensor [16] or variable of another type which is useful when calculating higher-order derivations. The edges of the graph represent the operations that are applied to particular nodes. The direction of the edge indicates whether some node is a parent of a given child node. One of the approaches to calculating the gradient is to extend the graph by adding additional nodes to the graph that provide a symbolic description of the desired derivatives. A detailed explanation of how the PyTorch implements gradient calculation can be found in [104].

From the practical perspective, the one who wants to build a model using the PyTorch framework has to define only how the forward propagation is calculated. The framework will take care of backward propagation.

[16]In the field of deep learning, a tensor is an object that has more than two dimensions and, in contrary to the tensor in physics, does not need to follow any transformation rules.

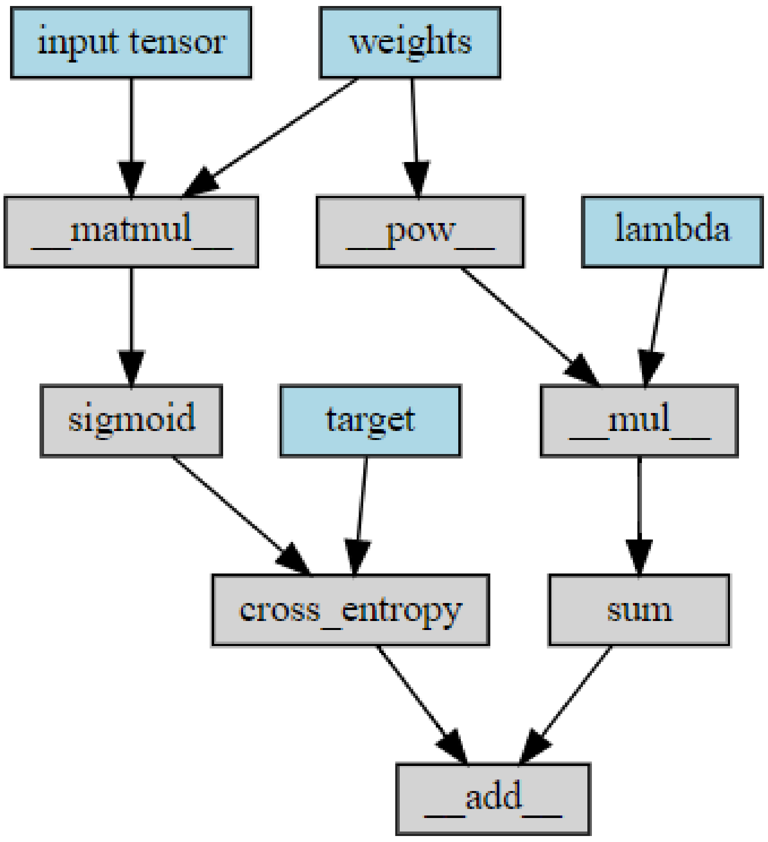


Figure 3.3.12: Computational graph used to perform the forward propagation and estimate the cross-entropy loss with L2 regularization for a single layer full-connected neural network. The blue rectangles represent tensors and the gray operations applied on those tensors. Figure generated by author's implementation of the autograd [58].

3.3.6 Hyper-parameter optimization

The model's hyper-parameter is a non-learnable parameter that has to be set priory to the whole training process. Each of the models has its own set of hyper-parameters, which depends on the model's architecture. For instance, one of the hyper-parameters of the Decision Tree, described in section 3.3.3.1 is its maximum depth, which indicates what the maximum number of splits used to construct the tree is.

One of the essential concepts regarding the hyper-parameters is that its influence on the model's performance is not equal. Some of the hyper-parameters are more important than the others. Additionally, the interaction between each of the hyper-parameters is usually unknown and complex.

Hyper-parameter optimization is a procedure of searching for a set of hyper-parameters to achieve high prediction accuracy. The wrong choice of the hyper-parameters may lead to a significant decrement of the classification power of the model. There are several hyper-parameter tuning techniques, but three are presented within the scope of this thesis. The difference among each of the method is a strategy to choose the set of S trial points $\{\lambda^{(1)}, \ldots, \lambda^{(S)}\}$ to evaluate $f(x; \lambda)$ in order to find the $\lambda^{(i)}$ parameters that work the best. Within the scope of this thesis, three methods are explained; see section 3.3.6.2 and 3.3.6.3.

3.3.6.1 Cross Validation

One of the methods to get more reliable measurement of the classifier performance is k-Fold cross-validation [33], which is schematically shown in Figure 3.3.13. This method randomly split the data into k independent subsets. For instance, if $k = 5$, 80% of data is used to train the model while 20% of the data sample constitutes the validation set.

Subsequently, one of the subsets is used to check the model's performance and the reaming data to train it. Each fold generates one prediction. That created set of scores can be used to estimate the statistical uncertainty of the classifier's performance. The cross-validation result is more robust and reliable than a single score calculated on a single test set.

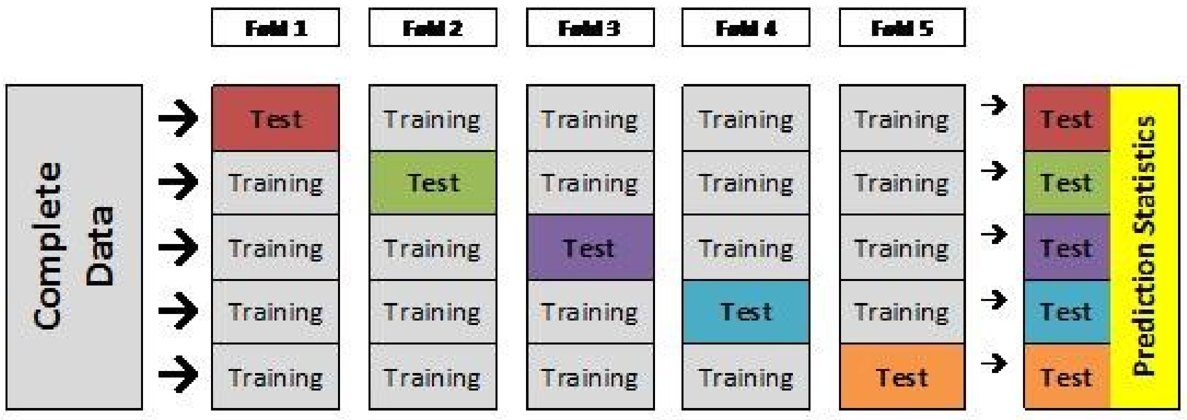


Figure 3.3.13: The visualization of k-Fold cross validation idea.

This technique is advantageous when deciding which set of hyper-parameters works best. The cross-validation provides a metric performance distribution rather than a single measurement. The drawback of this method is that for more complicated models, which training takes a lot of time, it becomes infeasible.

3.3.6.2 Grid and Random Search

Grid Search, except for manual search, is the most basic methodology to tune hyper-parameters. It splits the hyper-parameter space into rectangular hype-regions. Each of the regions represents one trial. In other words, Grid Search choose a set of values per each variable $(\lambda_1 \ldots \lambda_K)$, where K is a number of hyper-parameters, and then a set of trials is formed by assembling every combination of values for each hyper-parameter; therefore the number o trials in a Grid Search is:

$$S = \prod_{k=1}^{K} |\lambda_k| \tag{3.38}$$

This product over K hyper-parameters makes Grid Search suffer from the curse of dimensionality because the number of joint hyper-parameters values, that has to be checked grows exponentially with

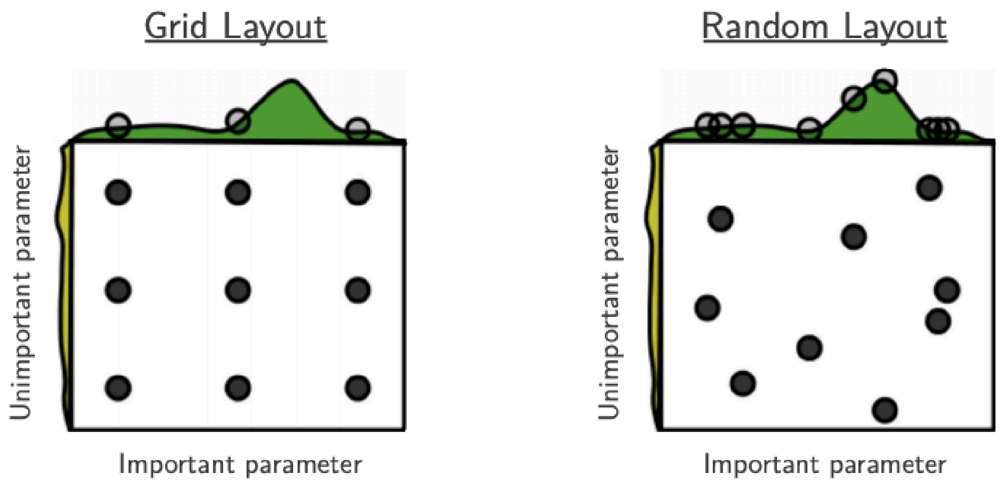


Figure 3.3.14: Comparison of the Grid Search and Random Search methods of hyper-parameters tuning. On both of the figures, the x-axis represents a hyper-parameter. The modification of its value has a negligible effect on model performance, and the y-axis presents a hyper-parameter that has a large impact on the model classification ability [27].

the number of hyper-parameters.

The only real difference between Grid Search and Random Search is on step 1 of the strategy cycle – Random Search picks the point randomly from the configuration space. The researcher needs to provide a set of distributions, one per hyper-parameter, to generate trials. From a practical perspective, the usual choice for these distributions is uniform and log-uniform distributions. The second one is a common choice when dealing with such hyper-parameter as the learning rate, where the hyper-parameter's value range that needs to be check is $\lambda_k \in (1 \times 10^{-6}, 1)$.

To compare these two methods a toy experiment was shown in Figure 3.3.14. It presented a search for nine trials for optimizing a function
$f(x, y) = g(x) + h(y) \approx g(x)$. With Grid Search, nine trials test $g(x)$ in three distinct places only, in comparison Random Search leveraged all nine trials to explore distinct value of $g(x)$. The failure of Grid Search is the rule rather than the exception in high dimensional hyper-parameter optimization [27].

3.3.6.3 Bayesian optimization

Both of the previously described methods have one drawback. Neither of them leverages the previous evaluation results to reject the regions where there is a small probability of finding the maximum value of the unknown objective function $f(x)$, which is a nonlinear function defined over a compact set $\mathcal{A}$. This section is dedicated to describing the general idea of Bayesian Optimization, which can be applied to any regression problems. It is not limited to the optimal hyper-parameter search. The enhancement to these methods should make use of accumulated observations $D_{1:t} = \{x_{1:t}, f(x_{1:t})\}$, where x_i is a ith sample, and $f(x_i)$ is a observation of the objective function at x_i. [17]. One of the methods that fulfil this

[17]In the case of hyper-parameters optimization, f is a specific metric used to measure model performance, within this thesis, the ROC AUC was selected, and x represents a vector of hyper-parameters

condition is Bayesian optimization.

Bayesian optimization is a technique widely used for solving the regression problems when function evaluation is expensive, and the derivative is usually unknown. Thus the usual optimization methods like Gradient Descent, are not applicable. The hyper-parameter optimization problem belongs to this area. To be more precise, in the case of seed classification model training takes a significant amount of time (in the case of XGboost, one trial takes more than an hour), and the model performance derivative with respect to, for instance, the tree depth is not defined. The method's name comes from the famous Bayes theorem. In the case of an optimization problem, the posterior distribution can be expressed in the following manner:

$$P(f|D_{1:t}) \propto P(D_{1:t}|f) \cdot P(f) \tag{3.39}$$

where $P(D_{1:t}|f)$ is a likelihood function, $D_{1:t}$ is a dataset of previous observation that contains t consecutive trials, $P(f)$ is a priory function that express one's belief on a objective function f, and $P(f|D_{1:t})$ is a posterior distribution. The assumption of the Bayesian optimization method is the function f has to be **Lipschitz-continuous**, which means that exist some constant C, such for all pairs $x_1, x_2 \in \mathcal{A}$ the following inequality holds:

$$||f(x_1) - f(x_2)|| \leq C||x_1 - x_2|| \tag{3.40}$$

where $|| \cdot ||$ denotes a norm.

The posterior probabilities capture updated beliefs about the unknown objective function. This may be interpreted as a single step of the Bayesian optimization procedure, which is an estimation of the objective function with a surrogate model. The surrogate modelling is a technique which builds the approximating model of the objective function using models that are cheap and fast to evaluate.

To sample efficiently trails x [18], Bayesian optimization has the second component - acquisition function $u(x|D_{1:t})$. This function is used to determine the next trial location x_{t+1}. The acquisition function should be able to find a trade-off between exploration (searching for a region where the surrogate function is very uncertain) and exploitation (searching for values of x where $f(x)$ is expected to be high), which is shown in Figure 3.3.16.

```
\label={alg:Bayesian Optimization}
for t = 1,2, .., do:
    Find the next sampling point $x_{t+1}$ by optimizing the acquisition function: $x_t = argmax_x u(x|D_{1:t})$
    Sample a possibly noisy objective function $y_{t+1} = f(x_{t+1}) + \varepsilon_{t+1}$
    Augment the data $D_{1:t+1} = \{D_{1:t}, (x_{t+1}, y_{t+1})\}$ and update the surrogate function
end for
```

Algorithm 3.5: Bayesian Optimization

3.3.6.3.1 Gaussian Process

A popular surrogate model for Bayesian optimization is a Gaussian process. A Gaussian Process is a random process where every point $x \in \mathcal{R}^d$, where d is a dimensionality of the optimization problem, for

[18]In the case of hyper-parameters optimization x corresponds to the vector of hyper-parameters $x = (\lambda_1, \lambda_2, \cdots)^T$

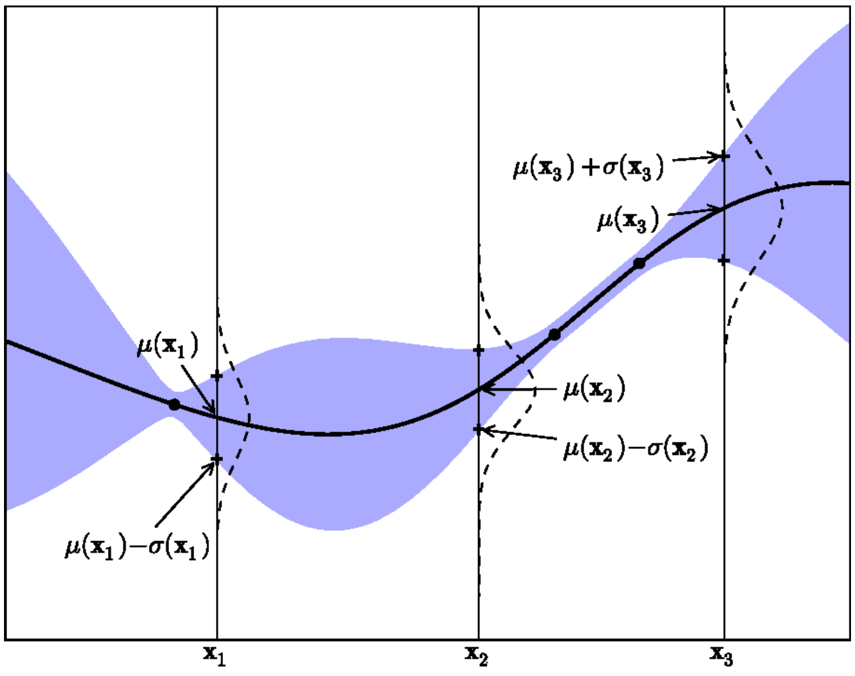


Figure 3.3.15: Simple 1D Gaussian process with three observations $x_{1:3}$. The solid black line is the mean prediction of the Gaussian Process, and the shaded area is one sigma region around mean value. Graph taken from [32].

the problem of hyper-parameter optimization d is equal to the number of hyper-parameters taken into consideration during the study, is assigned to a random variable f(x). The Gaussian Process can be interpreted as an extension of the multivariate Gaussian distribution to an infinite dimension or distribution over functions. For such a process, any finite combination of dimensions will be a Gaussian distribution. Gaussian Process is completely specified by its mean function m and k covariance function:

$$f(x) \sim \mathcal{N}\left(m(x), k(x, x')\right) \tag{3.41}$$

Intuitively, the Gaussian Process can be interpreted as a probability function that, for each value of x returns a mean and variance over all possible values of f at x, which is shown in Figure 3.3.15.

The convenient and common choice of mean function is $m(x) = 0$ [19]. The second building block of the Gaussian Process is the covariance function, also called kernel function. The literature discussed several choices of the kernel function, but within the scope of this thesis, only the most popular and basic one is discussed, i.e. squared exponential function:

$$k(x_i, x_j) = exp\left(-\frac{1}{2}||x_i - x_j||^2\right) \tag{3.42}$$

The function 3.42 approaches 0 when x_i and x_j get apart and 1 when they are very close to each other. The interpretation of the function is straightforward. When two points are close, they have a significant influence on each other. In the optimization task, the data come from an external model's evaluation

[19]Within the scope of this thesis the Gaussian process is limited to model a noise function in a form of $f(x) = g(x) + \varepsilon$ and $\varepsilon \sim \mathcal{N}(0, \sigma)$

and fits the Gaussian Process to get the posterior. In the hyper-parameters optimization case, the data refer to the model performance that was training using a particular set of hyper-parameters. Assuming that the data from the previous iterations is denoted as $\{(x_{1:t}, f_{1:t})\}$, the $f(t+1)$ is:

$$\begin{bmatrix} f_{1:t} \\ f_{t+1} \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} \mathbf{K} & \mathbf{k} \\ \mathbf{k}^T & k(x_{t+1}, x_{t+1}) \end{bmatrix}\right) \tag{3.43}$$

where

$$\mathbf{K} = \begin{bmatrix} k(x_1, x_1) & \dots & k(x_1, x_t) \\ \vdots & \ddots & \vdots \\ k(x_1, x_t) & \dots & k(x_t, x_t) \end{bmatrix} \tag{3.44}$$

and

$$\mathbf{k} = \begin{bmatrix} k(x_{t+1}, x_1) & k(x_{t+1}, x_2) & \dots & k(x_{t+1}, x_t) \end{bmatrix} \tag{3.45}$$

Formula 3.43 comes from the properties of the Gaussian Process, which states that the joint distribution of Gaussian distributed variables is also a Gaussian. Putting together equations 3.43 and 3.39 one can obtain the expression for the predictive distribution [115]:

$$P(f_{t+1}|D_{1:t,x_{t+1}}) \propto \mathcal{N}(\mu_t(x_{t+1}), \sigma_t^2(x_{t+1})) \tag{3.46}$$

where:

$$\mu_t(x_{t+1}) = \mathbf{k}^T\mathbf{K}^{-1}f_{1:t}$$
$$\sigma_t^2(x_{t+1}) = k(x_{t+1}, x_t) - \mathbf{k}^T\mathbf{K}^{-1}\mathbf{k}$$

To summarize Gaussian process allows leverage information obtained from the previous objective function evaluation.

3.3.6.3.2 Acquisition Function for Bayesian Optimization

This paragraph is dedicated to the second component of Bayesian Optimization, the acquisition function. The role of this object is to drive the search for the optimum solution, see the right side of Figure 3.3.16. It is defined in such a way that its high values correspond to the potentially high values of the objective function. The maximum acquisition function is used to select the next point at which the objective function is evaluated see right side of Figure 3.3.16. A typical choice of the acquisition function is Expected Improvement defined as:

$$EI(x) = \mathbf{E}\left(max(f(x) - f(\widetilde{x}), 0)\right) \tag{3.47}$$

where $\widetilde{x}$ is best location so far $\widetilde{x} = \underset{x_i \in x_{1:t}}{\text{argmax}} f(x_i)$.

The expected improvement can be evaluated analytically using the Gaussian Process model [115]:

$$EI(x) = \begin{cases} (\mu(x) - f(\widetilde{x}))\Phi(Z) + \sigma(x)\varphi(Z) & \text{if } \sigma(x) > 0 \\ 0 & \text{if } \sigma = 0 \end{cases} \tag{3.48}$$

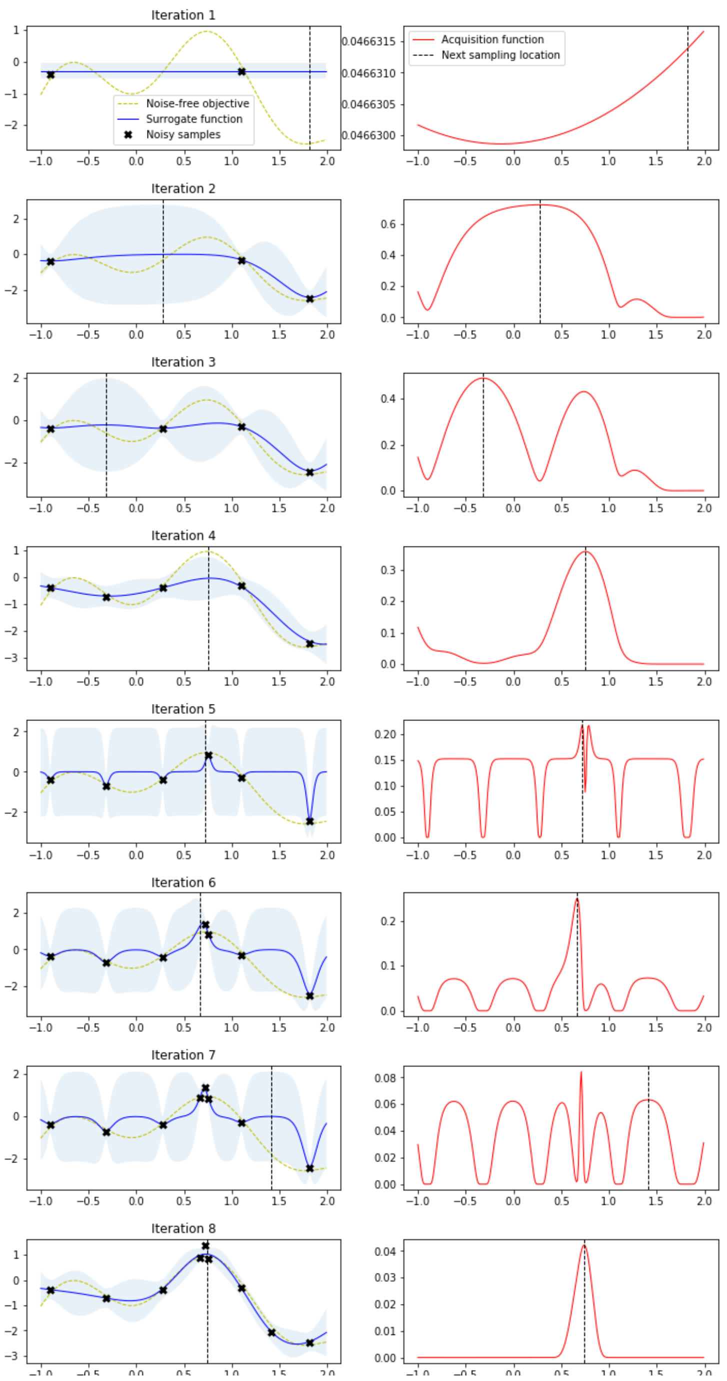


Figure 3.3.16: An example of using Bayesian optimization on a 1D optimization problem - finding maximum of function $f(x)$. The figures on the left side show a Gaussian process approximation of the unknown objective function (golden dashed curve) over ten iterations of the sampled value of the objective function. The figures on the right present the acquisition function (red curve). The acquisition function has a high value in the region where GP predicts a high objective, and the prediction uncertainty is high.

where $Z = \frac{\mu_x - f(\bar{x})}{\sigma(x)} - \xi$, $\mu(x)$ and $\sigma(x)$ are the mean and standard deviation of the Gaussian Process posterior predicted at x, Φ and φ are cumulative distribution function (CDF) and probability density function (PDF) of standard distribution respectively, ξ is the exploration-exploitation trade-off parameter, it is proportional to the amount, understood as a CPU-time or number of iterations, of exploration during optmization.

3.3.7 Model interpretation

One of the crucial problems when building [20] a complex Machine Learning model is lack of interpretability of its prediction. This fact raises the question of why one should trust the prediction that model gives. The Machine Learning model's interpretability is an active research area, and many ideas were recently published. To make sure that the model, that was trained to classify the track seeds provide reliable predictions two methods, apart from the usual quality metrics, were proposed: LIME (Local Interpretable Model-Agnostic Explanations) [116] and SHAP (Shapley Additive exPlanations) [100]. This section starts by presenting a brief overview of both methods. Section 4.12 resents the results obtained in order to understand the prediction of the trained seed classifier.

3.3.7.1 LIME - Local Interpretable Model-Agnostic Explanations

The discussion of the LIME should start from choosing models, that are human interpretable. An interpretable explanation needs to use a representation that is understandable to humans, regardless of the actual model's architectures or features used to make a decision. For instance, a possible interpretable representation of the text classification is a vector of binary values indicating the presence or absence of a certain word, even though the model may use more complex input features. To get a better intuition of what is the purpose of the LIME explanation, let consider a toy example where a Machine Learning model has to predict whether the patient has the flu or not. The input to this model is the patient's symptoms and other features such as weight or age. LIME selects the symptoms in the patient's history that are most relevant and led to the prediction. With these approximations and the domain knowledge, the doctor can make an informed decision about whether to trust the model's prediction or not. The visualization of this toy experiment is presented in Figure 3.3.17.

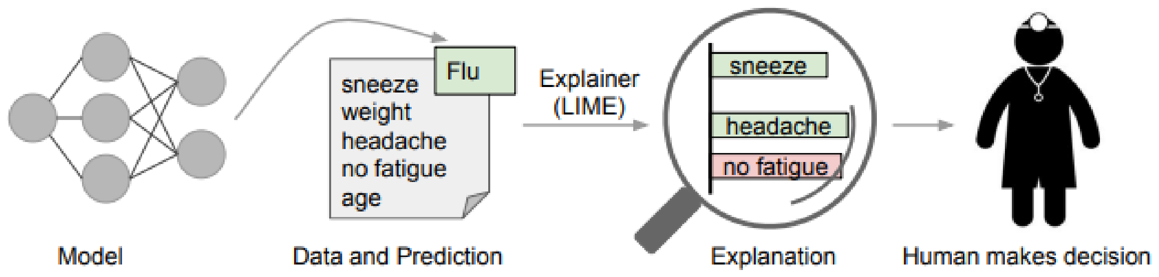


Figure 3.3.17: A toy example that visualizes the concept of LIME individual prediction explanation. A model predicts that a patient has the flu, and LIME highlights the symptoms in the patient's history that led to the prediction. Sneeze and headache are portrayed as contributing to the "flu" prediction, while "no fatigue" is evidence against it. Figure taken from [116].

In the case of the tabular data, the model that can be used as an interpretable approximation is a linear model [21]. That kind of explanation provides insights into the model for each studied event z. To

[20] in this context model building is understood as a complete model selection and validation processes

[21] the second type of interpretable model class is a decision tree, but within the scope of this thesis, only interpretation based on a linear model was studied

adopt this idea of explanation of the complex nonlinear decision boundaries, the LIME approximates the model response locally, around the point of interest. The point of interest is a single instance, e.g., one track seed that was classified as a true seed. Conceptually LIME approximation is very close to the Tylor series approximation, which infinitely differentiable function transforms to a power series around the specific point x_0:

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n \tag{3.49}$$

In the case of LIME the approximation is made by taking only the first order terms. To be more specific, let the model being explained is expressed as $f : \mathcal{R}^d \rightarrow \mathcal{R}$ and an explanation $g \in G$, where G is a class of interpretable human models. The $\pi_x(z)$ is a proximity measure kernel between an instance z to x, which is essential to define the local neighborhood of instance z.

Finally, let $\mathcal{L}(f, g, \pi_x(z))$ be a loss, which measures how faithful the local approximation g is. To reduce the complexity of the approximation model, the loss function is constructed by adding a regularization term that depends on a model complexity of $\Omega(g)$. For the linear model, the $\Omega(g)$ depends on a number of non-zero weights. Figure 3.3.18 visualize the idea of each component of the LIME explanation method. The explanation produced by LIME is obtained by finding the minimum of the following formula:

$$g^*(z) = \underset{g \in G}{\operatorname{argmin}}\ \mathcal{L}(f, g, \pi_x(z)) + \Omega(g) \tag{3.50}$$

The usual choice of $\Omega(g)$ is a K features LASSO regression [60]. Its detailed explanation can be find in section 3.3.2.0.2, equation 3.21. The weight of the explanation model represents the relative strength, measured as a influence on the model prediction for a given instance z, of each feature. The second reason why the Lasso regression is a usual choice is its ability to select significant features only, the weight of remaining are set to zero, see discussion in section 3.3.2.0.2.

From the practical perspective the LIME approximation is obtained by evaluating the fowling algorithm:

```
Require : Classifier $f$, $N$ data sample
Data : Instance $x$ and its interpretable version $x'$
Require : Similarity kernel $\pi_x(z)$
Require : Length of explanation $K$
$\mathcal{Z} \leftarrow \{\}$
for $i \in \{1, 2, 3, \ldots, N\}$ do :
    $z_i' \leftarrow$ sample_around($x'$)
    $\mathcal{Z} \leftarrow \mathcal{Z} \cup < z_i', f(z_i), \pi_x(z_i) >$
    end for
$w \leftarrow K - Lasso(\mathcal{Z}, K)$ with $z'(i)$ as features and f(z) as target
return $w$
```

Algorithm 3.6: Model explanation using LIME

3.3.7.2 Shap - SHapley Additive exPlanation

To understand the final method of analyzing the feature's importance, an explanation of the shapely value and the coalitional game theory need to be provided. The fowling paragraph was adopted from [132].

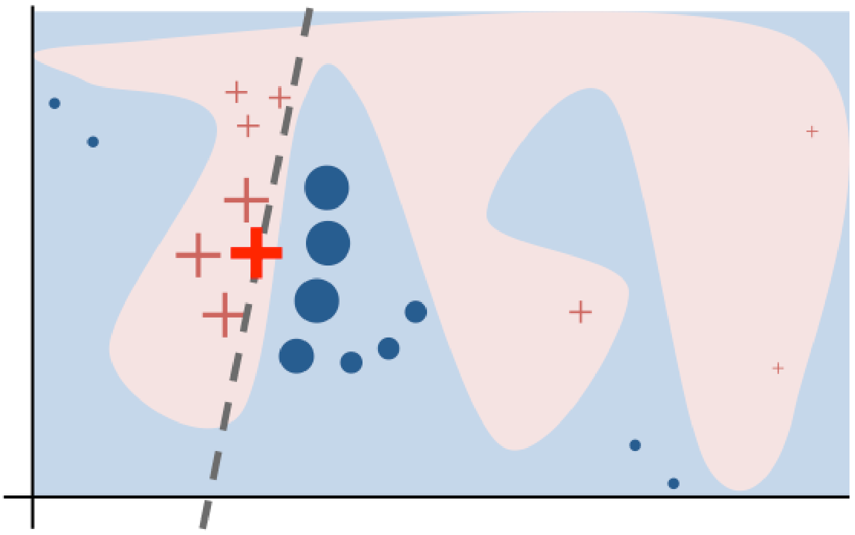


Figure 3.3.18: Toy example to present intuition for LIME. The black-box model's complex decision function f (unknown to LIME) is represented by the blue/pink background, which cannot be approximated well by a linear model. The bold red cross is the instance being explained. LIME samples instances gets predictions using f, and weighs them by the proximity to the instance being explained (represented here by size). The dashed line is the learned explanation that is locally (but not globally) faithful [116].

3.3.7.2.1 Coalitional game

The coalitional game is a tuple $< N, v >$, where $N = \{1, 2, \dots, n\}$ is a finite set of n players, $v : 2^N \to \mathcal{R}$ is a characteristic function, that describe worth of each coalition. The solution of the game is to find the operator $\varphi(v) = (\varphi_1, \dots \varphi_n) \in \mathcal{R}^n$, which assigns to $< N, v >$ vector of payoffs for each coalition participant. To find a fair solution operator $\varphi(v)$ needs to fulfill the following axioms:

- $\sum_{i \in N} \varphi_i(v) = v(N)$, where $v(N)$ is a value of grand coalition consisting of all players. This axiom is called *efficiency axiom*.
- if for two players i and j $v(S \cup \{i\}) = v(S \cup \{j\})$ holds for every S, where $S \subset N$ and $i, j \notin S$, then $\varphi_i(v) = \varphi_j(v)$. This axiom indicates symmetry property.
- The *dummy axiom* says if $v(S \cup \{i\}) = v(S)$ holds for every $S \subset N$ and $i \notin S$, then $\varphi_i(S) = 0$, and the player i is a dummy player, who has no influence on the coalition game outcome.
- for any pair of games $v, w : \varphi(w + v) = \varphi(w) + \varphi(v)$. This property is called additivity axiom.

Shapley theorem says, that for a given game $< N, v >$ exists a unique solution φ, which satisfies above axioms, and it is called the Sapley value:

$$\varphi_i^{shapley}(v) = \sum_{S \subseteq N, s=|S|} \frac{(n-s-1)!s!}{n!}(v(S \cup \{i\}) - v(S))) \qquad (3.51)$$

The proof of the above theorem can be found in the original Shapley's paper [127].

To illustrate, let us consider a United Nations Security Council example. The UN council consists of 5 permanent members, namely China, Russia, France, the UK, and the US, with veto power and ten non-permanent members. All permanent members must not veto, and the majority of members need to agree to pass the resolution. What is the Shapley value for each of the countries?

To simplify the example let us consider the game $N = \{1, 2, 3\}$. The player 1 is a permanent member with the veto power, and players 2 and 3 are non-permanent members. The value of this game is expressed: $v(\{1, 2\}) = v(\{1, 3\}) = v(\{1, 2, 3\}) = 1$ and $v(S) = 0$ otherwise. The Shapley value for this game obtained using formula 3.51 is $\phi_1 = \frac{2}{3}, \phi_2 = \frac{1}{6}, \phi_3 = \frac{1}{6}$. The conclusion is that both players 2 and 3 generate some value, so they deserve to get some reward and the player 1 gets the most of the reward, because it is the most important since no resolution can be made without it.

3.3.7.2.2 Shap method

To leverage the idea of the Shapley value for the problem of model explanation let assume that the explanation model g defined in the section 3.3.7.1 is a linear function of a binary variables:

$$g(z') = \varphi_0 + \sum_{i=1}^{M} \varphi_i z'_i \tag{3.52}$$

where $z' \in \{0, 1\}^M$ indicates whether the feature was observed ($z' = 1$) or unknown ($z' = 0$), and $\varphi_i \in \mathcal{R}$ are feature's value.

In order to evaluate the effect of missing features i on the model being explained f, it is necessary to define a mapping h_x, that each missing binary feature z' maps to the original function. Such a mapping allows calculating the effect $f(h_x(z'))$ of observing or not features. To compute the Shap values, the authors proposed [100] the following assumption:

$$f(h_x(z')) = \mathbf{E}[f(x)|x_S] \tag{3.53}$$

where S is the set of non-zero indexes in z', and $\mathbf{E}[f(x)|x_S]$ is the expected value of the function $f(x)$ conditioned on a subset S of the input features. Putting together equation 3.53 and 3.51 the shap value is calculated using the following formula:

$$\varphi_i = \sum_{S \subseteq N} \frac{(M - |S| - 1)!|S|!}{M!} (\mathbf{E}[f(x)|S \cup \{i\}] - \mathbf{E}[f(x)|S]) \tag{3.54}$$

Figure 3.3.19 visualizes the idea of model explanation and feature importance based on a Shap method.

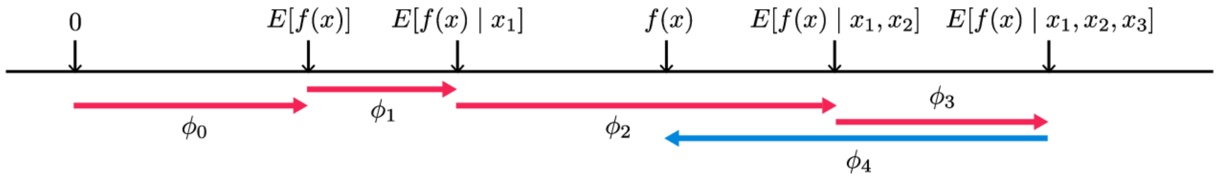


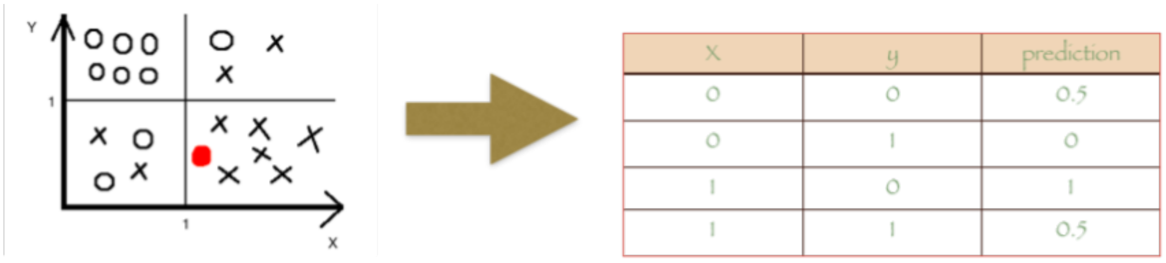
Figure 3.3.19: Shap values explain the output of a function f as a sum of the effect φ_i of each feature being introduced into a conditional expectation [101].

3.4 bonsai Boosted Decision Tree

Since the tracking algorithm is a part of the real-time LHCb High-Level Trigger system, both the execution time and memory footprint are important, so using the full continuous classifier is not an option. Instead, a binned BDT (called bonsai BDT or bBDT) classifier that meets the speed and memory criteria of the HLT is used. This section is dedicated to present the idea of the model's binarization, and the benefits drawback will be discussed. Similar idea was implemented within the LHCb HLT [69].

The idea is to replace the evaluation of each tree that constitutes the Gradient Boosted Classifier model [22] into an operation that is independent of the model's complexity. The best possible choice for such an operation would be the one that has a constant time computation complexity $O(1)$. A lookup table [23] is the right candidate. Thus it is a data structure that fulfils the above access time requirement. The remaining problem is how to convert a complex model into the lookup table. The solution to this problem is data discretization, which is a process of splitting an input space into k bins, and then for each of the combinations of the bins, the model's prediction is calculated, and the obtained value is saved.

To get a better understanding of how this process works, consider a 2D binary classification problem where "X" and "0" represent two possible classes, which are shown in Figure 3.4.1. For simplicity, let the number of bins $k = 2$, therefore in order to discretize the models, one needs to calculate four predictions, which is a number of bins times number of features. The outcome of this process is a table presented on the right side of Figure 3.4.1. In order to calculate the prediction for a given new instance x (denoted as a red dot), the classifier needs to find corresponding bin indices and extract value from the table. In this example, the prediction is 1.



x	y	prediction
0	0	0.5
0	1	0
1	0	1
1	1	0.5

Figure 3.4.1: bonsai Boosted Decision Tree idea visualization, see text for an explanation

[22]From the technical perspective gradient boosted model is very close to the big array of if-else statements
[23]The lookup table should be implemented as a hash table

Research is what I'm doing when I don't know what I'm doing

Wernher Von Braun

4 Machine learning based algorithm for long-lived particles reconstruction in LHCb

This chapter presents a Machine Learning based study related to the improvement of the Downstream Tracking algorithm. It starts by providing the general introduction to the LHCb track reconstruction methodology, and the Downstream Tracking algorithm is described, including a section focused on a seed classifier. Then, the performance of the entire Downstream Tracking algorithm is discussed. The final section is dedicated to present ideas on how to approach similar types of problems by leveraging more sophisticated Deep Learning models. Those models will be tested as a part of the development of the Downstream Tracking algorithm for the Upgraded LHCb experiment.

4.1 Track reconstruction

Tracking is one of the essential steps in the event reconstruction. This procedure is dedicated to reconstructing particle trajectories using a collection of hits that were created when a particle interacts with tracking stations. Tracks are essential to provide precision vertexing and momentum estimation. Without that information, no physics analysis can be performed. Moreover, track information is crucial for particle identification. In order to find a specie of particles the calorimeter clusters, muon station hits, and Cherenkov rings need associated tracks to be properly detected.

The tracking procedure consists of three major subroutines listed below:

- **Pattern Recognition**. Pattern recognition is the first step in the tracking reconstruction sequence, that is responsible for associating hits to a given particle. These algorithms have to deal with big combinatorics, which originates from the number of hits that are recorded by the tracking detectors.

The pattern recognition efficiency has vital importance. In order to reconstruct B meson decays, one has to reconstruct several tracks. As an example, let consider decay channel $B^0 \rightarrow K^{*0}(\rightarrow K^+\pi')\mu^+ + \mu'$, which is one of the most promising channels to detect a New Physics phenomena [50]. In this case, in order to accurately reconstruct B^0, meson four charged particles have to be reconstructed. Thus the efficiency of B meson reconstruction depends on the fourth-order on the track reconstruction efficiency. Moreover, Pattern recognition algorithms have very limited CPU time budget because they are executed as a part of software online trigger, for which timing is essential.

- **Track Fitting**. The tracks that were found by the pattern reconstruction are then fitted using a Kalman filter algorithm [87][66]. This method is capable of accounting multiple scattering and energy loss in the detector. This track reconstruction subroutine is the most CPU-time consuming because it has to perform the intense computation of material interaction and propagation through the magnetic field. The detailed description of how this method is utilized within the LHCb tracking can be found in [136].

- **Clone killing**. The final subroutine of track reconstruction is called **clone killing**. In this stage, tracks that have been reconstructed multiple times are eliminated. Those duplicated tracks originate from the redundant pattern recognition, when the algorithms create two tracks with similar hits content. The outcome of this step can be used for further physics analysis.

4.2 The LHCb track types

The tracks reconstructed in the LHCb detector are split into five categories. The splitting criterion is based on subdetectors in which they were reconstructed, which is shown in Figure 4.2.1.

The first category of the tracks is **VELO tracks**. That tracks are reconstructed using only hits measured in the Velo detector. Therefore, for this track, momentum measurement is impossible [1]. Nevertheless, Velo tracks also play an essential role to select isolated particles, for instance, when the analysis focuses on searching for a rare or forbidden decays.

The second type of track is **Upstream tracks**. They consist of Velo segments matched with hits in the TT detector. Those tracks are reconstructed to expand the detector acceptance for low momentum particles, which would otherwise be removed by the magnet. Moreover, because of the weak residual magnetic field in TT, the estimated momentum resolution for these tracks is worst than 10%.

T tracks are defined as those which have measurements only in the T stations. Despite the worst momentum resolution, estimated to be around 25%, they play a vital role in improving the purity of LHCB's photon reconstruction by identifying ECAL's clusters create by the electrons.

[1]As mentioned in section 1.3.1.1, in the Velo region there is no bending magnetic field, so the track are modeled as a straight lines

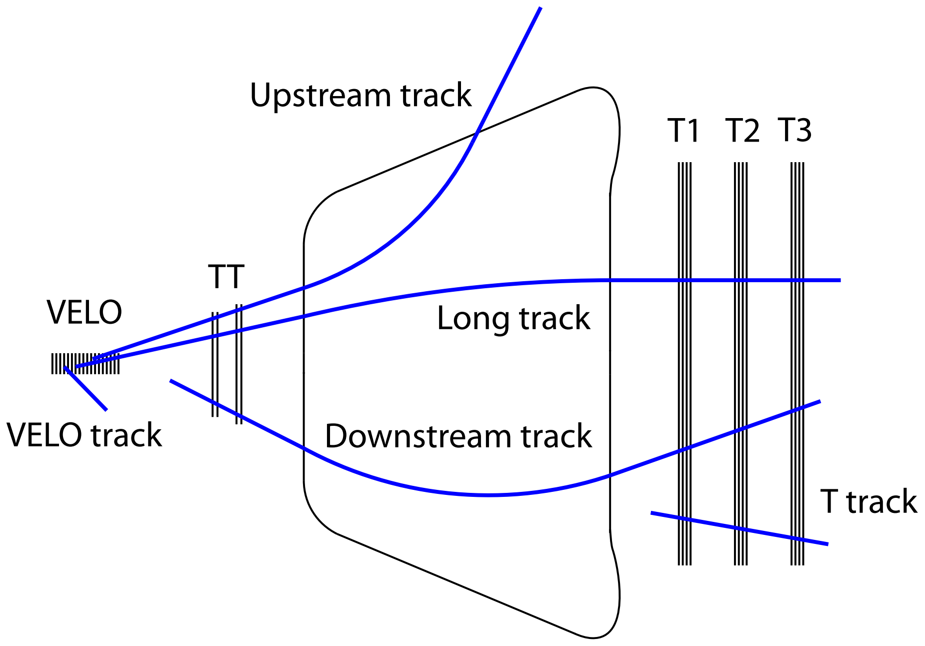


Figure 4.2.1: Track types available in the LHCb and corresponding subdetectors.

The most important category of the tracks is **Long tracks**, which contains those tracks originating in the vertex detector and traverse the full tracker acceptance, including passing through T and TT stations. These tracks provide the best momentum resolution for particles that traverse the full tracking detector and are used in the majority of LHCb analyses.

The final and the most significant, from this thesis perspective, group of tracks is **Downstream tracks**. The Downstream track contains T-segments matched to the hits in the TT detector. Those tracks are created by the daughters of the long-lived particles, which decays outside of the Velo detector. The reconstruction of the Downstream tracks allows to almost double the LHCb acceptance for long-lived particles. Those tracks are reconstructed by, the PatLongLivedTracking algorithm [57]. The accurate description of this algorithm is a topic of section 4.3.

Within the framework of LHCb experiment, a track is modeled as a series of straight line segments, known as track states. A track state is defined as a 5 dimensional vector of the form:

$$\vec{x} = \begin{pmatrix} x \\ y \\ t_x \\ t_y \\ \frac{p}{q} \end{pmatrix} \tag{4.1}$$

where: $t_x = \frac{\partial x}{\partial z}$ and $t_y = \frac{\partial y}{\partial z}$, p is a track momentum and q is a track's associated particle charge. It is worth to notice that LHCb uses the right-handed coordinate system, in which the $x - z$ is bending plane and the $y - z$ is non-bending one.

To visualize the complexity of tracking reconstruction problem Figure 4.2.2 presents a single event display, that was collected during Run II.

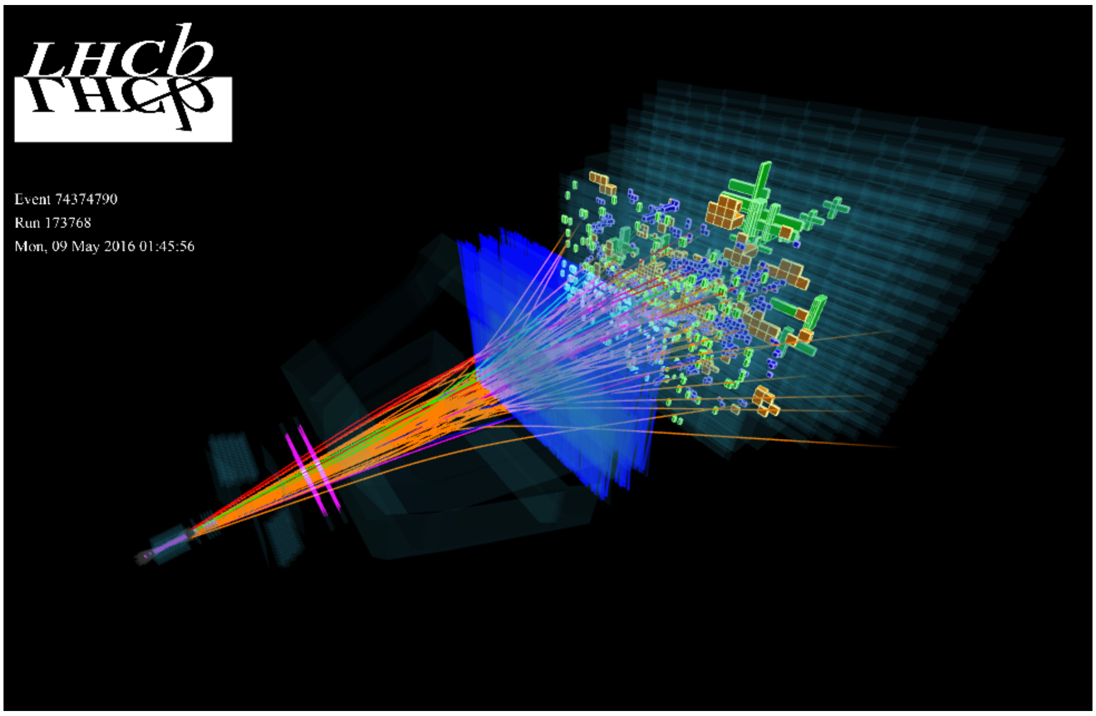


Figure 4.2.2: A typical LHCb event fully reconstructed during data taking on May 9th 2016 (during Run II). Particles identified as pions, kaon, etc. are shown in different colours. Figure taken from [96].

One of the analyses, which utilizes long-lived particles, is a study of time-dependent *CP*-violating asymmetries in $B_0 \rightarrow J/\psi K_S^0$ decays. Measurement of time-dependent *CP* asymmetries provides a valuable test of the Standard Model's flavor sector as well as an opportunity for discovery effects of physics beyond the Standard Model. The study of this asymmetry in the $B_0 \rightarrow J/\psi K_S^0$ decay mode provides a way to determine the effective *CP* phase. The *CP* violation asymmetry in $b \rightarrow c\bar{c}s$ decays of the B mesons is caused by the interference between mixed and unmixed decay amplitudes. A state initially prepared as a B^0 can directly decay into $J/\psi K_S^0$ or can oscillate into a $\bar{B}$ and then decays into $J/\psi\, K_S^0$. Taking the correction to the theoretical uncertainty, this amplitude is equal to twice the angle $\beta = arg[-\frac{V_{cd}V_{cb}^*}{V_{td}V_{tb}^*}]$ of the Unitary Triangle.

4.3 Downstream Tracking Algorithm overview

The description of the downstream tracking pattern recognition algorithm is based on [57].

Downstream tracks at LHCb are reconstructed in the following way. First, a standalone track reconstruction in the T stations with an algorithm called PatSeeding [41] is performed, creating T tracks, also

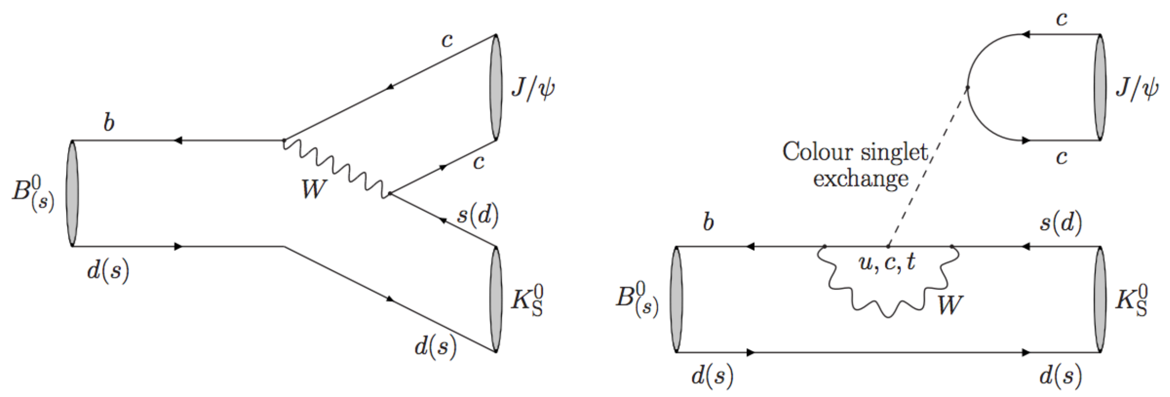


Figure 4.2.3: The Feynman diagram presenting the decay topologies contributing to the $B_0 \rightarrow J/\psi K^0_S$ channel: (left) tree diagram and (right) penguin diagram.

called **T-Seeds**. Roughly speaking, the downstream tracking algorithm's goal is to add TT hits that are likely to match those T-Seeds.

Those T-Seeds are filtered by a machine learning classifier to discard bad candidates which are tracks that do not represent the trajectory of a real particle. Accepted T-Seeds are then propagated back through the magnet to the TT station. Then the pattern recognition algorithm searches for clusters in the two x-layers of the TT detector. Those clusters already allow constraining the flight path of the particle with good precision. Then a cluster in the other x layer is searched for. Finally, clusters in u and v layers are added. Those selected clusters are then fitted to a parabola model using a χ^2 minimization technique [2]. The χ^2 test is defined as:

$$\chi^2(a) = \sum_{i=1}^{n} \frac{(f(x_i, a) - x_i)^2}{\sigma_i^2} \tag{4.2}$$

where a is a vector of free parameters being fitted, $f(x_i; a)$ is a model, and σ_i are the uncertainties in the individual measurement x_i.

To reduce spilover influence on a tracking reconstruction, a flag is set for each cluster if it is likely to have been created in a different collision than the current one. This flag is called a high-threshold bit. Tracks with a large number of high-threshold clusters are rejected. Finally, the best track is chosen according to the output of another multivariate classifier.

4.4 Downstream Track model

4.4.1 Propagation through the magnetic field

Charged particles traversing through the tracking stations, which are located outside of the magnetic field, follow a straight-line path. The bending of their trajectory inside the magnetic field can be approx-

[2]Fitting a model to the data involves construction of a test statistic, designated to measure the goodness of fit, and the model that for a given data minimize beforementioned statistic

imated by a sharp kink in the flight path at a given z position of a magnet point, called z_{mag}. The kink method is visualized in Figure 4.4.1. This z position depends on the parameters of the track and is also affected by inhomogeneities in the magnetic field. The numerical value of this position can be obtained using the parametrized empirical formula:

$$\begin{aligned} z_{\text{mag}} &= a_0 + a_1 \cdot t_y^2 + a_2 \cdot t_x^2 + a_3 \cdot 1/p \\ &\quad + a_4 \cdot |x_T| + a_5 \cdot |y_T| + a_6 \cdot |t_y|. \end{aligned} \tag{4.3}$$

where a are free parameters, t_x and t_y are the slopes of the last track state in the T stations. The absolute momentum, p, is estimated from the T track, using a simplified parametrisation that assumes a kink of the trajectory at the center of the magnet and the track to come from the point of origin. The observables x_T and y_T are the x and y positions of the last state in the T stations, respectively. Analyzing Equation 4.3, one can find that the variable z_{mag} depends mostly on the values in the first line, while the dependence on the ones from the second row are smaller.

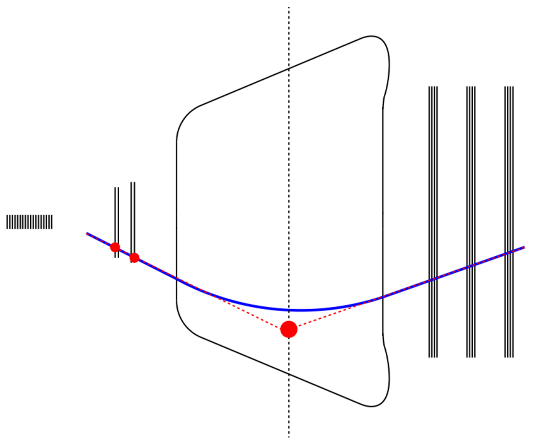


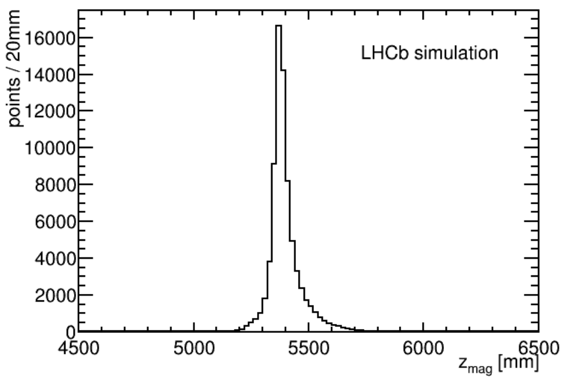
Figure 4.4.1: Sketch of the LHCb detector with the tracking system in the x-z plane. A downstream track (blue line) and its approximation outside the magnetic field (red dotted line) is shown. In the approximation the track undergoes a sharp kink at the magnet point (big red dot).

The a parameters are determined in using Monte Carlo simulations. Those numbers are obtained by fitting a straight line to the true position of the hits in TT, and a third-order polynomial to the true position of the hits in the T stations. The crossing point of both curves in the x-z projection determines the **true** value of z_{mag}. An illustration of z_{mag} determined in described way, and the difference between the values obtained with Equation 4.3 when using measured instead of simulated quantities are shown in Figure 4.4.2.

The x position of the magnet point is then determined by using a linear extrapoltion of the state in the T stations to z_{mag}.

$$x_{\text{mag}} = x_T + t_x \cdot (z_{\text{mag}} - z_T) \tag{4.4}$$

To correct the effect of magnetic filed on estimation of the y position, an additional correction is



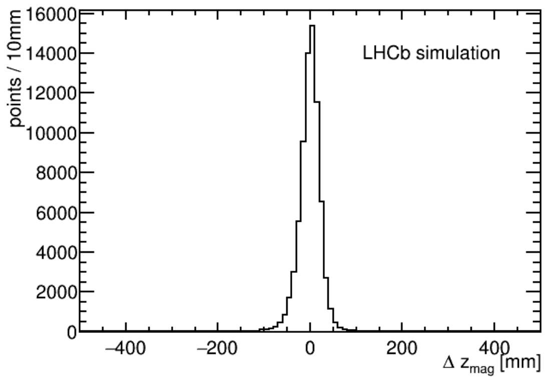


Figure 4.4.2: Left: **true** z position of the magnet point. Right: Difference between true and estimated z position of the magnet point. For the true position, simulated observables, for the estimated position measured observables were used.

applied:

$$y_{\text{mag}} = y_T + t_y \cdot (z_{\text{mag}} - z_T) - \beta_1 \cdot t_y \cdot \Delta^2_{\text{slope}} \tag{4.5}$$

where Δ_{slope} is the difference of the slopes in x before and after the magnet, defined as:

$$\Delta_{\text{slope}} = \left| x_{\text{mag}}/z_{\text{mag}} - t_x \right| , \tag{4.6}$$

and β_1 an empirical parameter.

The predicted slope in y in the TT is calculated as:

$$t_{y,TT} = t_y \cdot (1 + \beta_0 |t_y| \Delta^2_{\text{slope}}) \tag{4.7}$$

with β_0 an empirical parameter, determined on simulation, using regression with true observables, to correct for the effect of the bending in the magnetic field on the slope in y.

The track that constitutes of OT hits only requires a dedicated procedure since y resolution, and the resolution of the slope in y is rather poor. Thus, the following parametrization is used instead:

$$y_{TT} \quad = \quad y_T + t_y \cdot (z_{\text{mag}} - z_T) + t_{y,TT} \cdot (z_{TT} - z_{\text{mag}}) - \beta_1 \cdot t_y \cdot \Delta^2_{\text{slope}}, \tag{4.8}$$

where β_1 is determined on simulated data using regression with true observables. The current implementation of the algorithm does not apply any further track correction logic, therefore the slope in the T stations can be calculated as:

$$t_{y,\text{constr}} = y_T/(z_T + (\beta_0 |t_y| z_{\text{mag}} + \beta_1) \Delta^2_{\text{slope}}) \tag{4.9}$$

For tracks with hits in the OT only, corrected $t_{y,\text{constr}}$ is used in Equation 4.5 instead of the measured value t_y.

Determination of the magnet point is crucial since it is used as an additional constraint in the downstream track χ^2 fit, see section 4.5.6. Therefore, its uncertainty has to be determined and those are calculated by estimating the difference between the values of x_{mag} and y_{mag} obtained using simulated quantities and reconstructed quantities for the extrapolation. This uncertainty depends on Δ_{slope} and the types of hits that constitute the track. The following empirical parametrisations of the uncertainty were used:

$$\begin{aligned}
\sigma_{x,OT} &= (2.0 + 18.0 \cdot \Delta_{\text{slope}})mm \\
\sigma_{y,OT} &= (5.0 + 20.0 \cdot \Delta_{\text{slope}} + 50 \cdot \Delta^2_{\text{slope}})mm \\
\sigma_{x,IT} &= (1.0 + 16.0 \cdot \Delta_{\text{slope}})mm \\
\sigma_{y,IT} &= (2.0 + 15 \cdot \Delta_{\text{slope}})mm
\end{aligned}$$

The numerical values of those parameters were obtained by fitting residual distributions.

4.4.2 Determination of the momentum

The momentum of a downstream track mainly depends on the kink it undergoes in the magnetic field, but also shows a dependence on the slopes in x and y. The following parameterisation results in a momentum resolution of about 2% averaged over the full momentum spectrum.

$$p = \frac{\gamma_0 + \gamma_1 \cdot t_x^2 + \gamma_2 \cdot t_y^2}{\Delta_{\text{slope}}}, \tag{4.10}$$

where again the parameters γ_i were determined using the true position of the hits in simulation. This resolution contains both the effect from the detector resolution and from dependencies which are not accounted for in the parametrization. The momentum resolution is presented in Figure 4.4.3.

4.4.3 Track model in the TT

The downstream track is modeled as a parabola in the TT. This representation comes from the presence of a strong residual magnetic field in and before the TT, which deflects low-momentum particles from a straight-line trajectory. The track is modeled by the following function:

$$x(z) = x_0 + t_x \cdot (z - z_{\text{mag}}) + c \cdot (z - z_{TT})^2, \tag{4.11}$$

with z_{TT} the z position in the middle of the TT, z_{mag} the z position of the magnet point, t_x the slope in x in TT, and $c = 1.48 \cdot 10^{-5} \cdot \Delta_{\text{slope}}$. The initial value for x_0 is x_{mag}.

The parameter c is determined on simulation by fitting the true position of the hits in TT with a parabola and determining it as a function of Δ_{slope}. The deflection from a straight line is tiny. For instance, a track with a momentum of $3GeV/c^2$, is deflected of about $200\mu m$, which is roughly the strip pitch of a TT sensor.

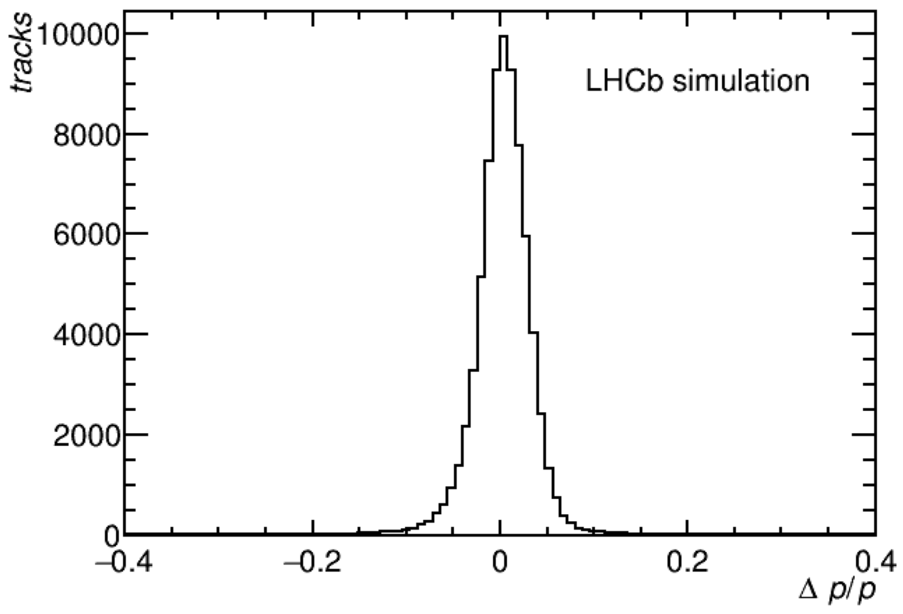


Figure 4.4.3: Momentum resolution for the initial track estimate. It amounts to about 2%.

The y position of the track is modeled as a straight line:

$$y(z) = y_0 + t_{y,TT} \cdot (z - z_{\text{mag}}), \tag{4.12}$$

where $t_{y,TT}$ is the track's slope in y given by Equation 4.11, and the initial value for y_0 is y_{mag}.

4.5 Pattern Recognition

This section provide an overview of the stages of the pattern recognition in PatLongLivedTracking, which are presented in the form if flow diagram in Figure 4.5.1. The detailed description on the first step, filtering T-Seeds, was moved into the separate section 4.6.

4.5.1 T-seeds reconstruction

The algorithm, that performs pattern recognition for the T-Seeds is called PatSeeding [41]. It consists of five distinct steps:

- hit preparation;
- track search per detector region;
- track search for tracks migrating from IT to OT;
- track search per Outer Tracker region for tracks at large $|y|$;

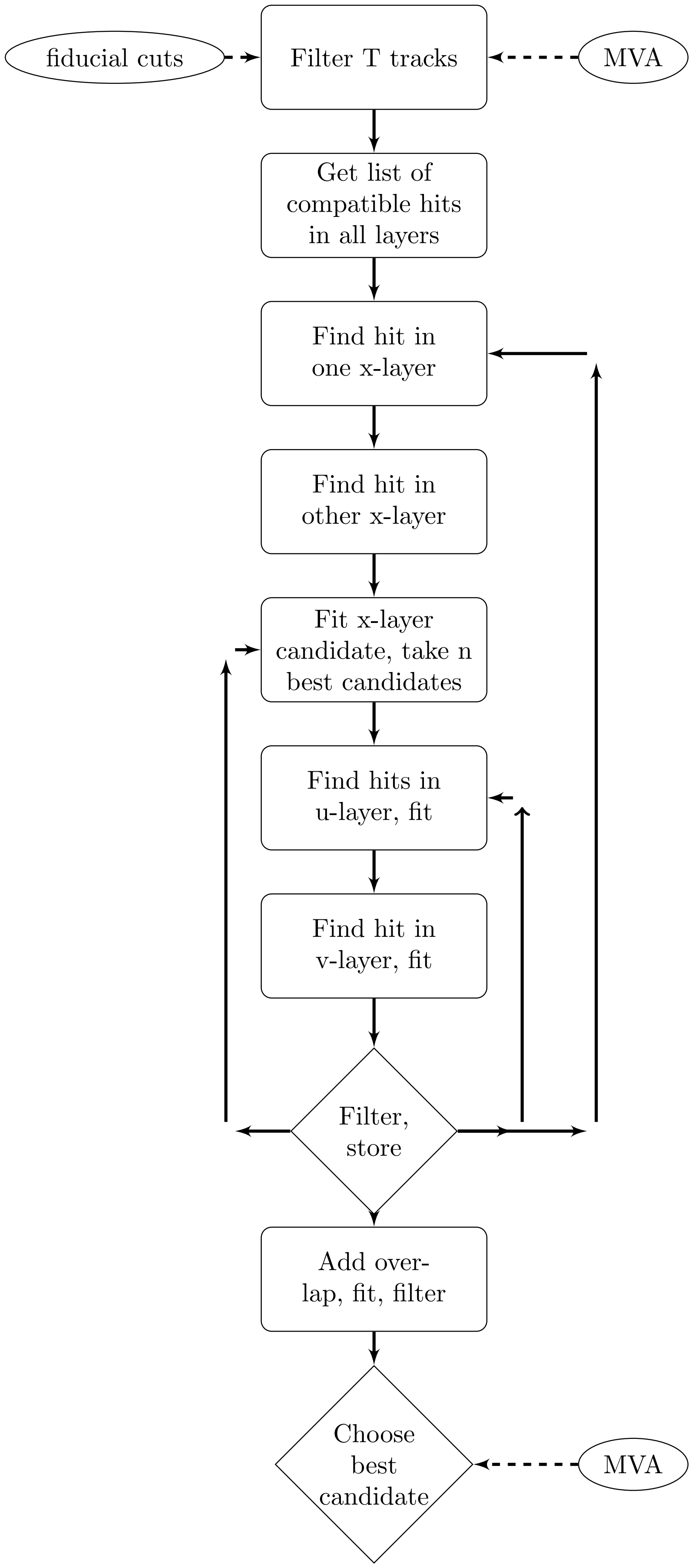

fiducial cuts
Filter T tracks
MVA
Get list of compatible hits in all layers
Find hit in one x-layer
Find hit in other x-layer
Fit x-layer candidate, take n best candidates
Find hits in u-layer, fit
Find hit in v-layer, fit
Filter, store
Add over-lap, fit, filter
Choose best candidate
MVA

Figure 4.5.1: Different stages of pattern recognition within the PatLongLivedTracking algorithm.

- validation of low-quality tracks left over from the per region search;

Within this algorithm, the track is described as a straight line in $x - y$ plane. For $x - z$ projection, a cubic model with three parameters is used:

$$\begin{aligned} x(z) &= c(z - z_{reference})^2(1 + d_{ratio}(z - z_{reference})) + b(z - z_{reference}) + a && (4.13) \\ y(z) &= b_y z + a_y && (4.14) \end{aligned}$$

d_{ratio} describes the correlation between the quadratic and cubic terms and is determined from Monte Carlo studies. The shift by $z_{reference}$ makes the fit more numerically stable ($z_{reference}$ is in the middle of the T stations, by default).

4.5.2 Search for compatible hits

In this initial reconstruction stage, the downstream track candidates are modeled by the straight line from the LHCb origin point (0,0,0) to the magnet point, which is the only constrain. This model is not always accurate since most particles of interest do not originate from the global LHCb origin (0,0,0), thus a correction to the x position of the search window is applied. The value of the search window is given by:

$$\delta x = \text{sign}(p \cdot \text{magPol}) \cdot (\texttt{XCorrectionConst}/p + \texttt{XCorrectionOffset})\,, \qquad (4.15)$$

with magPol the magnet polarity. This correction is shown in Figure 4.5.2, left.

Furthermore, a similar search window technique was applied to momentum-dependent search, which can be seen in Figure 4.5.2. The dependence of the window size on the momentum is given by:

$$\Delta x = \texttt{XPredTolConst}/p + \texttt{XPredTolOffset}, \qquad (4.16)$$

where p is the absolute value of the momentum. The parameters determining the size of the search window, `XPredTolConst` and `XPredTolOffset`, were derived on simulation, illustrated on the right of Figure 4.5.2.

Finally, the pattern reconstruction algorithm checks whether the hit is compatible with the expected y-position in the TT with a tolerance `YTol`. As the position of a hit in the TT can only be determined up to the length of a sensor module, this provides only a weak constraint. The x positions of the hits in the stereo layers are then updated, assuming the y position from the T track extrapolation.

All hits in all four layers which lie inside these tolerances are then stored in a container, and sorted according to the projection distance, this is the absolute distance of the hit to the predicted position.

The T track is not considered if less than three hits in the TT are found in total, or the x or the stereo layers do not contain any hit.

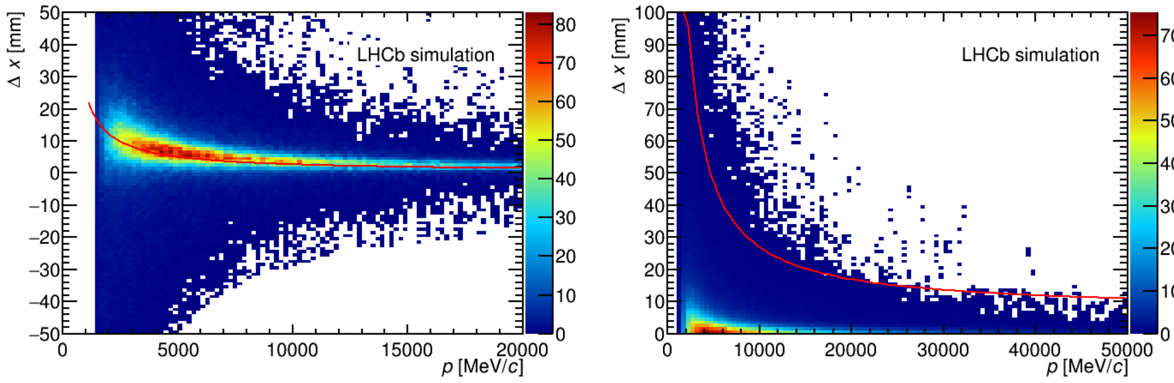


Figure 4.5.2: Left: Distance between extrapolated track (constrained to come from the LHCb origin (0,0,0)) and (truth-matched) hits before correction. The red line represents the correction function which is applied. Right: Distance between extrapolated track and (truth-matched) hits as a function of the momentum of the track for the preselection step, after the correction. The cut is show as the red line.

4.5.3 Search for hits in x layers

In this step, the algorithm iterates over all hits preselected by the previous step hits in the x layers to form preliminary track candidates.

The first hit that is added to the track candidate is chosen in one of the two x layers. This allows for a more precise determination of the slope of the track and the curvature, and therefore also of the momentum of the partially reconstructed downstream track.

Next, corresponding hits in the other x layer are searched for. The search window in the other x layer is defined as follows:

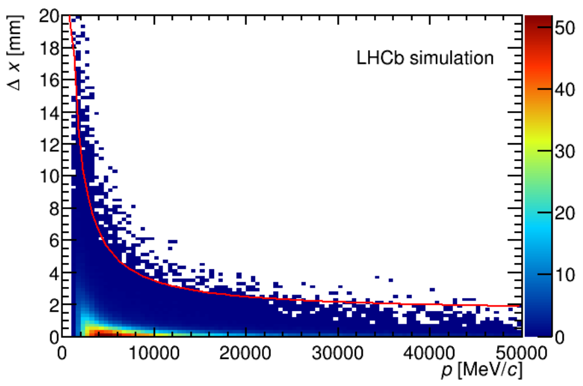
$$\Delta x = \texttt{TolMatchConst}/p + \texttt{TolMatchOffset}, \tag{4.17}$$

if Δx is smaller than a given value `MaxWindowSize`, otherwise `MaxWindowSize` is taken as the tolerance. This serves as a sanity check to exclude unphysically large values of the window size for low momentum particles. All hits within this window are then considered for further processing. As illustrated in the left hand plot of Figure 4.5.3, essentially all hits from true particles in simulation are enclosed within this region.

A χ^2/ndf fit to the x coordinate is then performed for each possible candidate, consisting of the first hit in the x layer, and one of the matching hits in the other x layer. The magnet point is used as a further point to add enough degrees of freedom for the fit. All track candidates are then sorted according to their χ^2/ndf value, and track candidates are discarded if the χ^2/ndf value is above:

$$\chi^2/\mathrm{ndf}_{\mathrm{max,\,x\,hits}} = \texttt{FitXProjChi2Const}/p + \texttt{FitXProjChi2Offset}. \tag{4.18}$$

If no hit could be found in the other x layer, the track candidate is not rejected. It is kept without fit to search for hits in the u layer. This allows for hit inefficiencies in the TT.



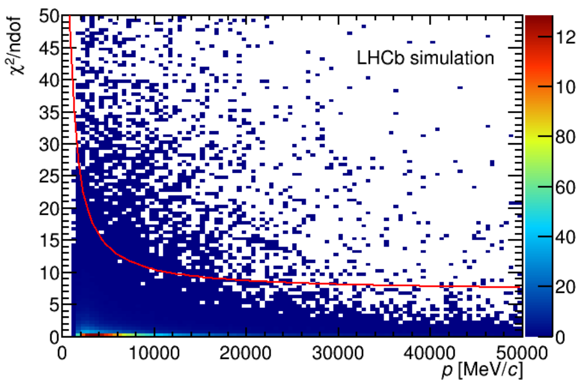


Figure 4.5.3: Left: Distance between extrapolated track with one x hit and (truth-matched) hit in the other x layer. The red line represents the cut on the distance. Right: χ^2 values for a fit to the x hits (truth-matched), with the cut represented as a red line.

Due to the large combinatorial background, there can be many ghosts in this selection. Track candidates with only two x hits are prone to be ghost tracks, and it is possible that the true combination of hits does not have the smallest χ^2/ndf. Therefore, in the next steps, the first `MaxXTracks` candidate has considered whose χ^2/ndf value is within the range (`MaxChi2DistXTracks`) to the lowest χ^2/ndf value.

In future studies, this section can be enhanced by training a machine learning model to select the best matching hits.

4.5.4 Search for hits in the u layer

The search window in the u layer can be smaller than those in the x layers, since the parameters of a track with two hits are reasonably well constrained. Hits are searched around the track extrapolated to the u layer, where the x position of the hit is updated by using the extrapolated y position. All hits within a search window are considered, where the window size is defined as

$$\Delta x = \texttt{TolUConst}/p + \texttt{TolUOffset}. \tag{4.19}$$

The parameters `TolUConst` and `TolUOffset` are determined from simulation, visualized in the left hand plot of Figure 4.5.4. The hits inside the window are then sorted according to their distance between the extrapolation of the track and their actual position. For each hit which passes this tolerance, a new track candidate is formed, until the maximum number of xu tracks (`MaxXUTracks`) is reached. Each of these candidate is then fitted with a χ^2 fit to obtain the best parameters for the final search in the v layer.

4.5.5 Search for hits in the v layer

The last step in the hit search subroutine is searching for the hits in the v. If the track's candidate already has three hits, a momentum dependent window is created according to the following formula:

$$\Delta x = \texttt{TolVConst}/p + \texttt{TolVOffset}. \tag{4.20}$$

Figure 4.5.4 (right) ilustrate this this search window. In case the track only contains two hits, Formula 4.19 is used to determine the size of the search window. The hit that is closest to the extrapolation is added.

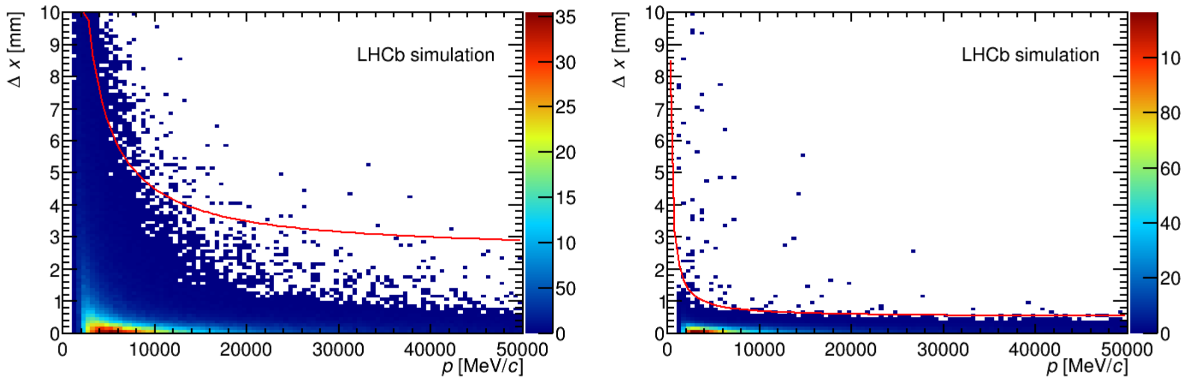


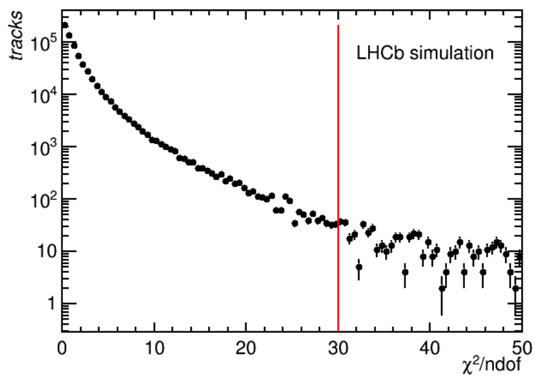
Figure 4.5.4: Left: Distance between extrapolated x-hits track and (truth-matched) hit in u layer. The red line represents the cut on the distance. Right: Distance between extrapolated xu-hits track and (truth-matched) hit in v layer. The red line represents the cut on the distance.

4.5.6 Calculation of χ^2 and outlier removal

At this stage, all possible hits in TT were added to the track candidates. It is the time to perform a χ^2 fit. If the fit χ^2/ndf is smaller than a given value (`MaxChi2` and `MaxChi2ThreeHits` for candidates with four and three hits, respectively, the candidate is accepted and passed on to the next stage. Otherwise, the hit that contributes most to the χ^2 is removed, and the fit is repeated. The procedure is repeated until either the χ^2/ndf is small enough, or alternatively, there are hits in less than three planes left. Furthermore, for each iteration of the fit, an explicit sanity check is made that each hit is still compatible with the estimated y position of the track. The χ^2 is defined as:

$$\chi^2 = \sum_{i \in hits} \frac{\left(x_i - (x_0 + t_x \cdot (z - z_{mag}) + c \cdot (z - z_{TT})(z - z_{TT}) + \left(\frac{dx}{dy}\right)_i y_0)\right)^2}{\sigma_i^2} \tag{4.21}$$

with $\left(\frac{dx}{dy}\right)_i$ the slope of the stereo plane and y_0 the displacement in y. Note that c is fixed, see section 4.4.3.



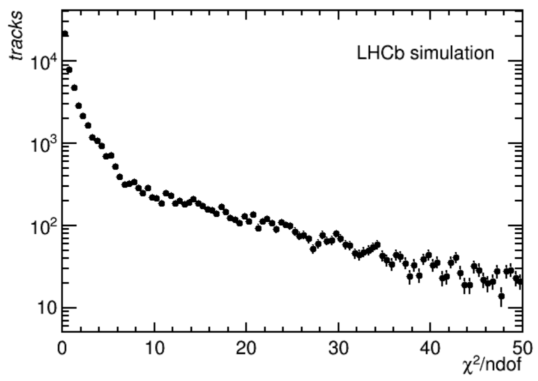


Figure 4.5.5: Left: χ^2 distribution for truth-matched tracks with 4 or more hits. The maximum allowed value is 30. Right: χ^2 distribution for truth-matched tracks with 3 hits. The maximum allowed value is 50.

4.5.7 Accepting the candidates

The track candidate is accepted in the final selection of track candidates, if it fulfills the following criteria. All cut values were obtained using simulated data, with the goal of keeping a balance between signal retention and background rejection.

- It has at least three hits in at least three different layers of the TT.
- The $\chi^2/ndf$is below a given threshold. This value is different for three-hit tracks (`MaxChi2ThreeHits`) and tracks with four or more hits (`MaxChi2`), see Figure 4.5.5.
- It contains at least as many hits as any other candidate for a given T track.
- The track has a pseudorapidity larger than 1.8 and smaller than 5.2.
- At least `NMinHighThresHits` of the hits have the high-threshold bitset.

The following checks are repeated, this time using the fit result instead of the initial estimates.

- The momentum estimate is compatible with the momentum estimate from the T track.
- The track has a minimum momentum `MinMomentum` and a minimum p_T `MinPt`.
- The track does not point into the beampipe.

4.5.8 Addition of overlap hits

For all track candidates whose are retained at this stage, hits in the overlap regions of the TT detector are searched for. TT modules are staggered in the z direction with an overlap, in order to cover the insensitive region of the modules and not to lose efficiency. A hit qualifies as an overlap hit if it is within a certain distance to the extrapolated track (`OverlapTol`), in the same layer as an already existing hit, but at a different z position.

As adding an extra hit will change the track's properties, the track is again fitted with the potential removal of outliers.

4.6 Selection of T tracks

T-Seeds together with TT hits constitute an input to the LongLivedTracking algorithm and their quality has a direct and significant impact on the algorithm performance. Therefore, the very first and crucial step is a filtering procedure, which purpose is to increase purity of the set of T-tracks by rejecting as many unreconstructible track as possible. This also allows to reduce the execution time since the combinatoric is greatly reduced. A previous analysis showed that T-Seed filtering based on a simple linear selection using track quality and kinematic variables is not feasible. Therefore, an idea of using machine learning techniques was proposed. This section is dedicated to present all studies, starting from preparing and understanding training dataset, selecting the best model, to attempting to understand the model's prediction.

The T-Seed selection problem is similar to the email spam detector. In both cases it is desired to keep all true events and remove as much as possible false events. The nature of this problem reflects in selection of the cutoff probability threshold.

4.6.1 T-seed classifier: Data Collection

The very first step within the machine learning model training pipeline is a data collection. It is not possible to build any statistical model without collecting sufficient amount of the data. From the perspective of this project the data was generated by the Monte Carlo simulation. To retrieve required data a dedicated tool within the Brunel project was implemented. This tool is dedicated to match track with Monte Carlo particle. The particle is considered as matched to a given track if they share at least 70% of the hits. Such a link allows to assign a target value to a given track. The flag is a binary value whether a given track is a **true** one.

One of the most difficult problem during the data preparation step was choosing the track labeling policy. Within this process, for each T-Seed a target flag was assigned. What is crucial, every time when this strategy is changed, the entire process of model selection and training has to be redone, which happend at least three times during author's study. The final labelling strategy tells the track is consider as a reconstructable, later called a **true T-Seed**, when it meets the following criteria:

- Has to be associated with a valid Monte Carlo particle;
- From above association the electron is excluded;
- Has at least two hit associated with TT station;
- no hits in the Velo detector.

Those criteria can be justified by definition of the Downstream track. Those are the tracks created by the particles that decays outside of the Velo detector.

4.7 T-seed classifier Exploratory Data Analysis

The next steps that were performed is Exploratory Data Analysis (EDA). This stage is essential to understand the structure of the input data, find patterns, relationships, or anomalies that can influence the outcome of the model prediction.

The search for the best classifier was performed using a simulated data sample containing the $B^0 \to J/\psi K^0_S$ signal events. The entire dataset contains more than 2 million T-Seeds (both true and fake). To avoid bias that could be introduced by considering only one magnet polarization the training dataset consists of combined datasets generated for both magnet polarities, in the equal proportions. The following variables were considered as an initial feature set. Further input dimensionality enhancement was done in the feature engineering phase.

- χ^2/ndf: T-track χ^2/ndf determined by the PatSeeding algorithm, which is defined in the following way:
$$\chi^2 = \sum_{i \in hits} \frac{1}{\sigma_i^2} \left(\frac{x_i - x_{track}(z_i)}{\cos\alpha} \pm r_{drift} \right) \tag{4.22}$$
where: (x_i, z_i) is the x coordinates of the i-th hit at the $y(z_i)$ predicted form the track model, σ_i is the hit postilion uncertainty and r_{drift} is the drift radius for Outer Tracker hits or 0 for the Inner Tracker.

- p: the absolute momentum of a T-track;

- p_t: the transverse momentum;

- N_{hits}: number of hits constructing tho a given T-seed;

- x_{Ttrack}: the x position of the T track's first state;

- y_{Ttrack}: the y position of the T track's first state;

- t_x: the slope of the track in the $x - z$ plane;

- t_y: the slope of the track in the $y - z$ plane;

- r_{track}: the distance from the beam line, z-axis, calculated for the first state. The r_{track} is calculated as follow:
$$r_{track} = \sqrt{x^2_{Ttrack} + y^2_{Ttrack}} \tag{4.23}$$

- η: pseudorapidity.

The selected feature distributions are presented in Figure 4.7.1. It is clearly visible that the data is not-linearly separable and the distribution for both true and false classes are very similar to each other. This gives a first impression, that the classification task is hard.

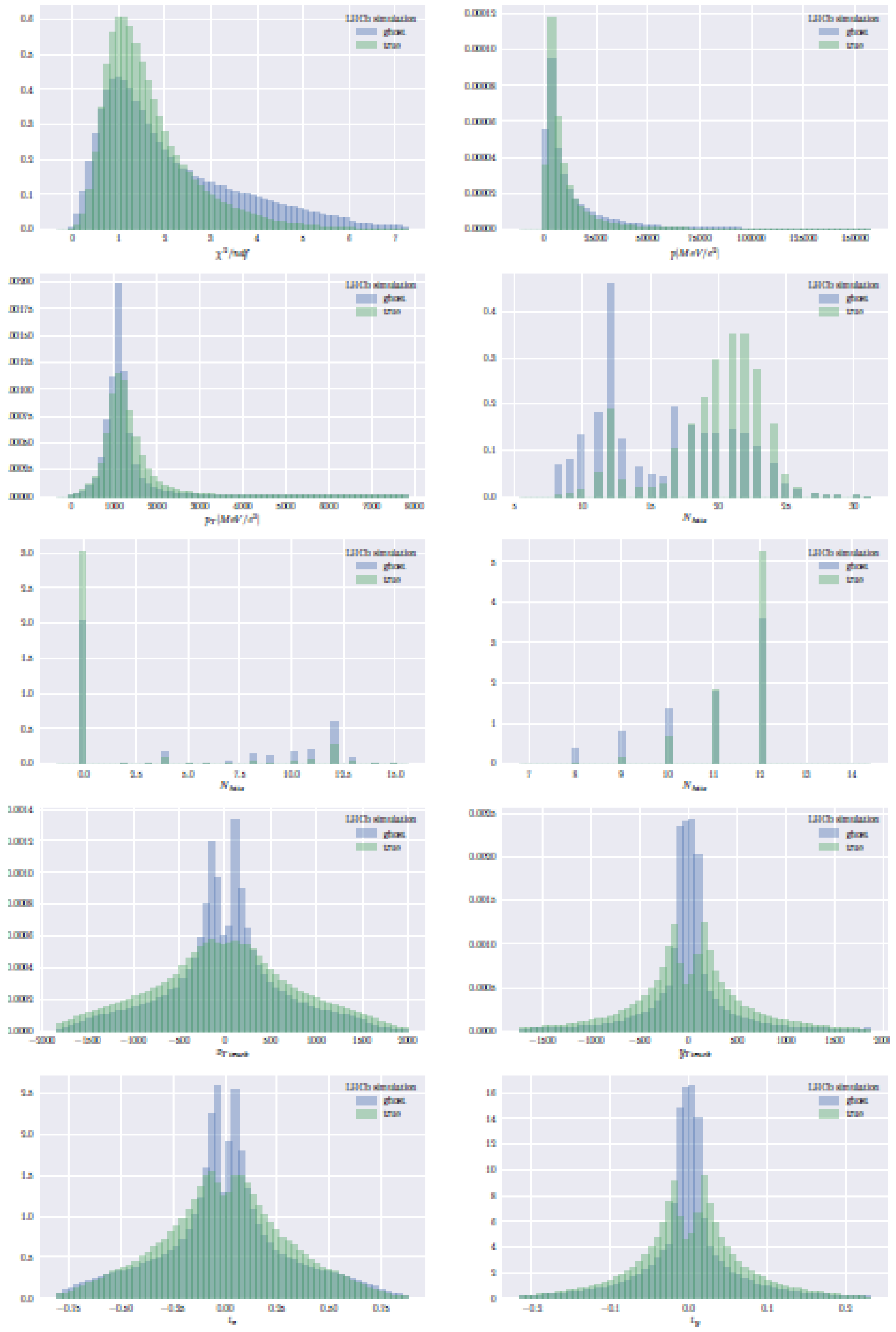


Figure 4.7.1: Distribution of the input variables used to train T-Seed filter. The sample was extracted using a $B_0 \to J/\psi K^0_S$ samples decays. The green solid histogram is the signal distribution, while the light blue distribution is the background.

One of the most popular way to show the dependencies among features is to use the Pearson correlation coefficient ρ. The formula for ρ is given by:

$$\rho(X, Y) = \frac{\mathbb{E}\left[(X - \mu_x)(Y - \mu_y)\right]}{\sigma_x \sigma_y} \tag{4.24}$$

This parameter has value in the range of $< -1, 1 >$. The value of 0 indicates no correlation, -1 and 1 implies strong negative and positive correlation respectively. In order to visualize all of the correlation coefficients, so called, correlation matrix was created. Figure 4.7.2 shows two correlation matrices, each of which is plotted for a different value of the target flag. There are no significant differences between true and false T-Seeds correlation matrices.

The Pearson correlation coefficient is a good metric to detect linear dependencies among features, however it is not sensitive to more sophisticated (like quadratic) relations. To overcome this limitation the pair-plot, showed in Figure 4.7.3, was made. A pairs plot allows to visualize both distribution of single variables and relationships between two features, which is represented as a scatter plot. It is particularly interesting to visually inspect how the variables are distributed for a given pair of attributes. To make this plot more readable the data were split into two groups according to the target flag. That kind of plot allows making an initial guess on the importance of the features, i.e., if two features are separated from each other, this may indicate that those features can have a significant impact on a classifier decision. Assuming the model will be capable of learning this separation.

4.8 T-Seed classifier: baseline

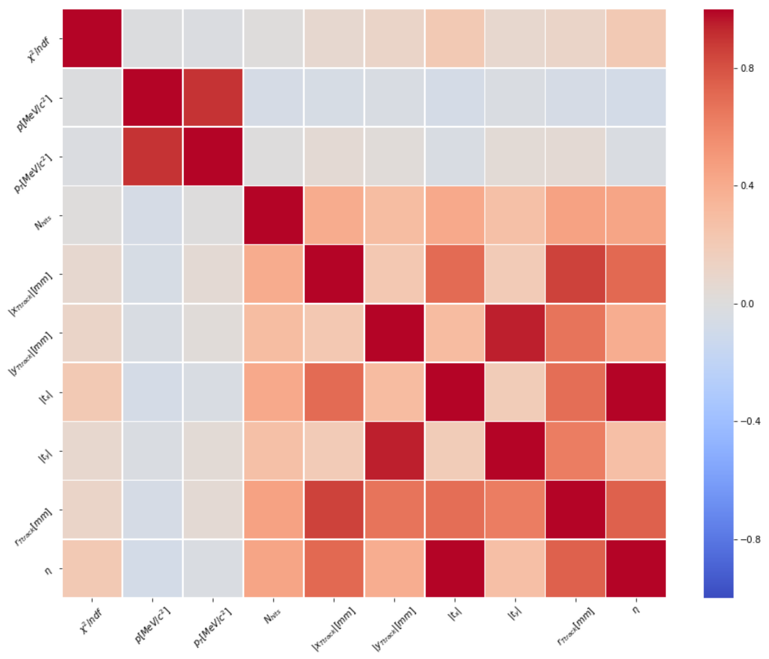
This section is dedicated to present the study baseline, which was implemented as training of two models, namely *k*-NN and logistic regression.

These two models were selected as a baseline due to their simplicity indicated as an amount of time that is required to train them. The simplicity, except for the model's underlying idea, is manifested by a small number of hyperparameters that has to be tuned. The baseline score is the entry point to the remaining machine learning study. This score should be over-performed by all of the more sophisticated models. It also provides an initial understanding of how hard the classification problem is. In some cases, excluding this one, the study baseline could be an average human performance on a given task.

4.8.1 *k*-NN

The *k*-NN described in section 3.3.1, is a model that has two tunable hyperparameters a number of neighbors *k* and a distance metric. The model training pipeline contains of the following steps:

- Data extraction;
- Data normalization: each of the features were normalized by removing the mean and scaled to unit variance. Centering and scaling happen independently on each feature by computing the relevant statistics on the samples in the training set;



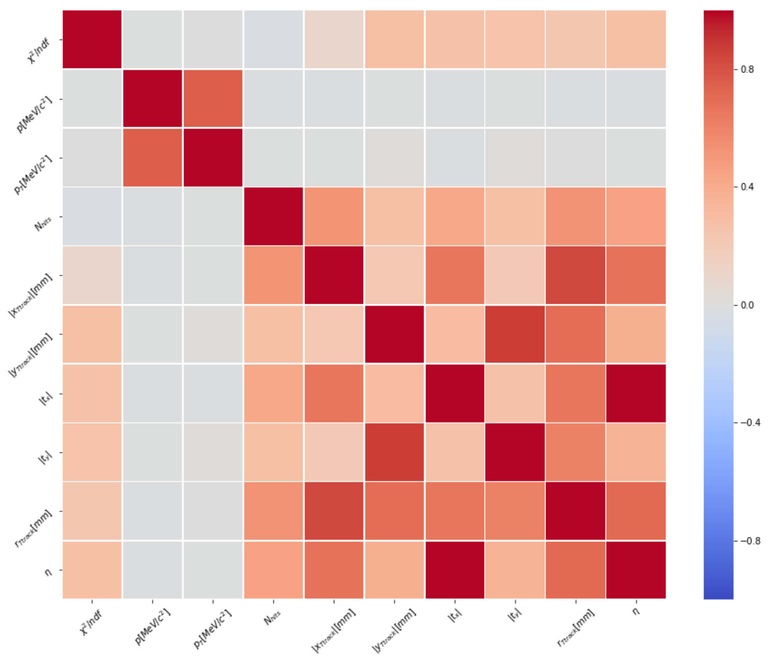


Figure 4.7.2: Pearson correlation matrix that were obtained using a subset of input features. The top plot shows correlation matrix for True T-seeds. The bottom plot presents similar Pearson correlation matrix generated using the ghost T-Seeds.

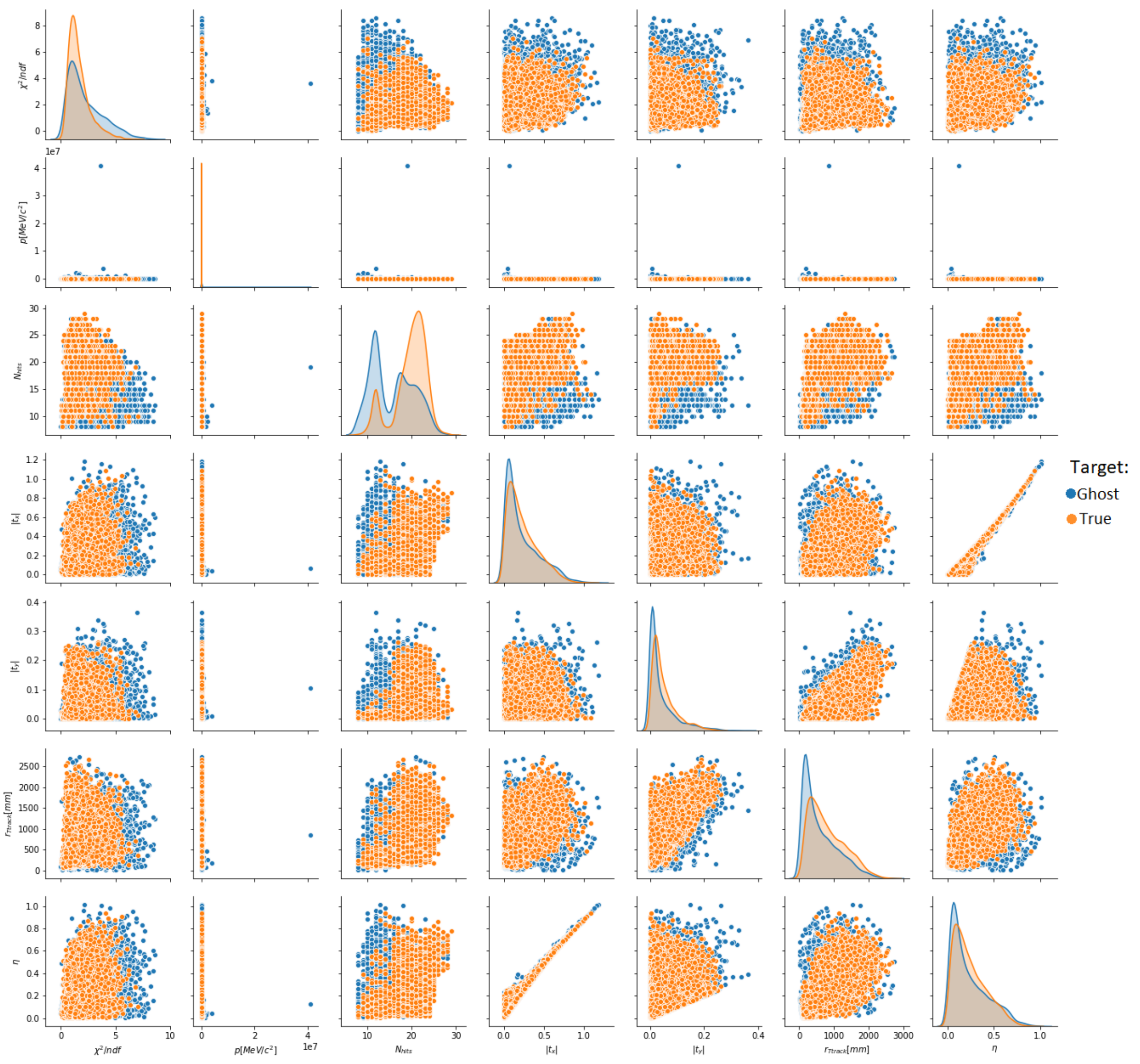


Figure 4.7.3: Pair-plots that were generated using subset of input features. The input data was separated based on the value of the target flag. The blue points represents ghost T-Seeds and the orange points represents true T-Seeds.

- Model training and performance evaluation.

The mentioned pipeline were implemented using the sklearn framework [112].

Figure 4.8.1 presents the area under the ROC curve score versus the number of neighbors. The score and its uncertainty were obtained using the 4-fold cross-validation method. The maximum score value (ROC AUC score 76%) were obtained for $k \approx 210$.

4.8.2 Logistic Regression

The second baseline mode is Logistic Regression described in section 3.3.2, and implemented using sklearn framework. The input to this model has to be normalized, which is the only preprocessing step that was applied, therefore the data processing pipeline was identical to the one described in section

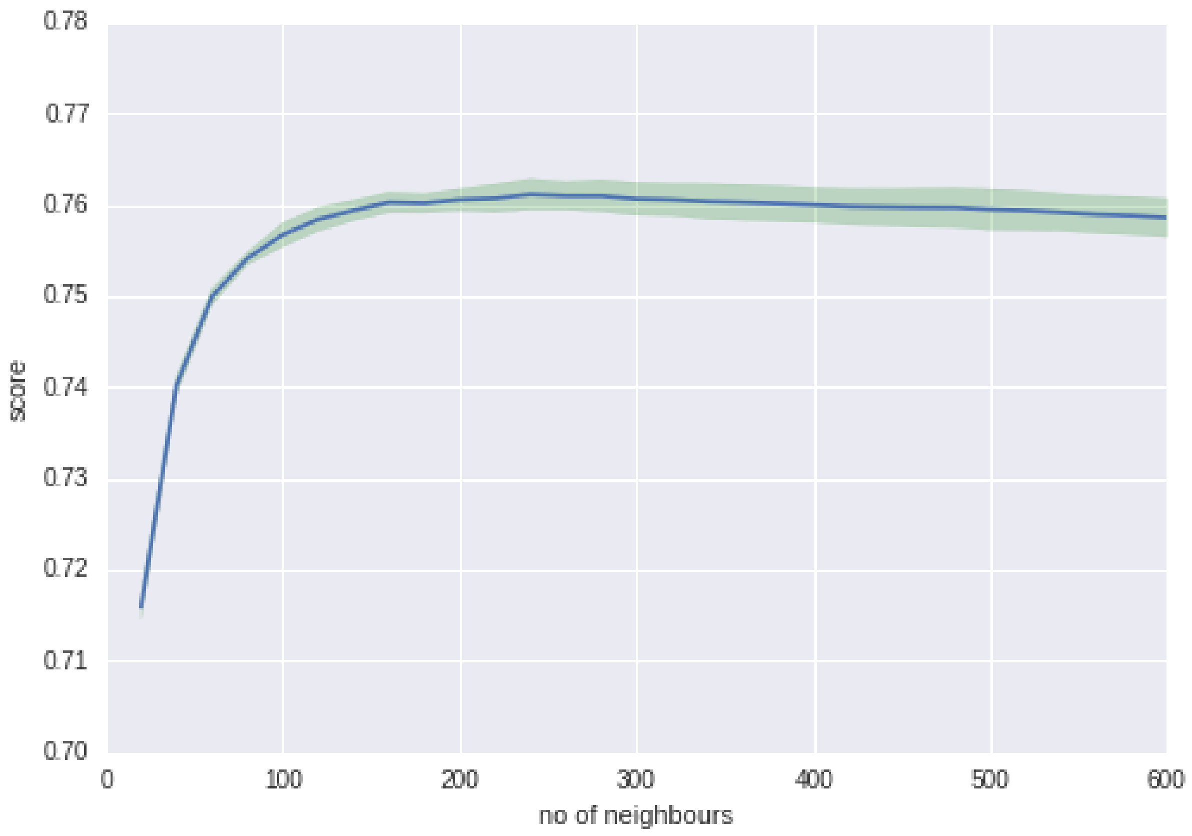


Figure 4.8.1: The k-NN performance plot, ROC AUC score vs the number of neighbor. The green shaded area represents score's one standard deviation region.

4.8.1. To maximize the classifier score analysis of the L_1 and L_2 regularization constants were conducted. These two hyperparameters were introduced to reduce overfitting, although as presented in Figure 4.8.2 none of them has significant impact on the classifier performance.

Finally, to check the stability of the prediction, the study of model performance versus a number of training examples was performed. The results are shown in Figure 4.8.3. This plot indicates the biggest limitation of the linear model. The model's performance is not very weakly affected by increasing the amount of data, that were used to train them.

The score, measure as ROC AUC, obtained by the Linear Regression model is about 74%, which is slightly worst than the result obtained by the kNN classifier.

4.9 T-Seed classifier: study based on XGBoost model

The previously discussed two models were selected to provide the baseline scores. This section presents the training strategy and results that were obtained using the XGBoost model.

The first step within this analysis was to train the model with default values of hyper-parameters. It was done to get a better understanding of the model initial performance and to estimate the number, or at least the order of magnitude, of weak learners that need to be added to the boosting model. This

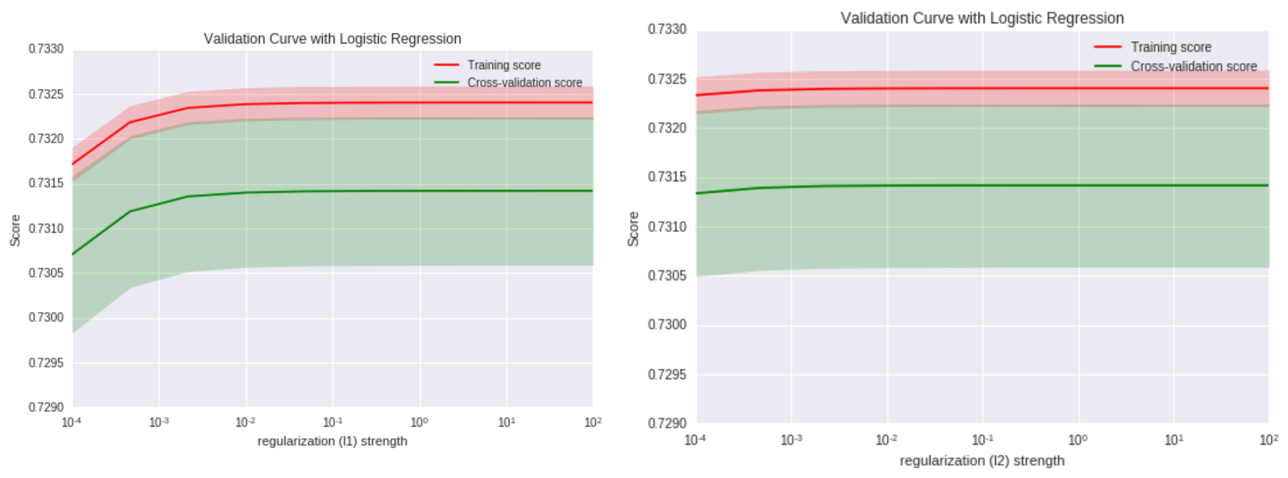


Figure 4.8.2: Logistic Regression validation curves. The plots present the impact of L_1 (left) and L_2 (right) regularization on classifier performance.

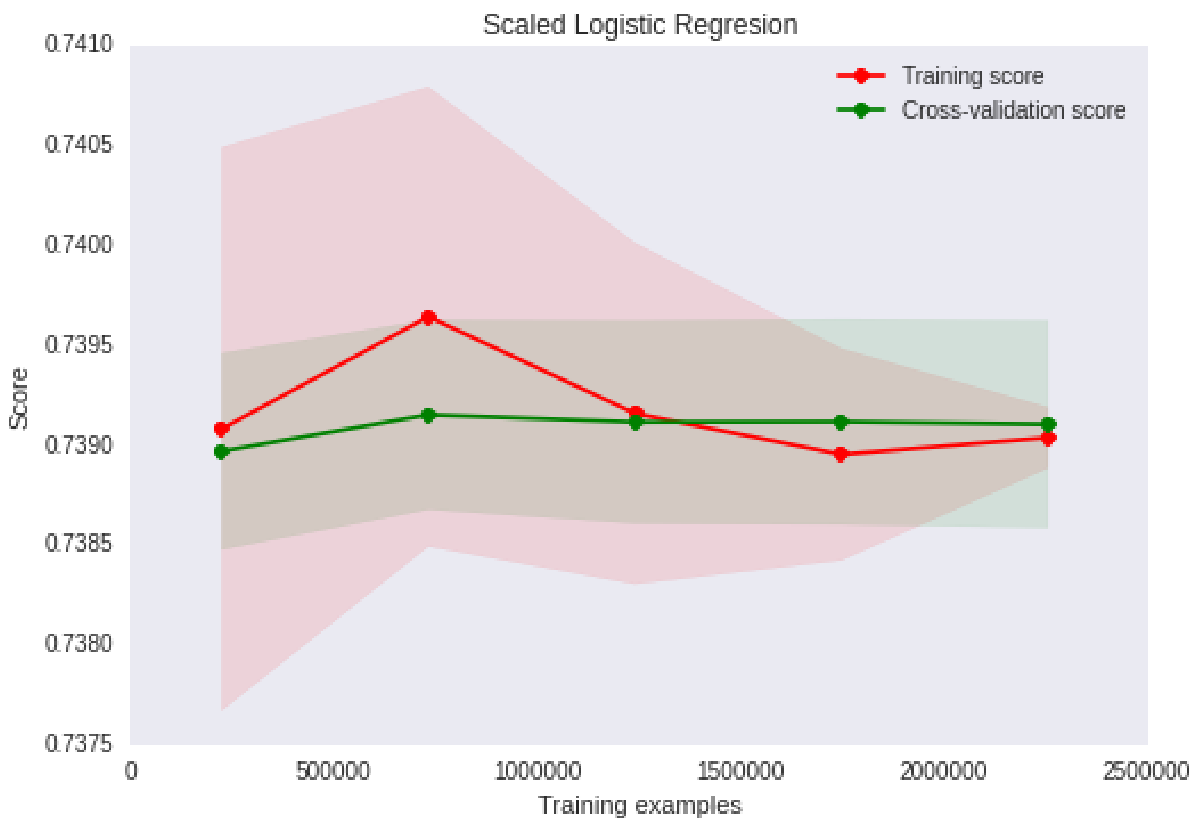


Figure 4.8.3: The Logistic Regression model performance vs. the number of training examples

number can be estimate by detecting moment when model performance starts to saturate.

The model saturation occurs when adding a new tree has no (positive) impact on the performance, measured using both training and the validation datasets. This effect can be easily detected using the **learning curve**. The learning curve plot is a two-dimensional plot with classifier performance on the y-axis and a number of trees on the x-axis. The saturation effect occurs when learning curve plateau.

The initial training estimated the number of trees to be around 700. The other hyperparameters, that were taken into consideration are listed below:

- **learning rate** - shrinkage parameters (see section 3.3.3), which control the weight of new tree added to the model;
- **max depth** - a maximum allowed depth [3] of each of the trees that constitute xgboost model, increasing this value makes week learners more complex and more likely to overfit thus, it is recommended to keep the value of this hyperparameter reasonably low.
- **gamma** - minimum loss [4] reduction required to make a further partition on a leaf node of the tree. Increasing the gamma values makes the boosting process more conservative.
- **min child weight** - a minimum sum of instance weight needed in a child to make a split. If the tree partition procedure step produces a leaf node with a sum of instance weight smaller than **min child weight**, then the building tree process is terminated. This parameter can be interpreted as a minimum number of instances needed to be in each node.
- **colsample by tree** - subsample ratio of features used to construct a particular tree. Subsampling occurs once for each tree.
- **reg alpha, reg lambda** - regularization factor L1 and L2 on weights.

Taking into consideration the dimensionality of this optimization problem, as well as the fact that each training iteration (or function evaluation) takes about 2 hours the Bayesian Optimization seemed to be the most promising approach to tune the hyperparameters. Although, all three hyperparameters tuning strategies described in section 3.3.6 were applied. The Bayesian optimization needs about 40 iterations to find the optimal hyperparameter set [5]. The Grid and Random Search provide similar results, although each of these methods requires more optimization steps to find a set that performs as good as the one produced by the Gaussian optimization.

Figure 4.9.1 presents the Bayesian optimization results in a form of coverage plot, where each of the scores is a ROC AUC score calculated using three-fold cross-validation method. Within the presented experiment the Gaussian process was selected as a surrogate function and Expected Improvement played the role of an acquisition function. Each training round was terminated once the performance on the test dataset has not improved after a fixed number of training iterations. This method is called **early stopping**, and its main purpose is to prevent training the model when it performance saturates. The positive side effect of this method is better control of the overfitting.

Using selected by the Bayesian optimization method set of hyperparameters, xgboost based model achieved a score, measured as an area under the ROC curve of 94%. Figure 4.9.2 presents three different learning curve plots. The first one (top left) shows the training progress measured as a reduction

[3] The tree depth is a maximum distance from the root to the leaf.

[4] In this context, the loss refers to the tree-related loss e.g., information gain

[5] In this context **optimal** is consider to be the best one for a given subset of hyperparameter's ranges, it is not guaranteed to be globally optimal

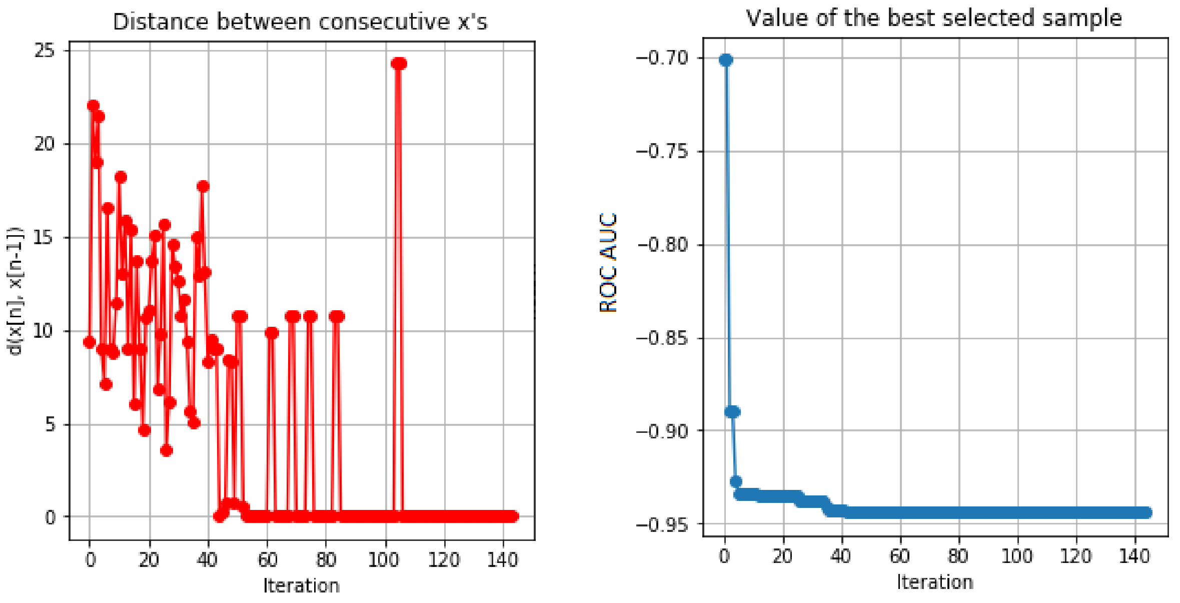


Figure 4.9.1: Bayesian optimization coverage plots. Left plot presents iterations vs. distance between consecutive selected hyperparameters sets, the right plot shows iterations vs. the ROC AUC score of the current model.

of the cross-entropy loss versus the number of trees constituted the model. The second one (top right) presents the binary classification error rate, which is calculated as a ratio between several wrongly classified cases to all test cases, vs the number of training iterations. The final, bottom one presents the area under the ROC curve score versus the number of weak learners. Those learning curve plots were generated using a test sample that constitutes 20% of the entire dataset.

The learning curve plot is one of the most important tools to investigate whether the model is overfitted visually. Overfitting refers to a model that has learned the training dataset too well, including the statistical noise or random fluctuations in the training dataset. This effect becomes apparent when the performance gap between training and validation curves increase when increasing the model complexity, which is shown in Figure 4.9.3. Figure 4.9.2 indicates that the model performs reasonably well, and the effect of over-training is not present [6].

4.10 T-Seed classifier as a bonsai Boosted Decision Tree

As described in section 3.4, the evaluation of the continuous complex classifier is too expensive (from the perspective of CPU time) to deploy that kind of model within the HLT2 online system. Therefore, the model was binarized and deployed in the form of bBDT. The structure of the bBDT is visualized in Figure 4.10.1.

The response of the binned BDT and its ROC curve are presented in Figures 4.10.2 and 4.10.3 re-

[6]This conclusion is based on a specific test set. It is not guaranteed to obtain the same test accuracy for all possible test sets.

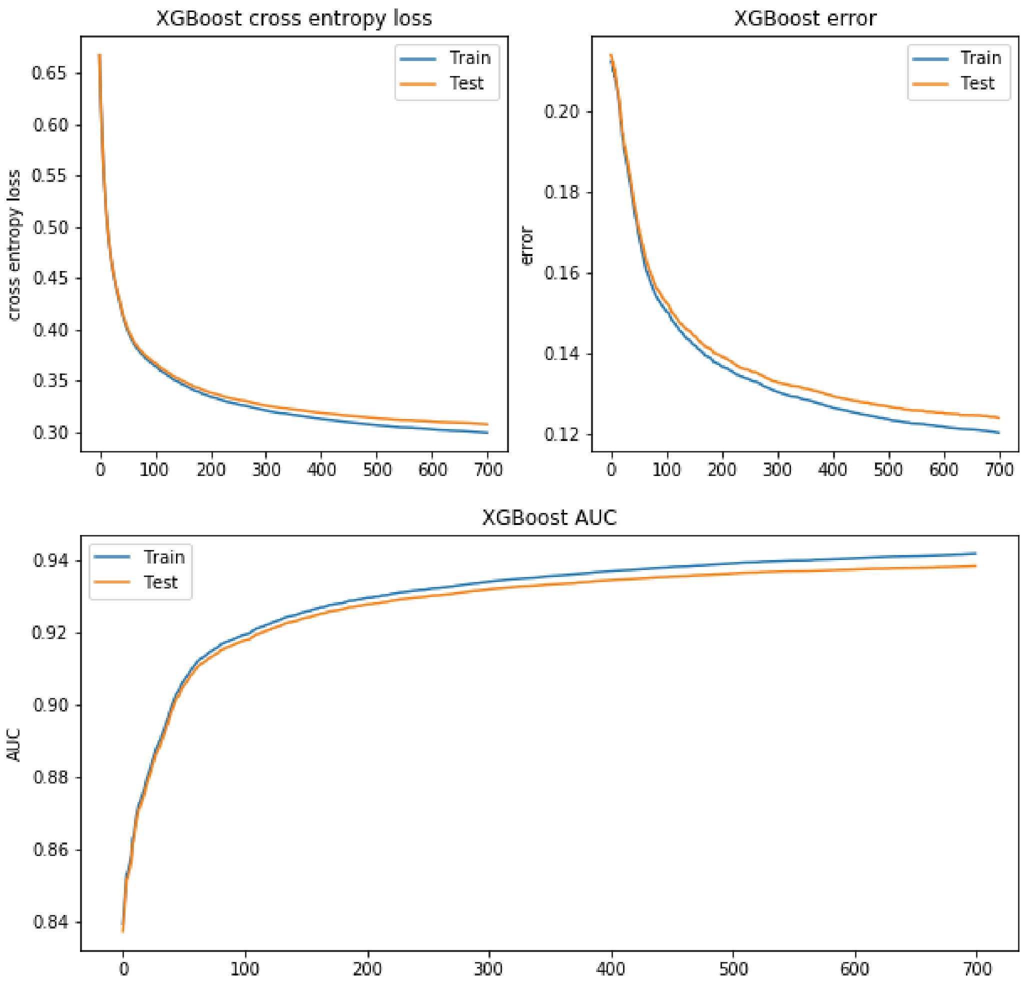


Figure 4.9.2: Learning curve for the XGBoost classifier. Upper left plot presents cross-entropy loss function vs. a number of week estimators, right upper plot shows miss classification vs the number of trees. Bottom plot presents the ROC AUC versus training iterations.

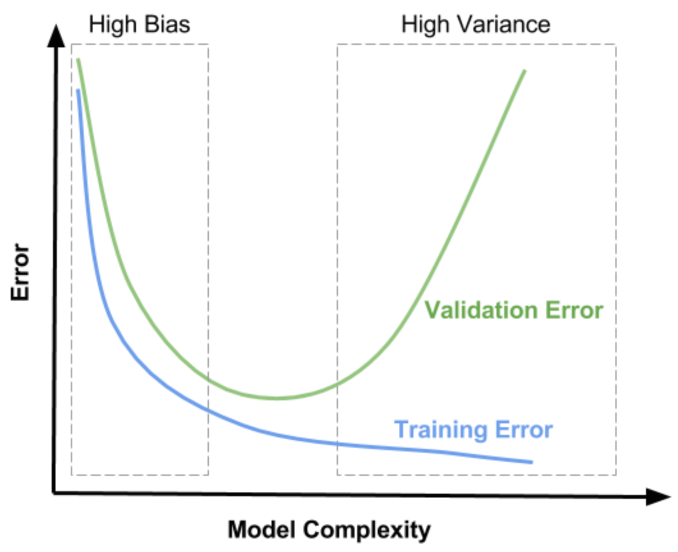


Figure 4.9.3: Cartoon presenting different behavior of the Boosted Decision Tree classifier. The leftmost region represents the model that is under-fitted which is reflected in decreasing learning curves. The rightmost region corresponds to the model that is over-trained, in such a case validation curve is increasing while train curve is decreasing.

spectively. A slight drop in the performance of the binned classifier with respect to the full one is visible. The figures of merit measured for the bBDT algorithm amounted to 87%. Based on the ROC curve the classification threshold was selected to be 0.07, section 4.14.4 contains a discussion on different values of the cut threshold and its influence on the track reconstruction algorithm performance. This highly conservative value of the threshold was advocated by the fact that the final classifier has to keep more than 99% of the true T-Seeds. Nevertheless, this threshold value allowed to remove 30 % of the fake seeds, thus significantly decreased the ghost rate, formally defined in section 4.14.

4.11 T-Seed classifier: studies based on the deep neural networks

The final model that was taken into consideration when working on a T-Seed selection model was deep neural network, described in 3.3.4. All described studies were conducted using PyTorch framework [110], which allows for tensor computation acceleration via graphics processing units (GPU) and supports automatic differentiation. Moreover, Pytorch supports the C++ interface, which makes the model's deployment within the LHCb trigger convenient. However, studies discussed in this section were never deployed within the LHCb trigger system due to the algorithm's submission deadline, thus the model based on bonsai Boosted Decision Trees was selected instead. Nevertheless, the author decided to include this section since it may be useful for the future studies.

Pytorch provides a number of utility classes that abstract a lot of complexity, such as data parallelization and batching. From a practical perspective, the developer needs to ensure a customized implementation of the following classes:

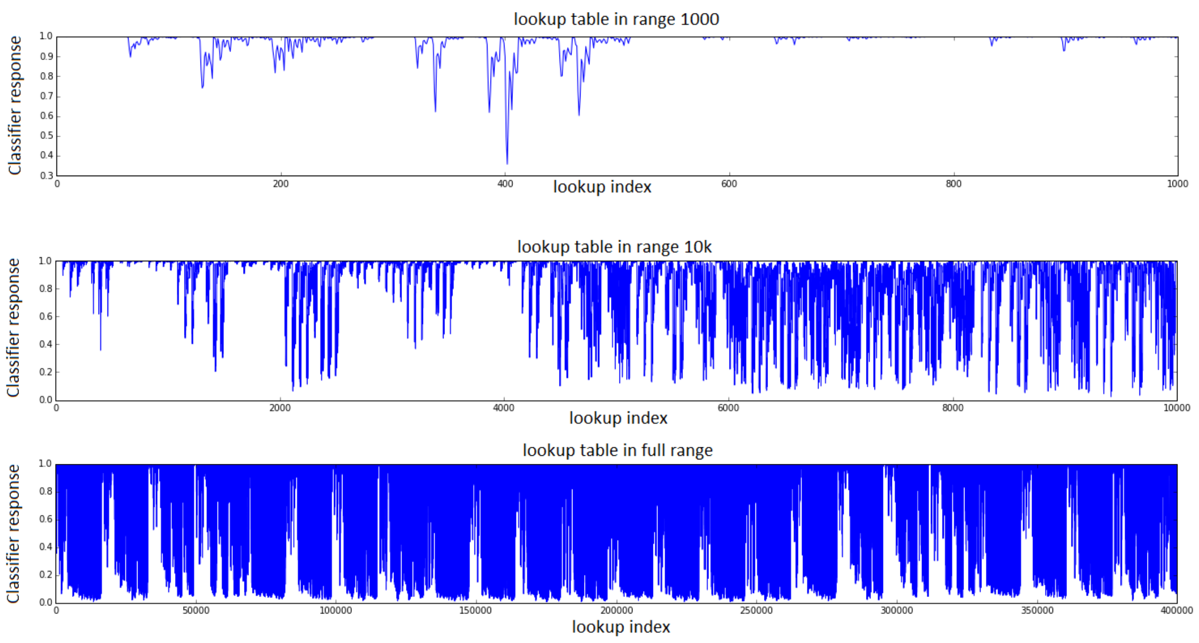


Figure 4.10.1: Structure of the bBDT lookup table. Each of the plot presents different range of lookup table indices (x-axies) versus response of the classiers (y-axies).

- **Dataset**. This class is dedicated to retrieving the input data, applying data transformations and optional augmentation logic, and converting processed data into the PyTorch tensor. Within the scope of the presented study, the transformations were limited to adding such features as pseudorapidity and the data normalization.

- **DataLoader**. The DataLoader combines a Datasets object with different samplers producing data batches. Samplers are implementations of different strategies for providing data to models.

- **Criterion**. The criterion is an abstraction of the loss function. Within this study's scope, only the Cross-entropy loss were used, however section 4.15.2 discusses possible alternatives.

- **Optimizer**. An optimizer is an object that holds the current state of the model and update parameters based on the computed gradients. The choice of the optimizer is considered as a model hyperparameter. However, the recommended and used in the presented studies choice is ADAM [91].

- **Model**. It is an abstraction that defines the forward propagation, and it contains all components (layers) of the model. Thus, the researcher has to override two methods. The first one is the *init* method, which defines and initializes the network's architecture. It is recommended to initialize the layers using, so-called, Xavier initialization [70]. The second method that has to be implemented is the *forward* technique. It defines how the input tensor is processed in order to get a vector of output probabilities. The framework takes care of the backward propagation, so the user does not need to manually calculate the gradients of the loss function.

- **Training loop**. This utility function defines the way how the model is trained and allows customizing training progress measurement. Roughly speaking, the training loop consists of two

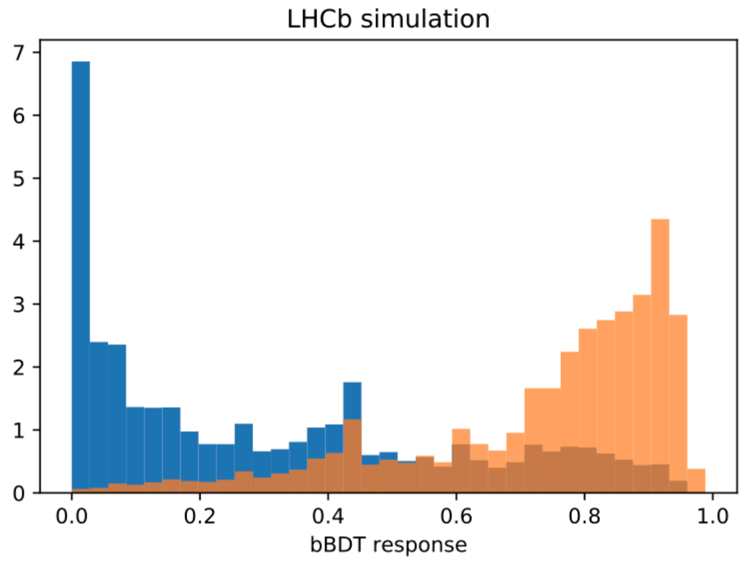


Figure 4.10.2: bBDT classifier output distribution, blue is background and orange is signal

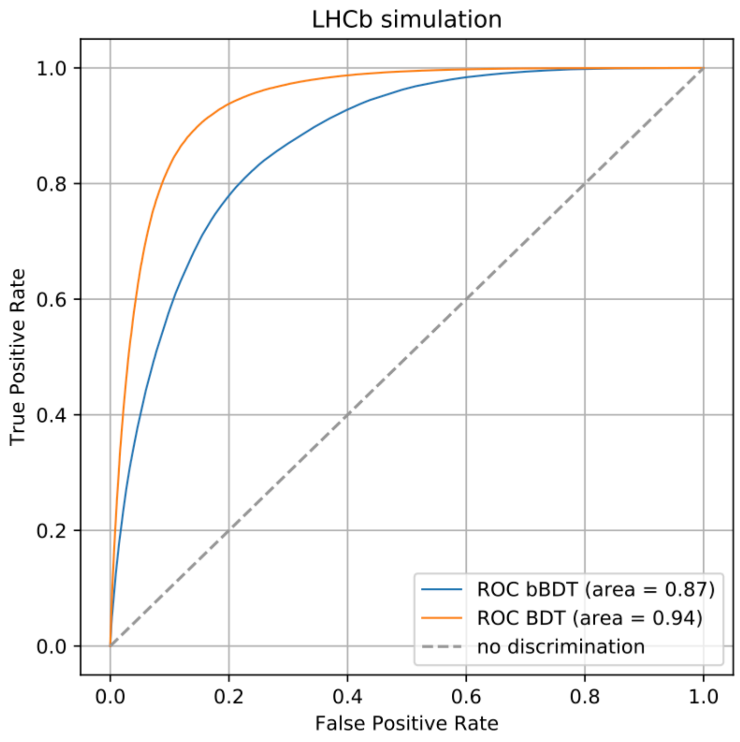


Figure 4.10.3: Comparison of ROC curves for selecting true T tracks using a simulated $B_0 \to J/\psi K_S^0$ sample before and after binning.

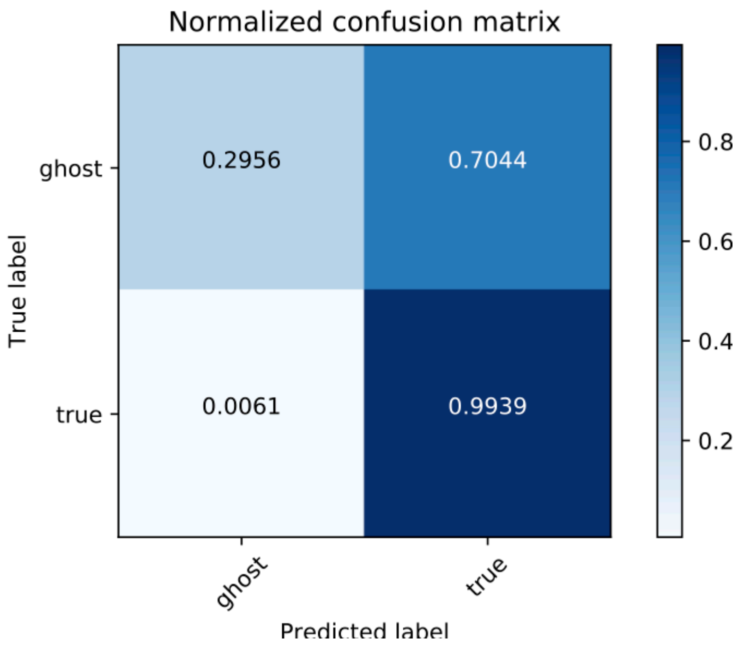


Figure 4.10.4: Normalised confusion matrix determined for the trained bonsai BDT (binned) model.

nested loops (one over epochs and the second one over batches of data). Within the presented studies' scope, the training progress was measured as loss function versus epoches and batches. To ensure that the model is optimal, quantities such as information flow and the changes in the model's weight distributions over time were monitored. An exemplary plot of the information flow is shown in Figure 4.11.1.

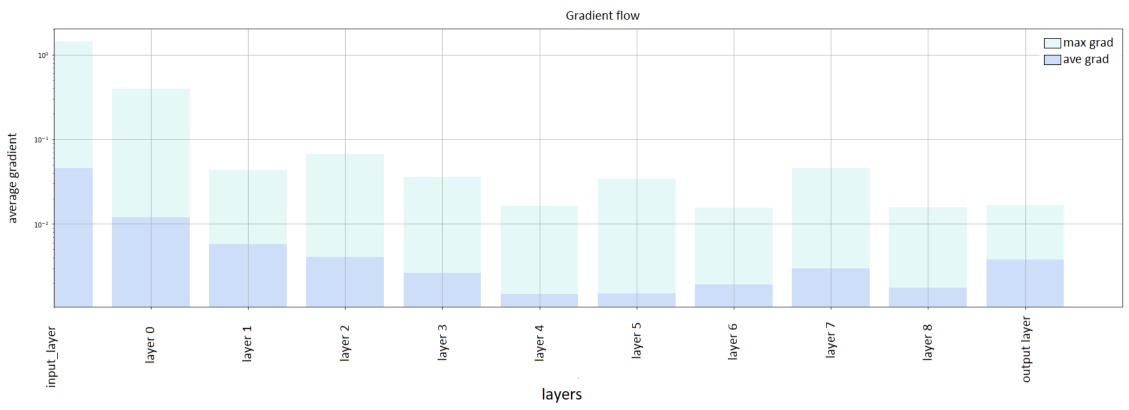


Figure 4.11.1: Information or gradient flow visualization. This figure was generated for network with nine hidden layers. The light blue bars indicate mean value of the information flow, while the dark blue bars represent maximum value of it.

After all of the beformentioned components were implemented, the next step is to check whether the model can overfit on a small sample of the data, this method was recommended in [88]. If the model is not able to reach a low error rate that may indicate some issues, bugs, or model misconfiguration.

The most time-consuming part of the study was the optimization of the network's architecture since the number of tested models exceeds one hundred, and training one model might take a couple of hours.

The architecture optimization procedure consists of the following steps. Start with a shallow network, that contains a single hidden layer only. Then iteratively extend the previously tested model by adding a hidden layer, which makes the model deeper. Within each iteration step, try to optimize the architecture of the current network. The following choices of network internal structure were considered:

- Use the constant number of neurons per hidden layer;
- Build a triangle-type network, where the number of the hidden neurons decrease with network depth;
- Optimize number of hidden units per layer using Bayesian Optimization;
- Use the Batch Normalization layer. This special layer normalizes the output of a previous activation layer by subtracting the batch mean and dividing by the batch standard deviation. For more details on how the Batch Normalization works, see [85].
- Use dropout layer. The dropout layer works by randomly setting the outgoing edges of hidden units (neurons that make up hidden layers) to zero at each update of the training phase. The detailed description of the dropout layer can be found in the original paper [131].

Within the presented study, models with up to fifteen hidden layers were tested. Figure 4.11.2 presents a selected four learning curves (cross-entropy loss versus the number of training epochs) each of them documents training progress of a model with a different number of hidden layers. Surprisingly, adding more than two hidden layers to the model has a neglectable influence on the model prediction performance, even though the number of the network parameters increased by two orders of magnitude. Thus, the recommended architecture of the network contains two hidden layers. Such a shallow network produce similar results, measured as a ROC AUC, to the optimized xgboost one, which is shown in Figure 4.11.3.

4.12 T-Seed classifier: model output interpretation

This section is dedicated to emphasizing the importance of the model's prediction interpretability. As presented before the T-Seed filter achieved satisfactory overall performance, but the question regarding the reliance still remains open. Within the scope of this thesis, three approaches to interpret Gradient Boosted Decision Tree model prediction have been tested; each of them has been described in section 3.3.7.1. The classical approach to interpret feature importance is to look at the model globally. The usual way to view the importance of the features is to investigate the following metrics:

- The feature **weight** is the number of times a feature appears in a tree across an ensemble of trees. For instance, if the model was built on top of the dataset consisting of 100 entries, five features

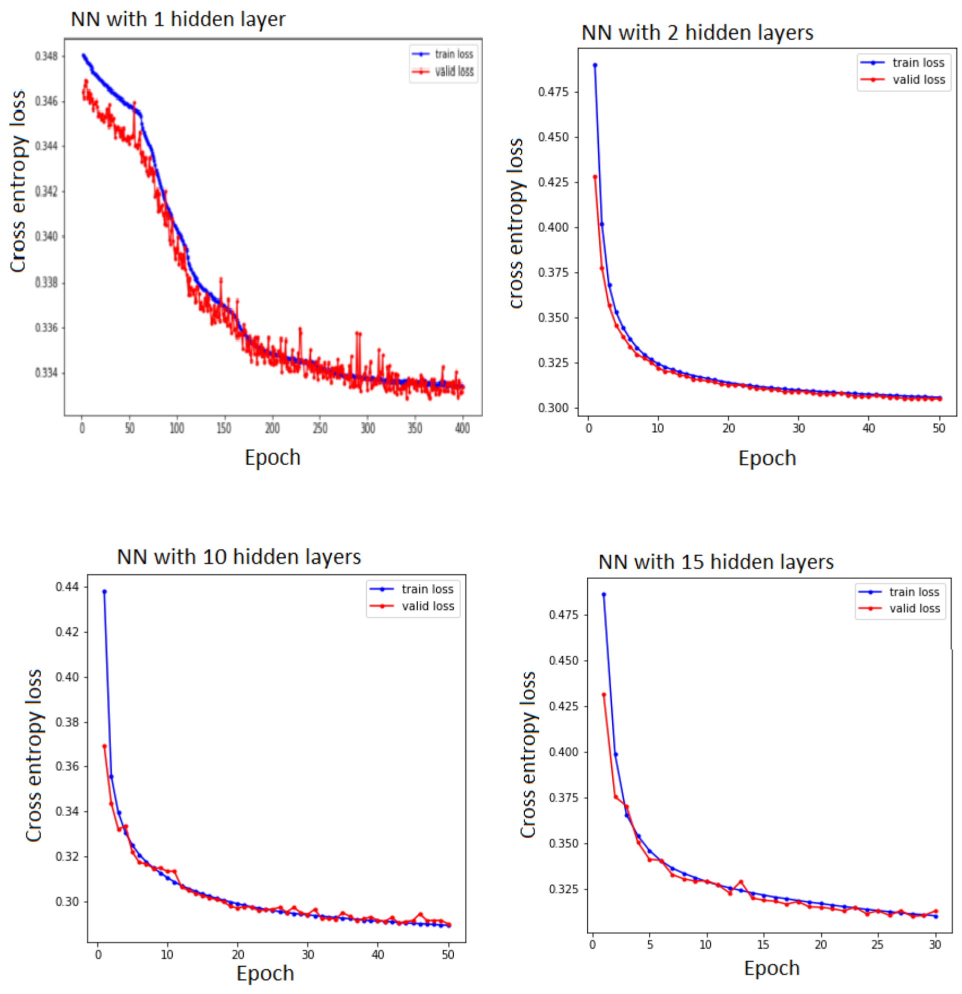


Figure 4.11.2: Fully-connected neural network learning rates. Each of the plot was obtained for model having different number of hidden layer. Upper left for one, upper right two, bottom left ten and bottom right fifteen.

and the model itself consist of 3 trees, and suppose the *feature1* occurs in the 1 splits, 2 splits, and 3 splits in each of three trees respectively, then the weight for *feature1* will be $1 + 2 + 3 = 6$.

- The **Gain** measures the relative contribution of the corresponding feature to model by taking each feature's contribution for each tree within the model. This metric can be interpreted as an average training loss reduction gained when using the feature for a split.
- The **Coverage** describes the relative number of observations related to this feature. In the above example if the *feature1* is used to decide the leaf node for 20, 10, 5 observations in the mentioned trees respectively, then the coverage for this feature is $20 + 10 + 5 = 35$.

Figure 4.12.1 presents three plots, each for a different metric described above. It is clearly visible, that these plots contradict each other. According to the feature weights, the most important feature is seed $\chi^2 dof$ and the N_{hits} is redundant, while the coverage metric scores it as a second most important.

To understand why some, possibly categorical, features have smaller weight value consider the following example. Let *feature1* be a binary feature, which is highly correlated with the target variable and its

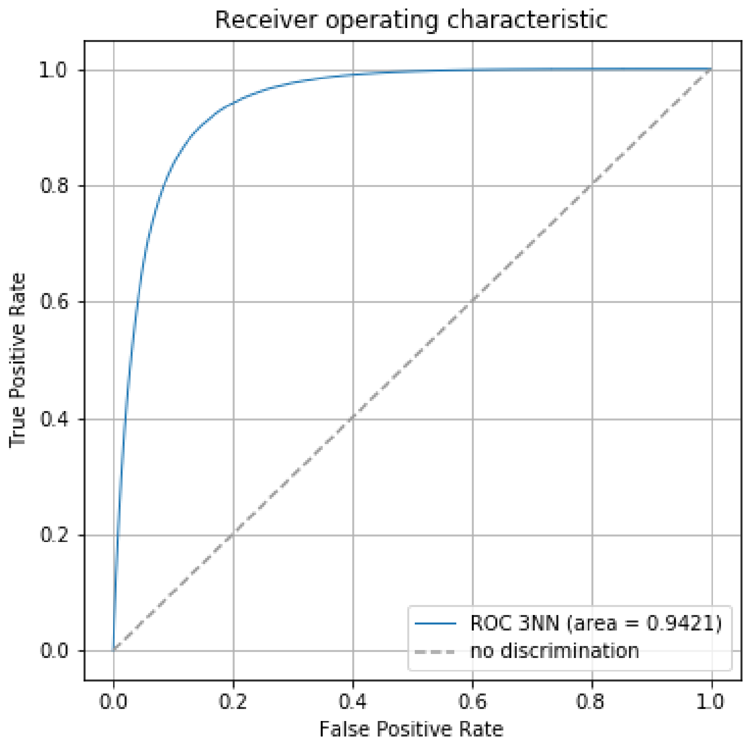


Figure 4.11.3: Fully-connected neural network containing two hidden layers ROC curve for selecting true T tracks using a simulated $B_0 \to J/\psi K_S^0$ sample.

inclusion or removal has a significant influence on classifier performance. Due to the nature of feature1, it has a tiny number of possible values compared to the other features. Such variable can be used at most once per tree, while continues ones may appear more often on a different level of the tree. Therefore, such a feature gets very low weight value. For this reason, none of the presented global metrics gives an ultimate estimation of the importance of the feature. To get a better understating of it, and taking into consideration the non-linear nature of the model's decision boundaries, it would be worth investigating different approaches of measuring the feature importance.

The second method to take a closer look inside the black box is based on the idea of Shapley value, described in section 3.3.7.2. The analysis of feature importance is usually done by presenting two groups of plots. The first one is presented in Figure 4.12.2. The standardized importance plot provides a notion of relative feature importance and can be compared with the classical plots described previously. Although, it does not provide any additional information regarding the impact that a particular feature has on the model's output. To overcome this limitation the Shap summary plot was introduced, which leverages individualized feature attribution to the model's decision. In order to make that kind of plot the features are sorted by their global impact $\sum_{j=1}^{N} |\varphi_i^{(j)}|$, then each of the dots corresponding to the Shap values $\varphi_i^{(j)}|$ are plotted horizontally, stacked vertically when running out of space. This concept allows achieving similar effect to the violin plots [81], which can be further enhanced by coloring the

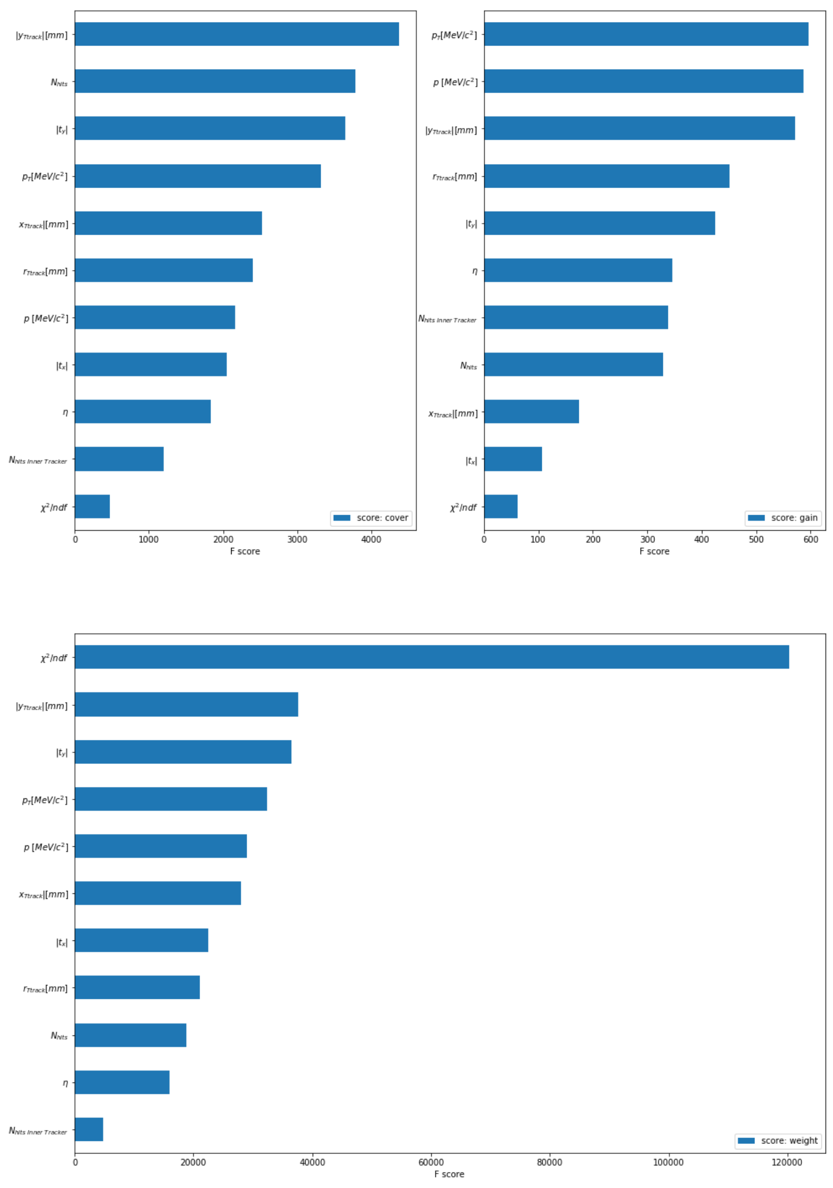
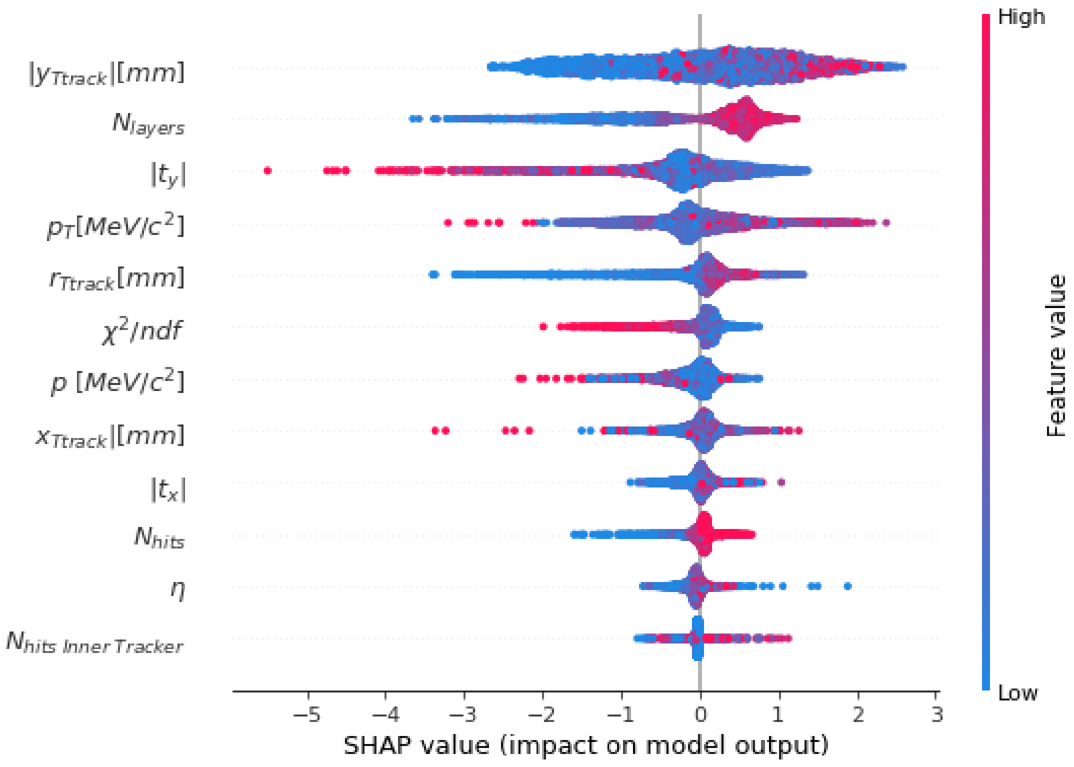


Figure 4.12.1: Feature importance plots for XGBoost model. Each of the plots were obtained for a different global feature importance metric. According to the coverage (upper left) the four most important features are $(|y_{Ttrack}|[mm]$,N_{hits},$|t_y|$, $p_t[MeV/c^2])$, gain (upper right) selects $(p[MeV/c^2]$,$p_t[MeV/c^2]$,$|y_{Ttrack}|[mm]$,$r_{track}[mm])$ as the most important ones and finally according to the weight metrics (bottom) such features as ($(\chi^2/ndf$,$|y_{Ttrack}|[mm]$,t_y,$p_t[MeV/c^2])$ are the most important ones. Each feature importance score is measured as a F-score.

dots according to the feature's value.

To get a better understanding of Shap values presented in Figure 4.12.2 let focus on one feature -



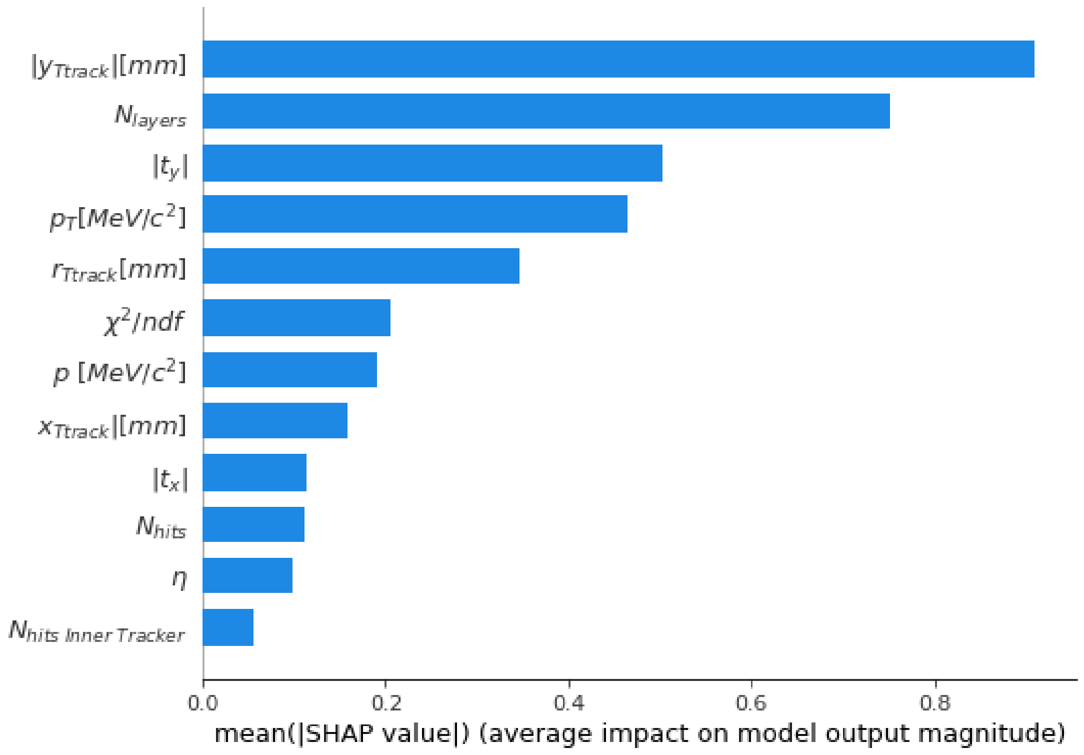


Figure 4.12.2: Shap summary plots of all T-Seed classifier features. The plots were obtained using a sample of 50 thousand test examples. The interpretation of this plot is straightforward, the higher Shap value of the feature, the higher influence on the decision of whether T-Seed is reconstructable. In the upper plot, each dot represents every feature for every individual test example that was run through the model. Dots are colored by the feature's value (red high, blue low). The bottom plot presents the standard feature importance bar plot. The x-axis is essentially the average magnitude change in model output when a feature is hidden from the model.

N_{layers}. It is the second most important one to decide whether a T-Seed is a ghost. The impact of this feature on a model's output varies smoothly as the value of the feature increases, which is indicated by the smooth color gradient. The long tail reaching to the left means that the extreme values of this variable can significantly increase the probability that this T-Seed is not a valid track segment. This is consistent with the priory intuition saying that seeds, that were reconstructed using a small number of hits are more likely to be ghosts.

The second type of Shap based plot is dependence plot presented in Figure 4.12.3. The concept is similar to the pair-plot, which are usually created in order to visualize dependencies between two features. The shap dependence plots also take into consideration the Shap values. The idea is to make a scatter plot, shap value versus feature value, and apply color scheme that is based on the value of the second feature.

Figure 4.12.3 (upper left) is the one that is the easiest to interpret. It shows the shap value versus the number of layers. It is clearly visible that T tracks with the numbers of measurements smaller than 15 push strongly the classifier toward negative decisions. This effect is escalated when the track has a big value of $\chi^2/\,dof$. This means the classifier decisions are compatible with an initial understanding of the problem, the track that was reconstructed using a small number of hits and it's fit to the track model has a poor quality is more likely to be a ghost.

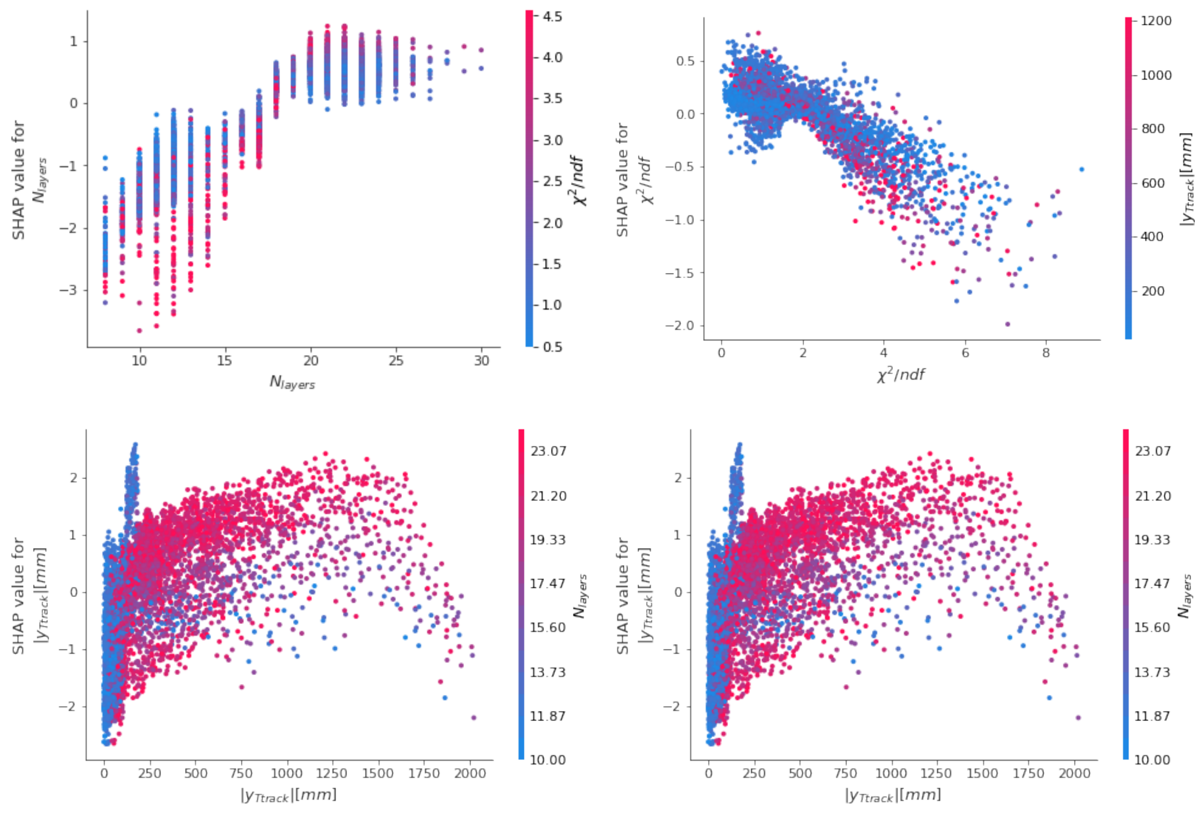


Figure 4.12.3: Shap dependence plots. Each dot is a track. The x-axis shows the value of the **feature 1** and the y-axis is the Shap value attributed to this feature. To show the dependence between **feature 1** and **feature 2** the color of particular dot depends of the value of the **feature 2**.

The result of the evaluation of the third and final approach, the one that is based on LIME model, is shown in Figure 4.12.6. The presented LIME outcome were obtained for two randomly chosen test examples. Each of these examples represents one of the target classes. It provides a good indication of the importance of the features and highlights why a particular decision was made. The length of each bar is proportional to the feature's value times linear regression weight associated with this feature. It is easy to note that one of the most important features is the number of hits, that were used to reconstruct T-Seeds, and this is consistent with the initial intuition.

4.13 Final machine learning model to select the best downstream track candidates

In the final stage of the downstream tracking pattern recognition algorithm, a neural network is used to find the best downstream track candidate for a given input T-Seed. The network used is a fully-connected neural network with one hidden layer built using fifteen nodes. A rectified linear unit was used as an activation function for the hidden layer. For the output layer, a sigmoid function was applied. The following input variables were used:

- $\log(\chi^2/ndf)$: Natural logarithm of track χ^2 given by the fit
- $\log(p)$: Natural logarithm of the total momentum
- $\log(p_t)$: Natural logarithm of the transverse momentum
- Δp: Difference between momentum estimate from the T track and final momentum estimate of downstream track candidate.
- $\log(|\Delta x \text{ magnet}|)$: Natural logarithm of x displacement with respect to the point in the magnet after the fit
- $\log(|\text{dist}_{\text{initial}}|)$: Natural logarithm of the distance of the hits with respect to the initial track estimate
- $\log(|y_{track}(0)|)$: Natural logarithm of the absolute y position at $z = 0$
- $\log(|x_{track}(0)|)$: Natural logarithm of the absolute x position at $z = 0$
- # fired Layers: Number of layers in the TT with hits on this track

The network was trained on simulated $D^{*+} \rightarrow D^0\pi^+$ decays. The sample was chosen, such that also tracks from D decays are considered, which tend to have a softer momentum spectrum than tracks from B decays and are more similar to low-momentum background tracks. Only downstream reconstructible tracks that are not electrons were used as signal events. As a background sample, tracks that could not

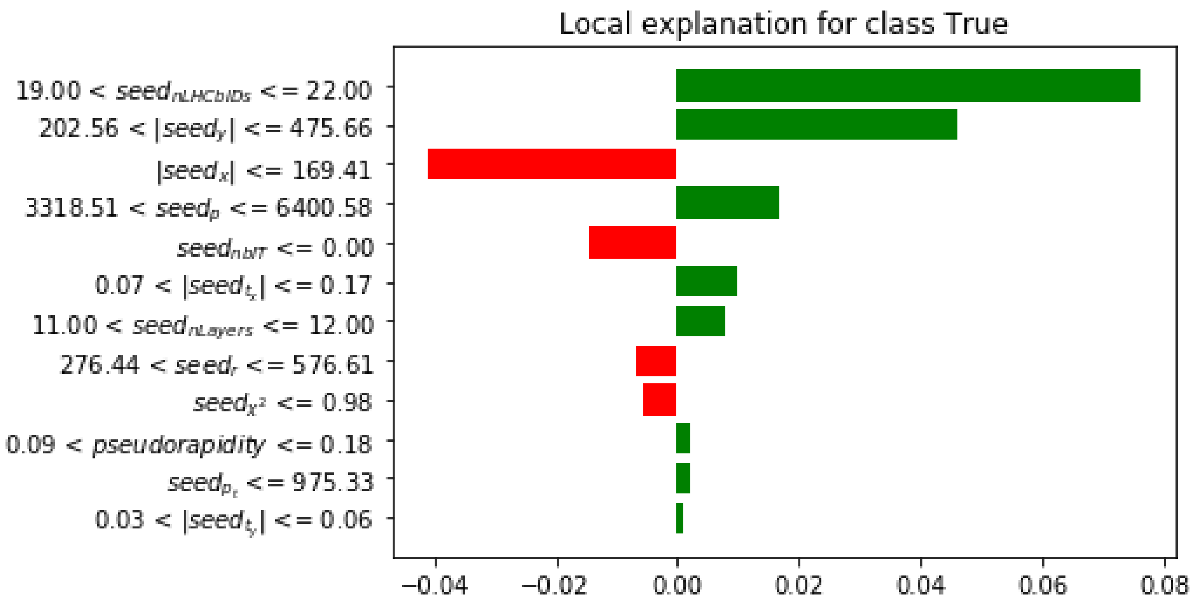


Figure 4.12.4: LIME explanation for randomly chosen True example

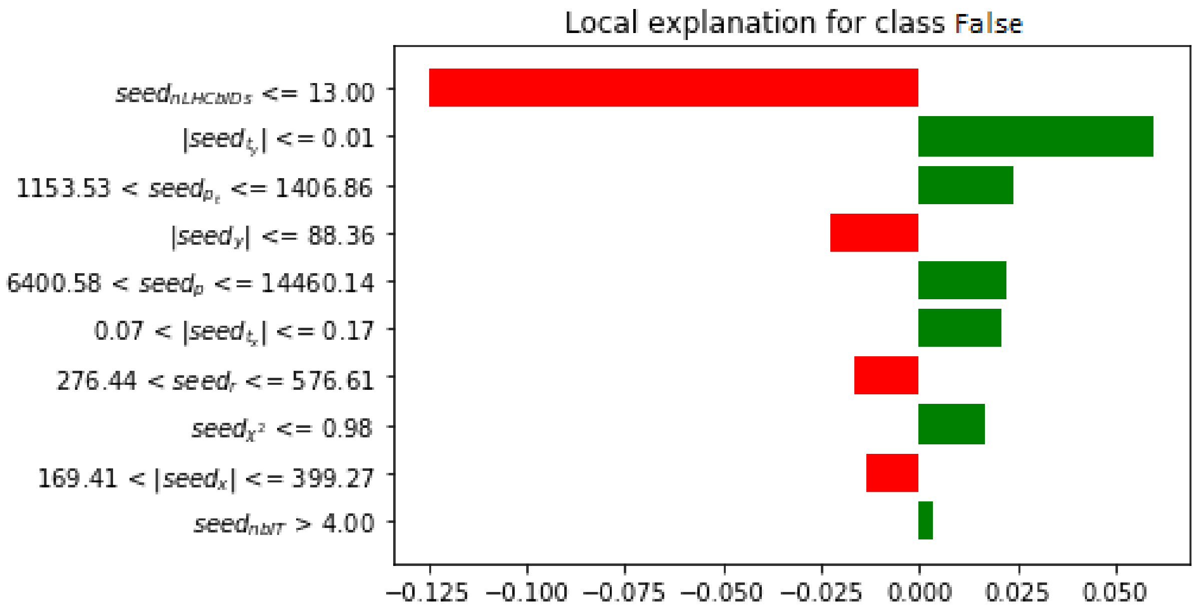


Figure 4.12.5: LIME explanation for randomly chosen Ghost example

Figure 4.12.6: Exemplary outcome of the LIME for two randomly chosen examples. Those samples were drawn from the test dataset. The red color indicates that the features pushed prediction toward ghost.

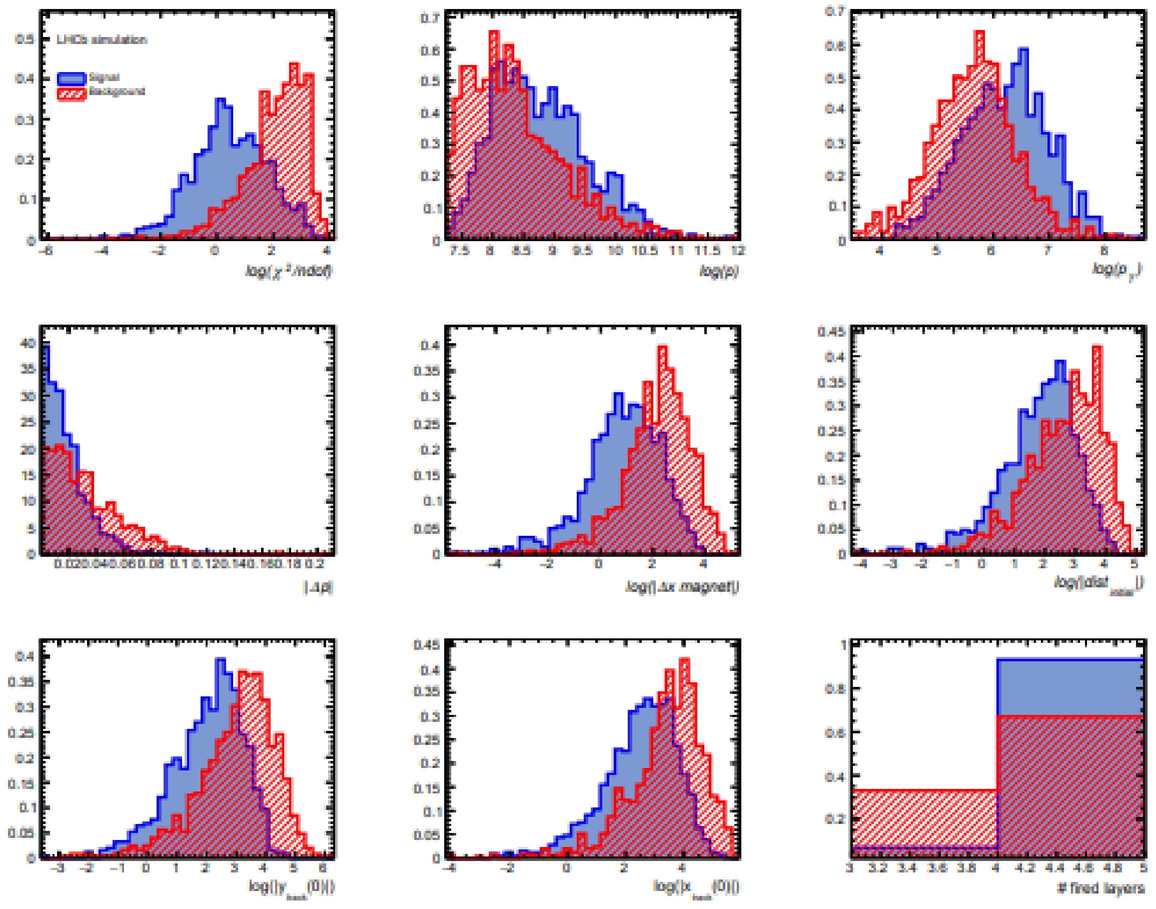


Figure 4.13.1: Input variables for the training of a Fully-connected neural network to select the best downstream track candidates. The training dataset consists of $D^{*+} \to D^0\pi^+$ decays. The blue solid histogram is the signal distribution, while the hashed red distribution is the background.

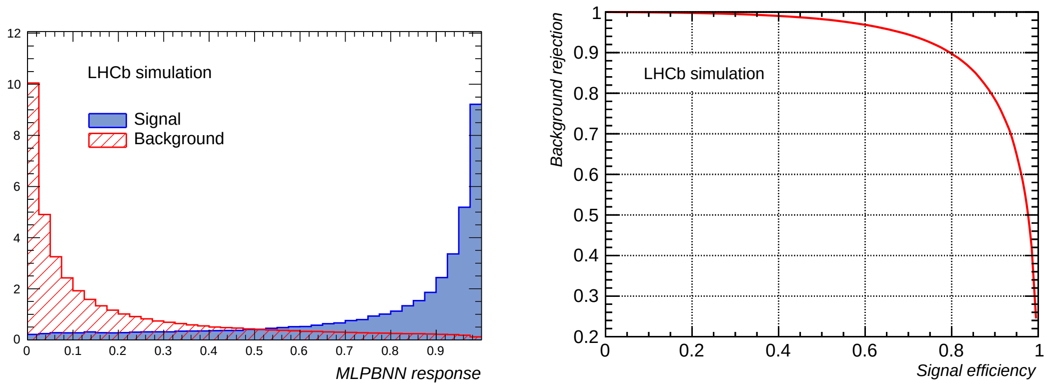


Figure 4.13.2: Left: Classifier output distribution, blue is signal, red is background. The default cut is at 0.1. Right: ROC curve for the fully-connected in LongLivedTracking algorithm. Both are obtained on a sample of $D^{*+} \to D^0\pi^+$ decays.

be matched to a single simulated particle were used. The distributions of input variables are shown in Figure 4.13.1. The classifier output distribution and the ROC curve is shown in Figure 4.13.2.

A candidate is selected out if its probability of being a true downstream track is larger than a specific tunable threshold.

4.14 Physics Performance of the modernised downstream tracking

The previous sections describe the LongLived pattern recognition algorithm. This one is dedicated to showing the physics performance of this algorithm. The performance of the track reconstruction algorithm can be measured using various methods. The most intuitive one is to measure the efficiency of reconstructing a charged particle in the LHCb acceptance as a downstream track using Monte Carlo simulated samples. The second one evaluates the algorithm's performance via analysis based on real data collected by the detector. This approach determines and compares the pattern recognition algorithms performance by measuring the reconstructed long-lived particle invariant mass resolution.

4.14.1 Monte Carlo based Downstream Tracking efficiency

The pattern recognition algorithm performance measurement can be determined from Monte Carlo simulations by comparing the number of tracks that the algorithm managed to find (reconstructed tracks) with the maximum number of tracks that it could possibly find (reconstructable tracks).

To discuss the efficiency of the pattern recognition algorithm it is obligatory to define a number of quantities that are used in this study. First of all, the MC particle is said to be **reconstructible**, if it left the sufficient number of hits in the detector. The particle reconstructible criteria vary for one subdetector to another [118], and they are summarized here:

- Velo reconstructible is a particle that has at least three hits in R and ϕ sensors.
- TT reconstructible particle is consider when it has at least one hit in the first two planes (TTa) and one hit in the last two planes (TTb).
- T-Station reconstructible is a particle that has at least one x and one stereo hit in each of the three T-Stations.

Therefore a particle is considered reconstructible as a Downstream track if it satisfies the TT and T-Stations reconstructibility criteria. In order to associate the reconstructed tracks to the reconstructable ones a cross-check of hits in both of them has to be done. A reconstructed track is considered as matched to a simulated Monte Carlo particle if they share at least 70% of the hits. Based on this definition the following pattern recognition performance metrics can be defined:

- The **Tracking efficiency** (ε) is defined as a ratio between the number of reconstructed and matched tracks to the total number of reconstructable tracks.

$$\varepsilon = \frac{\text{reconstructed} \cap \text{matched}}{\text{reconstructable}} \tag{4.25}$$

- The **ghost rate** is the amount of reconstructed tracks not associated to any of the Monte Carlo particle, thus having less than 70% of matched hits, with respect to the total amount of tracks found by the pattern recognition algorithm

$$\text{ghost rate} = \frac{\text{not matched}}{\text{reconstructable}} \tag{4.26}$$

Within the scope of this analysis the efficiency of a Downstream Tracking algorithm is calculated as the one that includes the efficiency of a Long-Lived Tracking pattern recognition algorithm combined together with the efficiency of the T-Seed reconstruction.

In the presented studies, the analysis was performed on a sample constituted of 50,000 simulated events of both magnet polarities. Two types of decays were simulated: $B^0 \rightarrow J/\Upsilon K_S^0$ and $D^{*+} \rightarrow D^0\pi^+$. The first type of decay was selected to represent B meson decays and the second one for a charm. There are four different categories of tracks that were considered within the scope of this performance analysis on MC data:

- ε_{TT+T} : efficiency for all downstream reconstructible particles;
- $\varepsilon_{TT+T,p>5Gev/c}$: efficiency for all downstream reconstructible particles with momentum $p > 5GeV/c$;
- $\varepsilon_{TT+T,fromB/D}$: efficiency for all downstream reconstructible particles from a decay of B or D meson;
- $\varepsilon_{TT+T,fromB/D,p>5GeV}$: efficiency for all downstream reconstructible particles from a decay of B or D meson with momentum $p > 5GeV/c.$;

The overall efficiency numbers are collected in upper half of Table 4.14.1 and the ghost rate related numbers are presented in Table 4.14.2 (upper half). Plots of the efficiency and the ghost rate as a function of momentum, transverse momentum, and pseudorapidity are shown in Figures 4.14.1, 4.14.2 (obtained using $B^0 \rightarrow J/\Upsilon K_S^0$) and 4.14.5, 4.14.7 (for $D^{*+} \rightarrow D^0\pi^+$ decay). No uncertainty is given, because presented numbers were obtained on a Monte Carlo samples not on a collision data. The statistical uncertainty are negligible, at the permille level.

It was also important to measure the downstream tracking efficiency using the samples that were fitted with Kalman Filter and a quality cut on a $\chi^2 ndf$ was applied. Those tracks were also processed by a clone killing algorithm to remove duplicates, which were reconstructed as long and downstream track simultaneously. The performance numbers referring to those tracks are collected in Table 4.14.1(lower half), and Table 4.14.2 (lower half) gives ghost rates. The performance numbers after these three steps seem to be worse than before, as the processing steps applied also reject correct tracks, and the clone killer is inefficient. One of the reasons why it happens, the clone killer sometimes falsely classifies downstream tracks as long tracks (for example when the Velo part is a ghost), which then gets removed from the downstream statistics. Furthermore, clone killer prefers long tracks over downstream tracks.

Downstream tracking efficiency strongly depends on the momentum and transverse momentum of the tracks. The explanation of the phenomenon can be based on factors such as the search windows do not cover the full region necessary to find all tracks. Therefore, for the low momentum tracks, search windows are generally larger than for high momentum tracks, which increases the number of hits within the search window, thus increases the chances that the pattern recognition identifies the wrong hits.

In addition, the downstream tracks that were also reconstructed as long tracks are less prone to be ghosts, which is why the ghost fraction actually increases after the Kalman Filter, the clone killer and the ghost probability cut. However, it should be noted that in a physics analysis of a decay channel, a strict cut is normally placed on the ghost probability variable of the downstream tracks, which reduces the ghost fraction significantly.

Table 4.14.1: Reconstruction efficiency of downstream tracks on simulated samples of $B^0 \to J/\Upsilon K^0_S$ and $D^{*+} \to D^0\pi^+$ decays. The upper half are non-filtered tracks, the lower half is after the clone killer, Kalman Filter and ghost probability cut. This efficiency includes the efficiency of T-Seed reconstruction and efficiency of the LongLived Tracking algorithm. Due to a softer momentum spectrum, the efficiency is lower in the $D^{*+} \to D^0\pi^+$ sample.

filter	decay type	ε_{TT+T}	$\varepsilon_{TT+T,\, p>5GeV}$	$\varepsilon_{TT+T,\, \text{from}B/D}$	$\varepsilon_{TT+T,\, \text{from}B/D,\, p>5GeV}$
No	$B^0 \to J/\Upsilon K^0_S$	73.3%	80.1%	81.4%	85.4%
No	$D^{*+} \to D^0\pi^+$	71.3%	78.0%	76.8%	81.4%
Yes	$B^0 \to J/\Upsilon K^0_S$	70.0%	76.7%	79.0%	83.2%
Yes	$D^{*+} \to D^0\pi^+$	67.3%	73.7%	73.1%	77.3%

Table 4.14.2: Ghost fraction of downstream tracks on simulated samples of $B^0 \to J/\Upsilon K^0_S$ and $D^{*+} \to D^0\pi^+$ decays. The upper half are non-filtered tracks, the lower half is after the clone killer, Kalman Filter and ghost probability cut. This ghost fraction includes the ghosts produced in T-Seed pattern Recognition and Long-Lived Tracking. Due to a softer momentum spectrum, the ghost fraction is higher in the $D^{*+} \to D^0\pi^+$ sample.

filter	decay type	fraction of ghosts
No	$B^0 \to J/\Upsilon K^0_S$	29.5%
No	$D^{*+} \to D^0\pi^+$	30.3%
yes	$B^0 \to J/\Upsilon K^0_S$	39.2%
yes	$D^{*+} \to D^0\pi^+$	40.2%

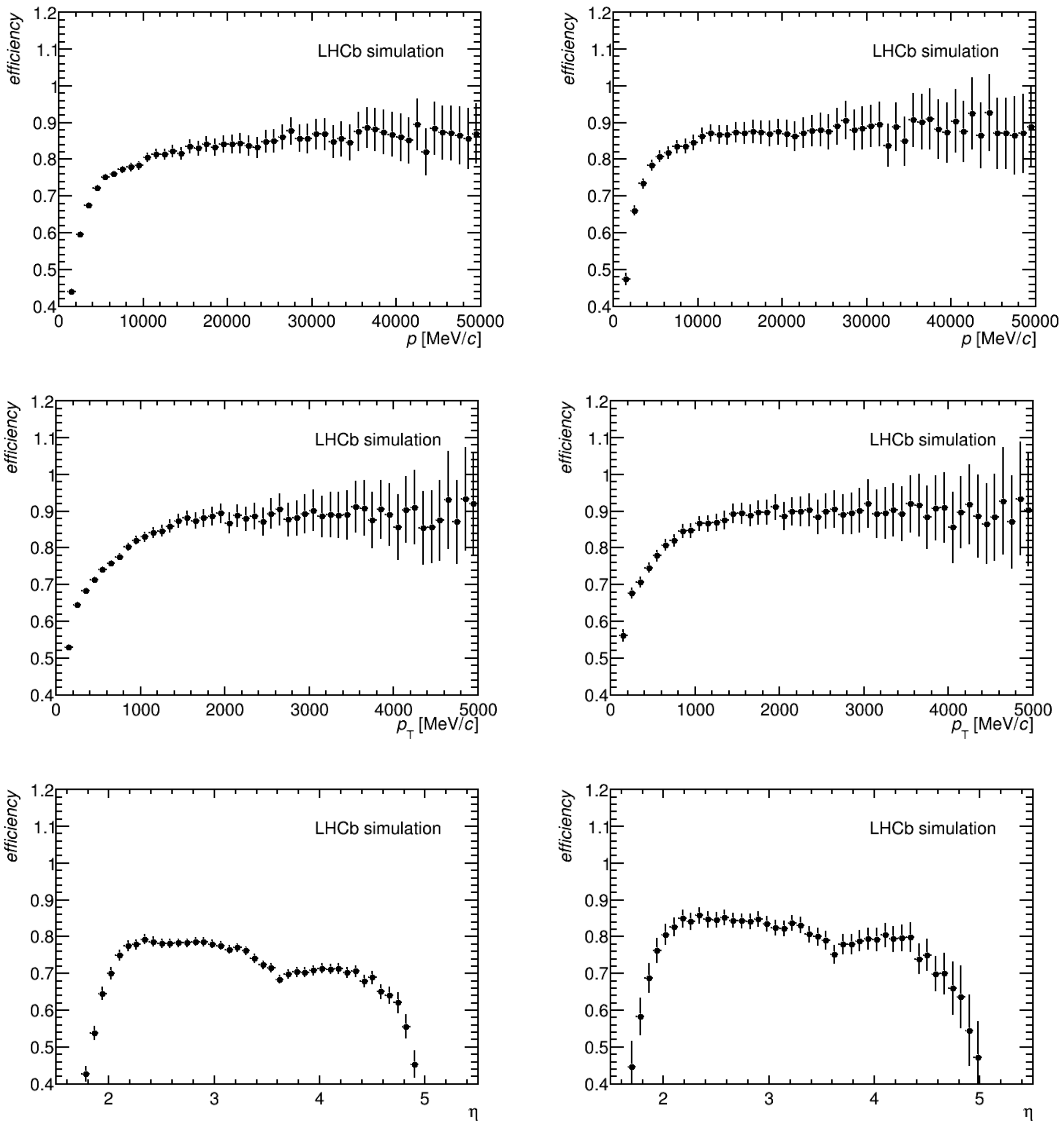


Figure 4.14.1: Efficiencies to reconstruct downstream tracks as a function of momentum (top row), transverse momentum (middle row) and pseudorapidity (bottom row). The left column is for all downstream reconstructible tracks, the right column for all downstream tracks from a decay chain of a B or D meson. The efficiencies are obtained on a simulated sample of $B^0 \to J/\Psi K_S^0$ decays.

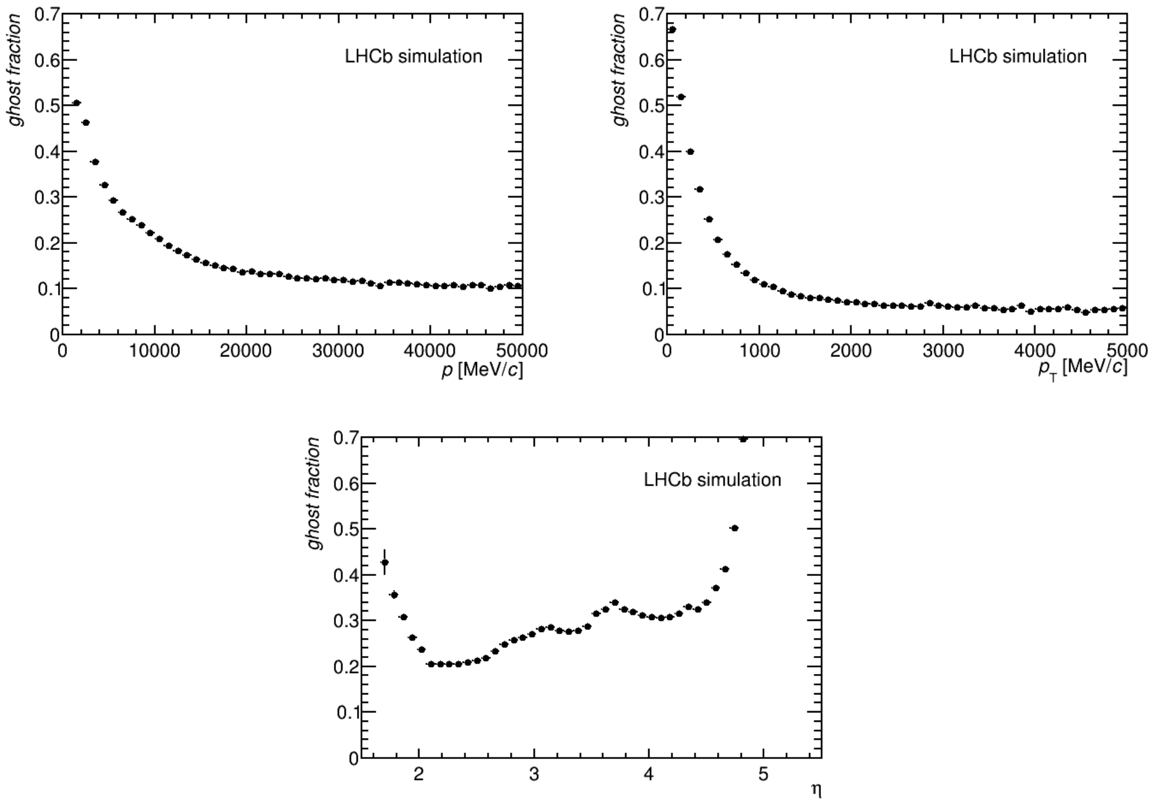


Figure 4.14.2: Ghost fraction in downstream tracks as a function of momentum (top left), transverse momentum (top right) and pseudorapidity (bottom). The ghost fractions are obtained on a simulated sample of $B^0 \to J/\Psi K_S^0$ decays.

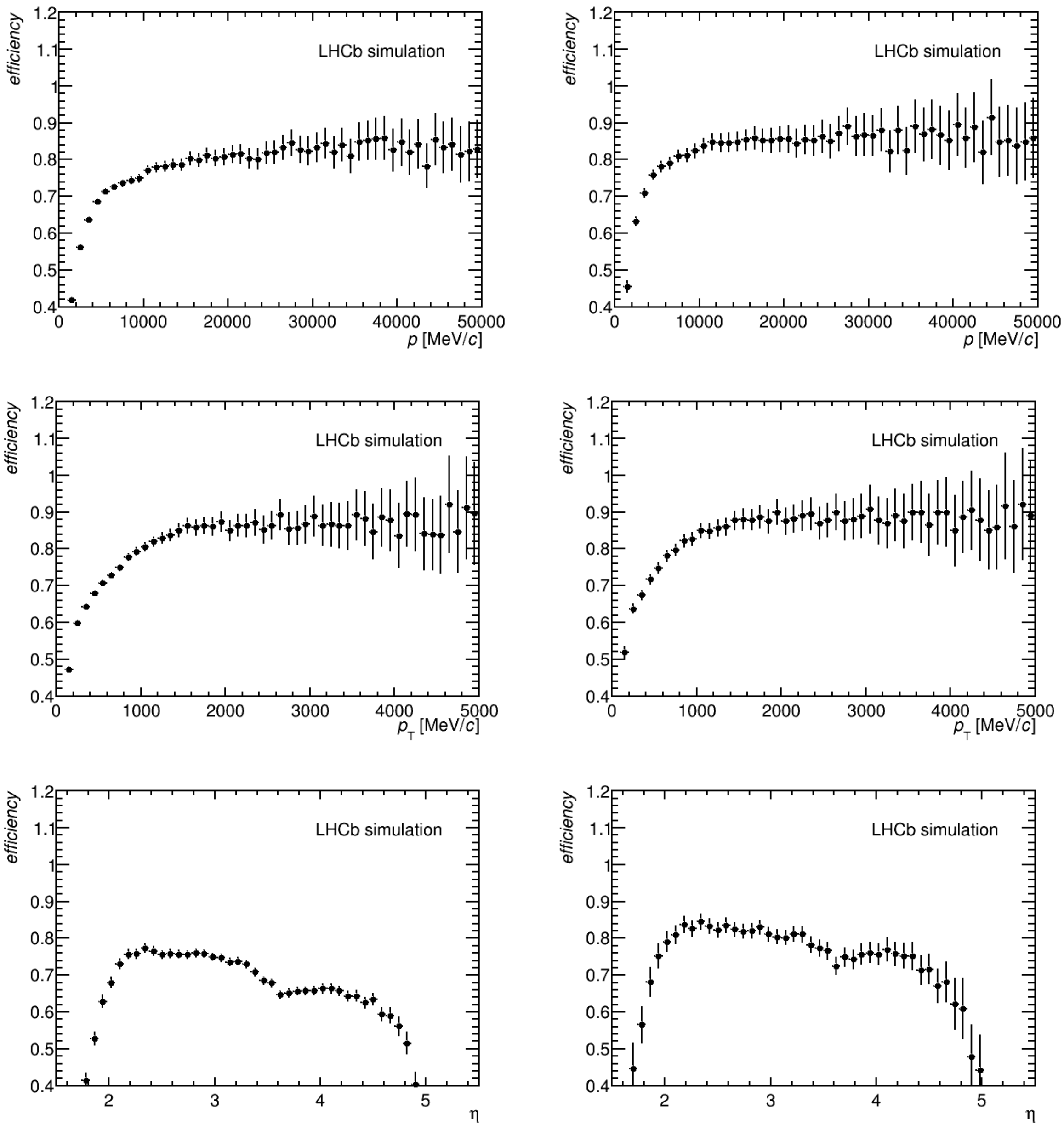


Figure 4.14.3: Efficiencies to reconstruct downstream tracks, after clone killing, Kalman filtering and the cut on the ghost probability, as a function of momentum (top row), transverse momentum (middle row) and pseudorapidity (bottom row). The left column is for all downstream reconstructible tracks, the right column for all downstream tracks from a decay chain of a B or D meson. The efficiencies are obtained on a simulated sample of $B^0 \to J/\Psi K^0_S$ decays.

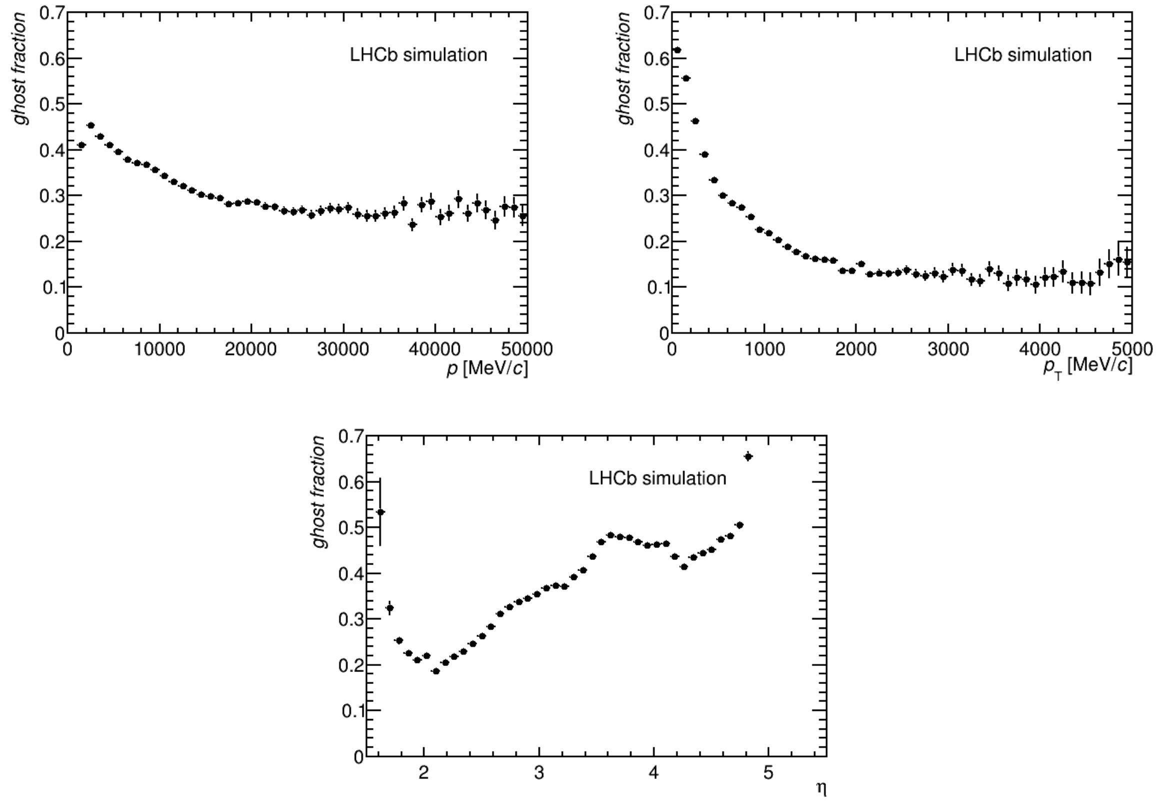


Figure 4.14.4: Ghost fraction in downstream tracks, after clone killing, Kalman filtering and the cut on the ghost probability, as a function of momentum (top left), transverse momentum (top right) and pseudorapidity (bottom). The ghost fractions are obtained on a simulated sample of $B^0 \to J/\Psi K_S^0$ decays.

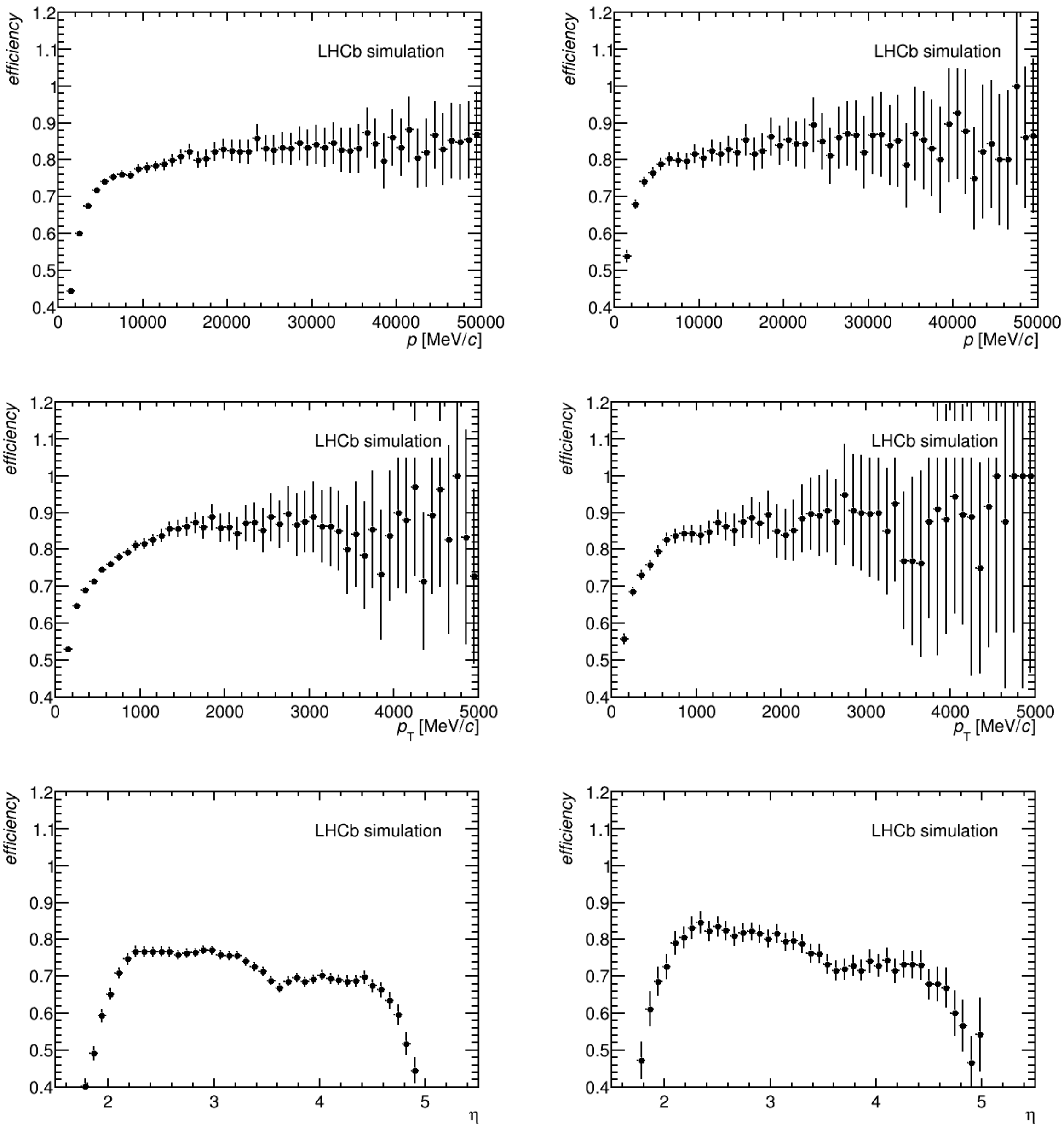


Figure 4.14.5: Efficiencies to reconstruct downstream tracks as a function of momentum (top row), transverse momentum (middle row) and pseudorapidity (bottom row). The left column is for all downstream reconstructible tracks, the right column for all downstream tracks from a decay chain of a B or D meson. The efficiencies are obtained on a simulated sample of$D^{*+} \to D^0\pi^+$ decays.

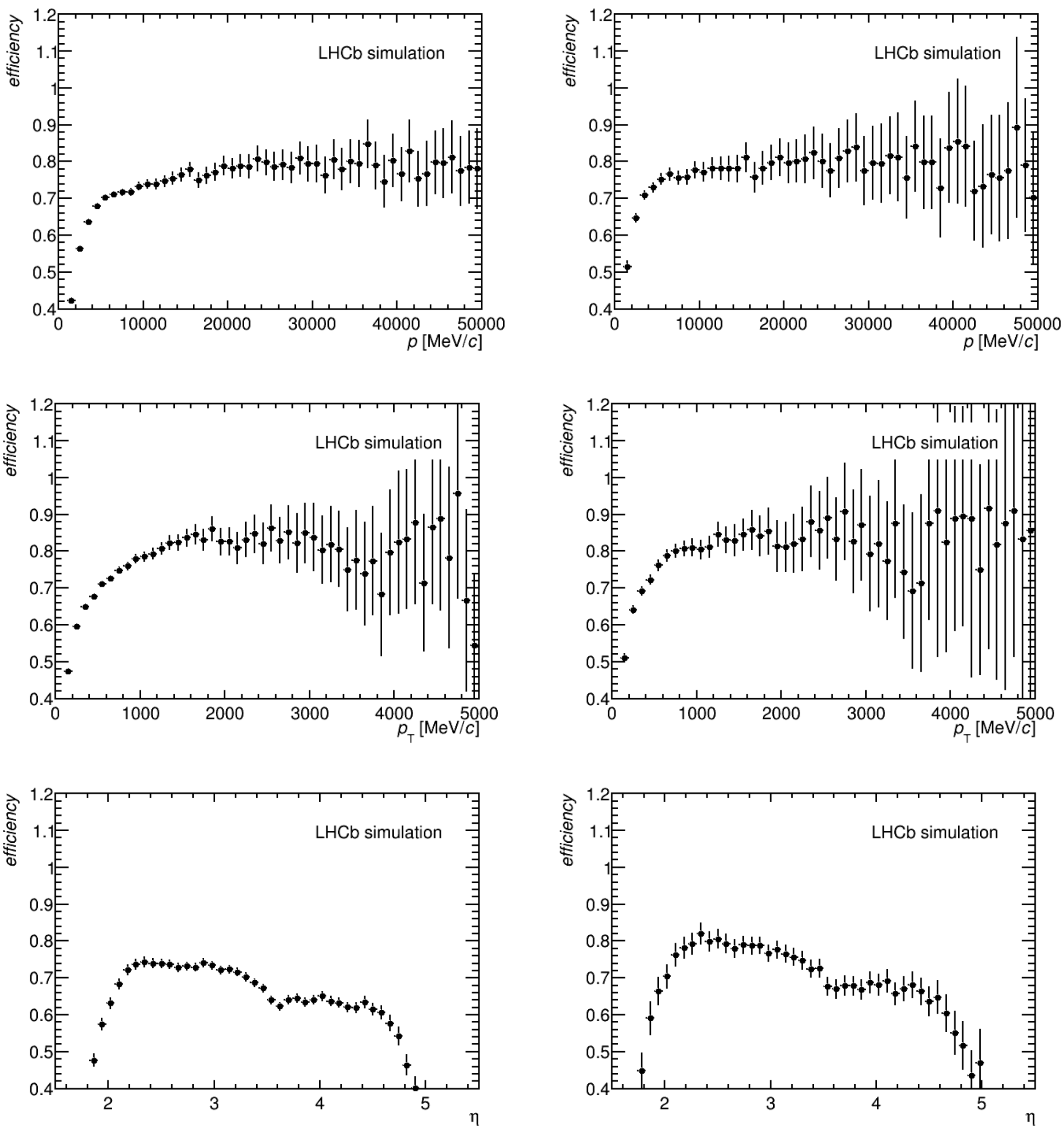


Figure 4.14.6: Efficiencies to reconstruct downstream tracks, after clone killing, Kalman filtering and the cut on the ghost probability, as a function of momentum (top row), transverse momentum (middle row) and pseudorapidity (bottom row). The left column is for all downstream reconstructible tracks, the right column for all downstream tracks from a decay chain of a B or D meson. The efficiencies are obtained on a simulated sample of $D^{*+} \to D^0\pi^+$ decays.

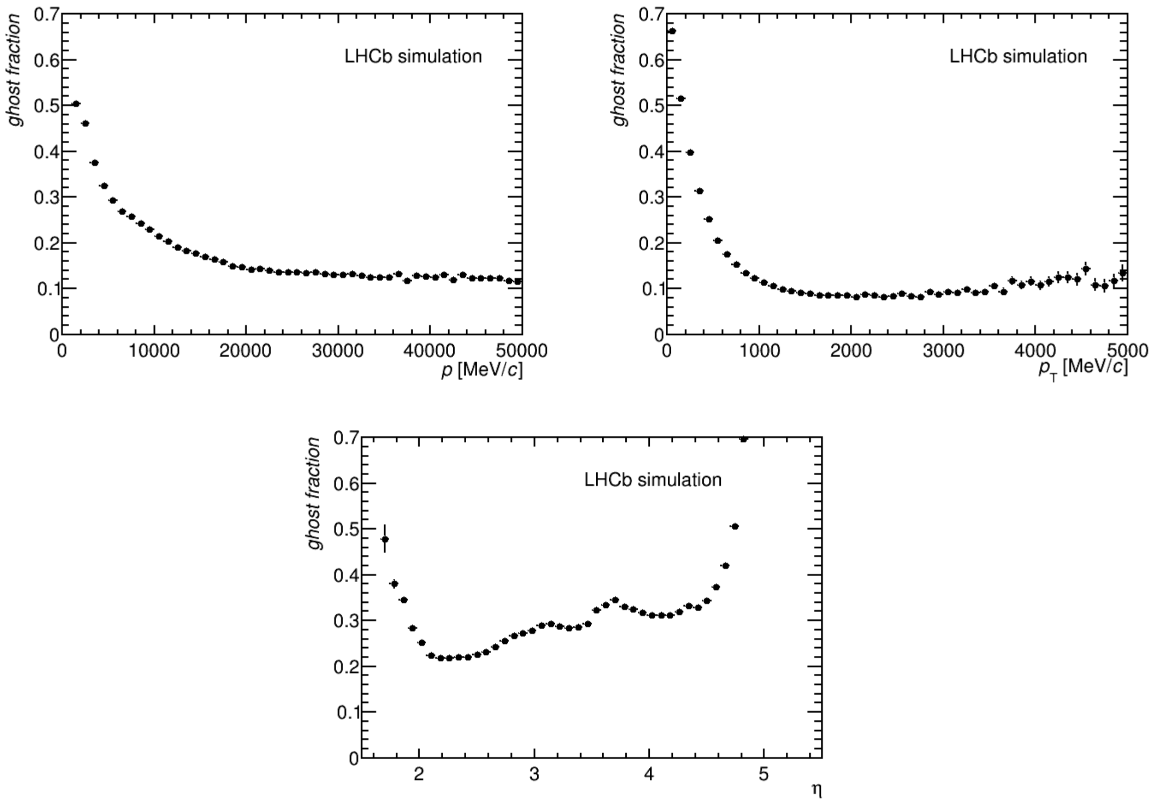


Figure 4.14.7: Ghost fraction in downstream tracks as a function of momentum (top left), transverse momentum (top right) and pseudorapidity (bottom). The ghost fractions are obtained on a simulated sample of $D^{*+} \to D^0\pi^+$ decays.

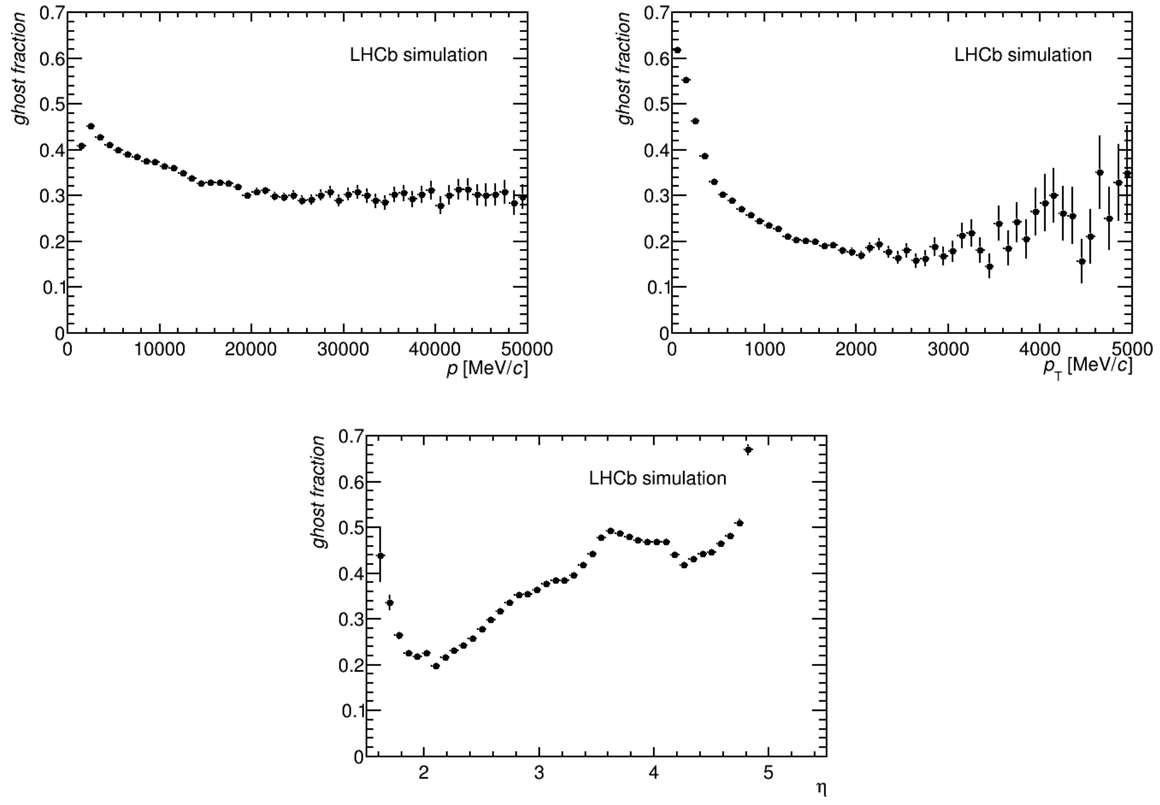


Figure 4.14.8: Ghost fraction in downstream tracks, after clone killing, Kalman filtering and the cut on the ghost probability, as a function of momentum (top left), transverse momentum (top right) and pseudorapidity (bottom). The ghost fractions are obtained on a simulated sample of $D^{*+} \to D^0\pi^+$ decays.

4.14.2 Comparison between Long-Lived Tracking algorithm and its predecessor

The analysis presented within this section is dedicated to comparing the performance of the Long-Lived Tracking algorithm and its predecessor called PatDownstream [40]. The corresponding performance numbers are given in Tables 4.14.3, 4.14.4, and 4.14.5.

Figures 4.14.9 and 4.14.10 show the efficiency and ghost rates of downstream tracks reconstructed via these two algorithms as a function of momentum, transverse momentum, and pseudorapidity. Those plots visualize tracking algorithm performance that was obtained using a simulated sample of B decays and no other processing was applied.

Figures 4.14.11 and 4.14.12 present the same quantities, after Kalman filtering and clone killer. It is clearly visible, that the performance gain and the strong reduction of ghost rates were measured for the new algorithm. It is presumed that the significant ghost rate reduction (about 16%) was achieved due to the good performance of the T-Seed classifier, which was one of the major goals of the author's study.

Table 4.14.3: Comparison of reconstruction efficiency of downstream tracks made with Long-Lived Tracking algorithm or PatDownstream, on simulated samples of $B^0 \rightarrow J/\Psi K^0_S$. This efficiency includes the efficiency of PatSeeding and Long-Lived Tracking algorithm.

algorithm	ε_{TT+T}	$\varepsilon_{TT+T,\ p>5GeV}$	$\varepsilon_{TT+T,\ \mathrm{from}B/D}$	$\varepsilon_{TT+T,\ \mathrm{from}B/D,\ p>5GeV}$
Long-Lived Tracking	73.3%	80.1%	81.4%	85.4%
PatDownstream	68.3%	74.4%	77.1%	81.6%

Table 4.14.4: Comparions of reconstruction efficiency of downstream tracks made with PatLongLivedTracking or PatDownstream, on simulated samples of $B^0 \rightarrow J/\Psi K^0_S$, after the Kalman Filter and clone killer. This efficiency includes the efficiency of PatSeeding and PatLongLivedTracking. The efficiency decreases compared to Table 4.14.3 due to inefficiency of the clone killer.

algorithm	ε_{TT+T}	$\varepsilon_{TT+T,\ p>5GeV}$	$\varepsilon_{TT+T,\ \mathrm{from}B/D}$	$\varepsilon_{TT+T,\ \mathrm{from}B/D,\ p>5GeV}$
Long-Lived Tracking	70.0%	76.7%	79.0%	83.2%
PatDownstream	63.7%	71.3%	74.5%	79.2%

Table 4.14.5: Ghost fraction of downstream tracks on simulated samples of $B^0 \rightarrow J/\Psi K^0_S$ for PatLongLivedTracking and PatDownstream, once before and after the clone killer, Kalman Filter and ghost probability (CKG) were applied. This ghost fraction includes the ghosts produced in PatSeeding and PatLongLivedTracking (PatDownstream). The ghost fraction is higher after the clone killer as tracks, also reconstructed as long tracks, are not present in that number.

algorithm	fraction of ghosts	fraction of ghosts after CKG
Long-Lived Tracking	29.5%	39.2%
PatDownstream	46.3%	52.1%

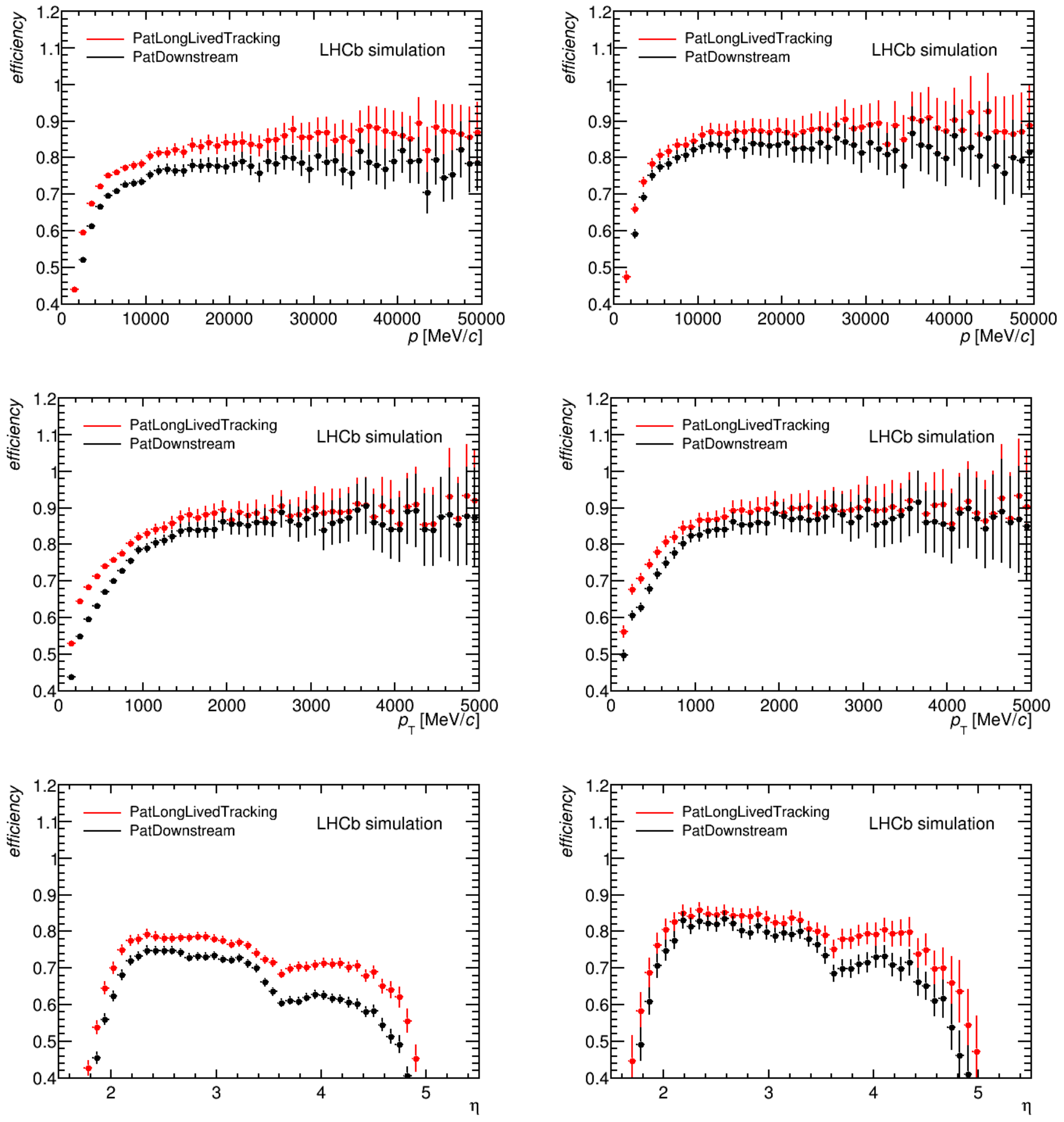


Figure 4.14.9: Efficiencies to reconstruct downstream tracks as a function of momentum (top row), transverse momentum (middle row) and pseudorapidity (bottom row), in red for Long-Lived Tracking algorithm and in black for PatDownstream. The left column is for all downstream reconstructible tracks, the right column for all downstream tracks from a decay chain of a B or D meson. The efficiencies are obtained on a simulated sample of $B^0 \to J/\Psi K_S^0$ decays.

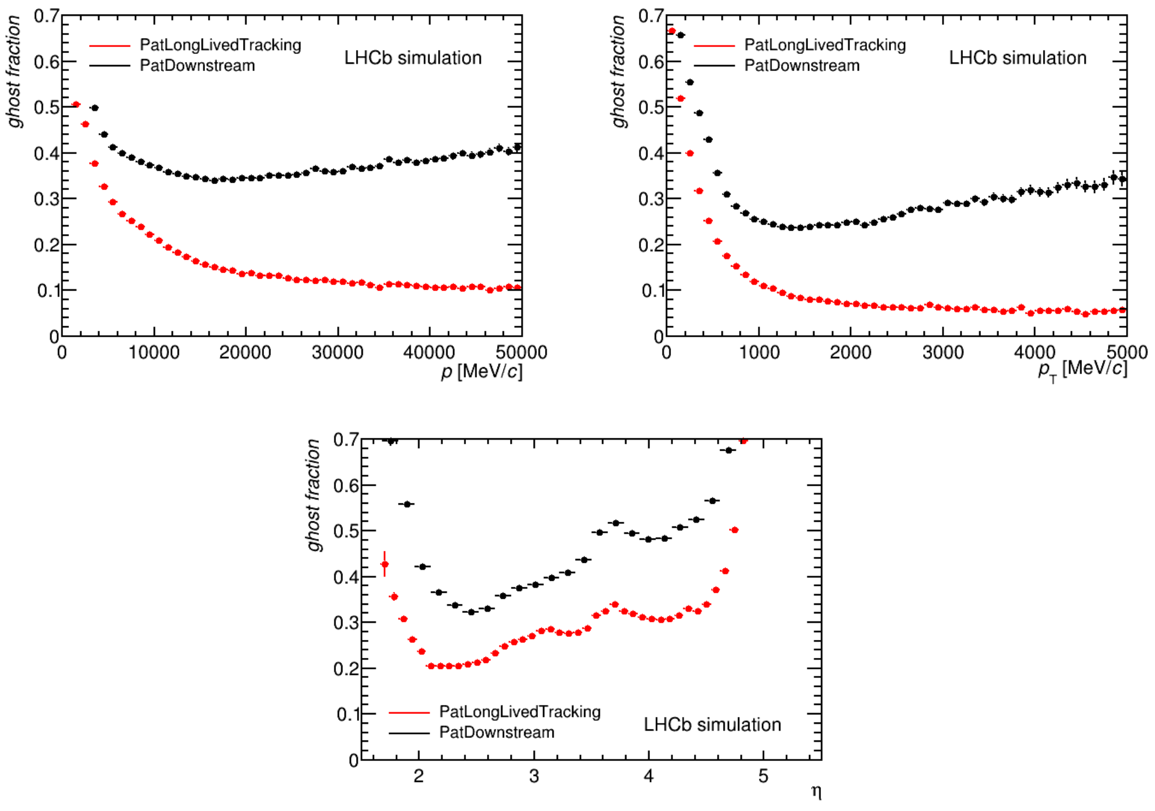


Figure 4.14.10: Ghost fraction in downstream tracks as a function of momentum (top left), transverse momentum (top right) and pseudorapidity (bottom), in red for Long-Lived Tracking algorithm and in black for PatDownstream. The ghost fractions are obtained on a simulated sample of $B^0 \to J/\Psi K_S^0$ decays.

4.14.3 Performance measured using collision data

The improvements introduced to the pattern recognition algorithm should be reflected in improvements to the reconstruction of long-lived particles. To investigate this, the reconstruction of K_s and Λ_0 in minimum bias events [7] were analyzed.

4.14.3.1 Event selection

This analysis focuses on the algorithm's relative event yield difference between the new algorithm and the baseline. Thus the selection was taken from the standard LHCb software. The following list contains an explanation of each cut applied to select the data that were used to reconstruct both K_s and Λ_0:

- $p(\pi_1, \pi_2 \vee p)$ - minimal momentum of daughter particles;
- $p_t(\pi_1, \pi_2 \vee p)$ - minimal transverse momentum of daughter particles;

[7] The minimum bias samples are those which try to reproduce proton-proton collisions as close to the reality as possible, with no bias from restricted trigger conditions.

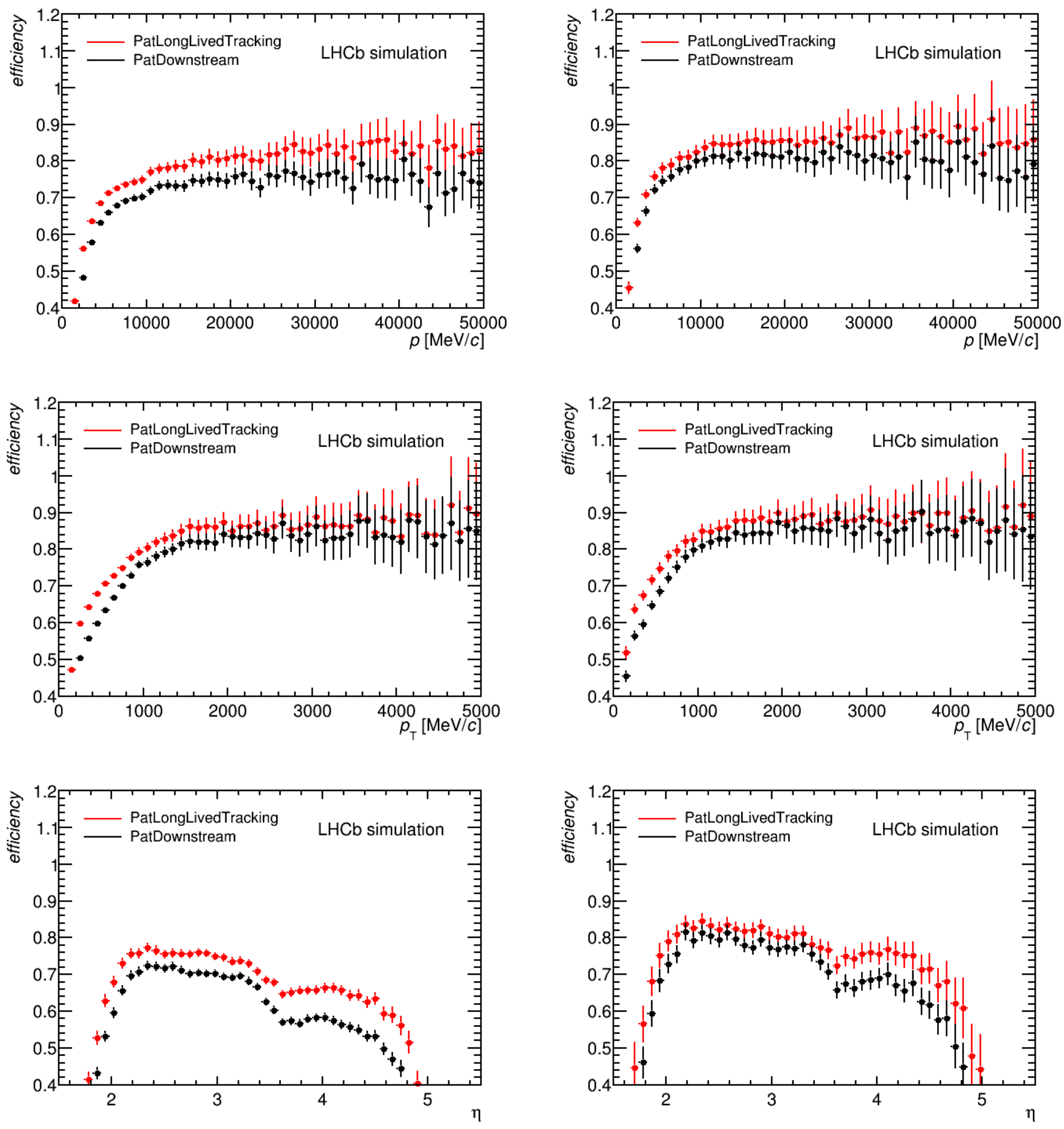


Figure 4.14.11: Efficiencies to reconstruct downstream tracks, after clone killing, Kalman filtering and the cut on the ghost probability, as a function of momentum (top row), transverse momentum (middle row) and pseudorapidity (bottom row). The left column is for all downstream reconstructible tracks, the right column for all downstream tracks from a decay chain of a B or D meson. The efficiencies are obtained on a simulated sample of $B^0 \to J/\Psi K^0_S$ decays.

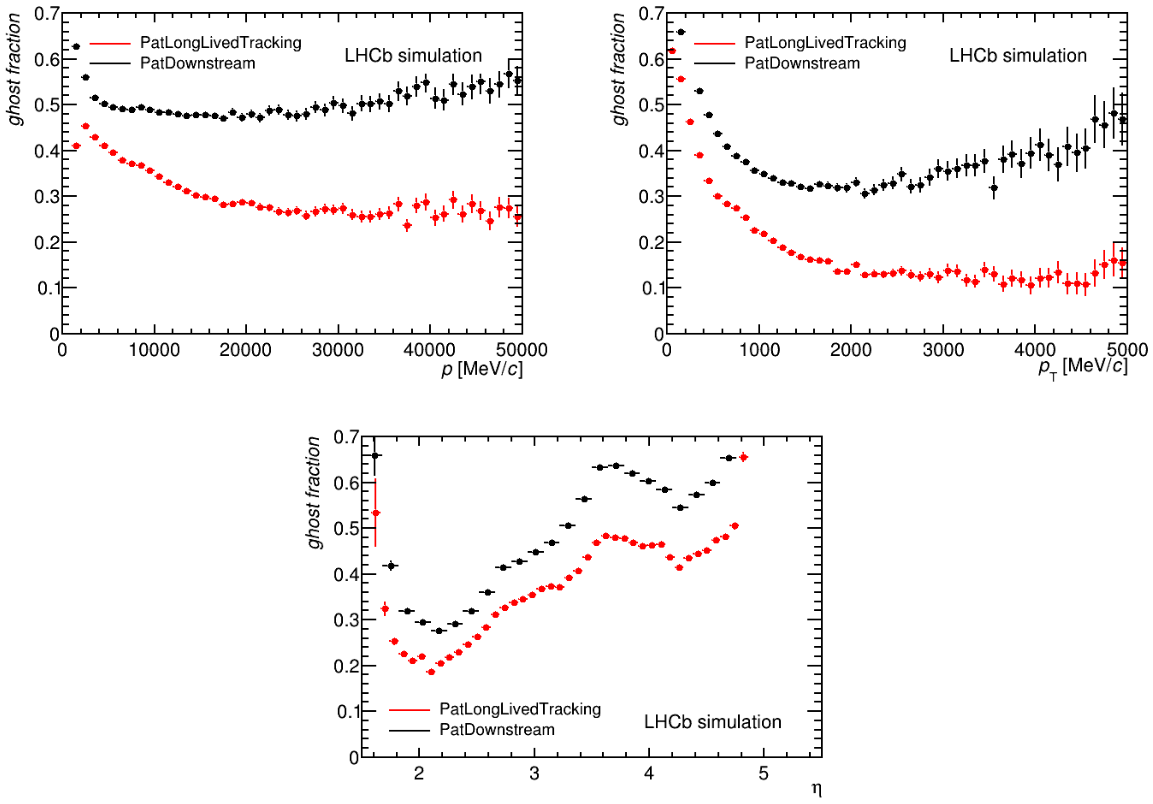


Figure 4.14.12: Ghost fraction in downstream tracks, after clone killing, Kalman filtering and the cut on the ghost probability, as a function of momentum (top left), transverse momentum (top right) and pseudorapidity (bottom). The ghost fractions are obtained on a simulated sample of $B^0 \to J/\Psi K^0_S$ decays.

- $\chi^2 ndf$ of $\pi \vee p$ tracks - maximal $\chi^2 ndf$ of the daughter tracks;
- track type - the type of track that were taken into consideration;
- $\chi^2 ndf\,(K_S \vee \Lambda_0)$ - maximal $\chi^2 ndf$ of the vertex fit;
- $\Delta M(\pi_1 + \pi_2 || p)$ - the difference between reconstructed invariant mass of the mother particle and its mass taken from the Particle Data Group (PDG) repository [135]. This invariant mass is calculated by taking sum of 4-momenta vectors [8] of the two daughter particle tracks;
- $\Delta M(K_S \vee \Lambda_0)$ - the difference between reconstructed invariant mass and the PDG mass after vertex fit. The tracks has been properly propagated to the K_s or Λ_0 vertex.

Table 4.14.6: Selection criteria for K_s and Λ_0

cut variable	K_S	Λ_0
$p(\pi_1, \pi_2 \vee p)$ [MeV]	>2000	>2000
$p_t(\pi_1, \pi_2 \vee p)$ [MeV]	>50	>100
track type	Downstream	Downstream
$\chi^2 ndf$ of $\pi \vee p$	<4	<4
$\chi^2 ndf\,(K_S \vee \Lambda_0)$	<10	<10
$\Delta M(\pi_1 + \pi_2 \vee p)$ [MeV]	<100	<100
$\Delta M(K_S \vee \Lambda_0)$ [MeV]	<100	<100

The cuts for K_s and Λ_0 are given in Table 4.14.6. The sample that was used to perform this analysis consisted of about 100,000 events. Table 4.14.7 presents signal yield of K_s and Λ_0 and the background. In both cases, an increase in signal yield and background reduction (sample purity and yield) is visible. Figures 4.14.13 and 4.14.14 show the reconstructed invariant masses of a discussed particles. The model for mass distribution is a single Gaussian with an exponential background. Although a more complicated fit model would, in principle, be appropriate, e.g. a double Crystal Ball function, these fits were very unstable and showed problems converging. To avoid these complications and have a consistent mass model, a single Gaussian solution was chosen. As only ratios of event yields are considered, the error from using a single Gaussian only largely cancels. In case of Λ_0 decays the number of reconstructed particles increase by 7% and the number of background events was approximately 4% lower in favour of the new algorithm. More gains can be noted for the K_s decays reconstruction were the number of background events, when processed by the new algorithm, dropped by more than 11% and the sample purity increased by 22%. It should be noted that these studies were only performed as the sanity checks to confirm that the algorithm selecting tracks using computational intelligence can be better that the one based purely on statistical approach [9].

[8] 4 vector is a generalization of the classical three-dimensional momentum vector to four-dimensional space-time. The covariant form of this vector can be written as: $p = (p^0, p^1, p^2, p^3) = (\frac{E}{c}, p_x, p_y, p_z)$

[9] A number of full physics selections based on the new algorithm have been performed after the Run 2 concluded data taking. Results showed that the modernised trigger code improved long-lived particles reconstruction capabilities

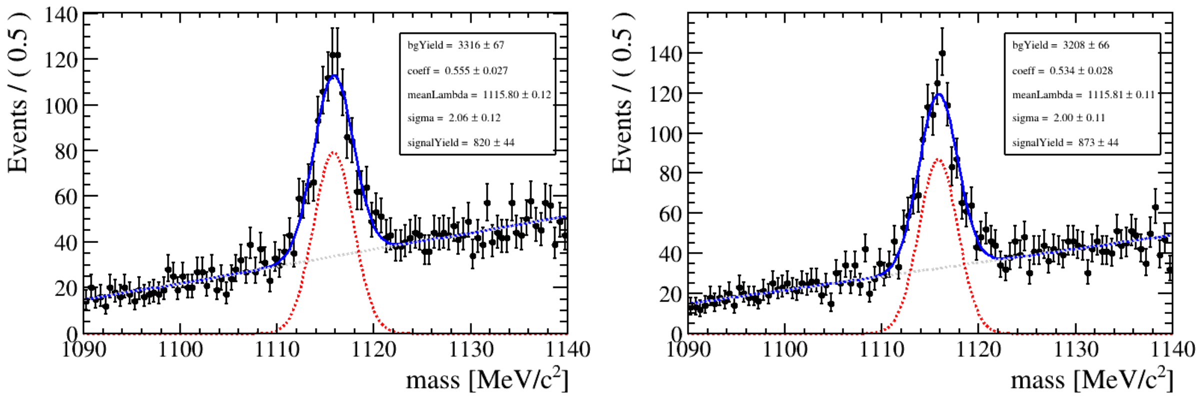


Figure 4.14.13: Invariant mass distribution of reconstructed $\Lambda_0 \rightarrow p^+ + \pi^-$ candidates. The left plot present results generated using the baseline version of the Downstream tracking reconstruction algorithm and the right plot is a similar graph for Long-Lived tracking algorithm. The red dashed line is the signal modeled as a Gaussian distribution, the gray dashed line shows the background and the straight blue line shows the combination.

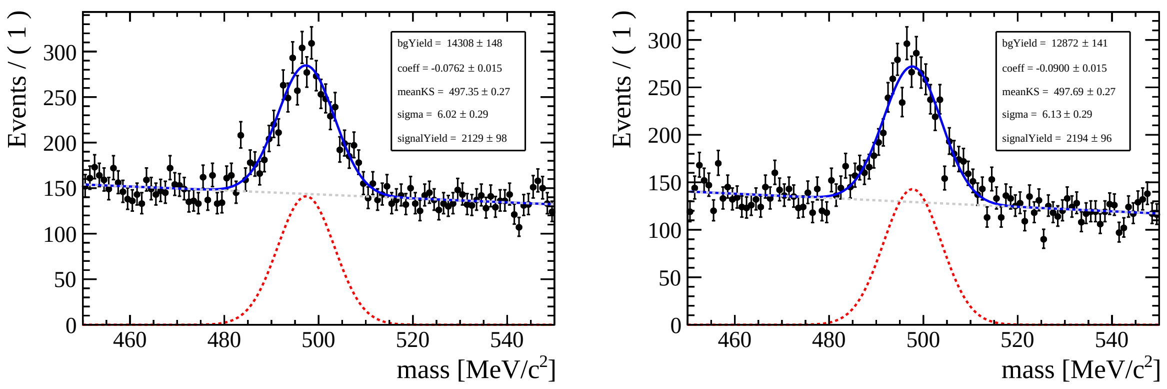


Figure 4.14.14: Invariant mass distribution of reconstructed $K_S \rightarrow \pi^+ + \pi^-$ candidates. The left plot present results generated using the baseline version of the Downstream tracking reconstruction algorithm and the right plot is a similar graph for Long-Lived tracking algorithm. The red dashed line is the signal modeled as a Gaussian distribution, the gray dashed line shows the background and the straight blue line shows the combination.

Table 4.14.7: Signal yield and background of reconstructed K_S and Λ_0 using minimum bias samples. "Original" algorithm refers to the one before improvements, the "new" consists of two Machine Learning classifiers.

version	decay	signal	background	S/B ratio [%]	Δ signal [%]
original	$K_S^0 \to \pi^+\pi^-$	2129 ± 98	14308 ± 148	14.9	0
new	$K_S^0 \to \pi^+\pi^-$	2194 ± 96	12872 ± 141	17	3
original	$\Lambda_0 \to p^+\pi^-$	820 ± 44	3316 ± 67	24.729	0
new	$\Lambda_0 \to p^+\pi^-$	873 ± 44	3208 ± 66	27.2	6.5

4.14.4 Tuning of the Downstream tracking algorithm

One of the key algorithm's parameters that has a strong influence on the pattern recognition performance is a T-Seed classification threshold. As described in section 3.3.6, its value was chosen to select more than 99% of true T-Seeds. This value corresponds to a very conservative classifier, which requires having extreme confidence to classify a given track as a ghost. As a result, the amount of suppressed background is reduced. Analyzing the ROC curve presented in Figure 4.10.3, one can conclude that devoting a tiny amount of signal data, the classifier can significantly reduce the background. This section is dedicated to fine-tuning the classification threshold. The presented analysis is based on measuring relative events yield for two channels, K_s and Λ_0. The selection criteria were described in section 4.14.3. The results obtained in various physics analyses as well as further simulation studies performed after the new algorithm has been commissioned proved that this very conservative approach should be changed for the Run 3 data taking period. On top of that it may be very beneficial to remove all the seed segments flagged as being a part of the long tracks from the input of the long-lived track reconstruction algorithm.

Table 4.14.8 presents the results of fine-tuning procedure. Those numbers were taken from the K_s and Λ_0 invariant mass fits, which are shown in Figures 4.14.15 and 4.14.16 respectively. Each of the rows represents a different classification threshold value. It is clearly visible that increasing the value of T-Seed classification threshold parameter allows significantly reduces the background while preserving almost all signal events. One can conclude that increasing this value from 0.07 to 0.2 allows to suppress 18% more background.

To summarize this study, increasing the value of the classification threshold to 0.2 allows reducing the background of about **26%** while keeping almost the same number of signal events, measured in K_s channel, with respect to the Long-Lived tracking predecessor.

4.14.5 Processing time

As the Long-Lived Tracking algorithm processing time depends significantly on a type of machine, that is used to run it, and the event multiplicity it is hard to give an absolute unbiased number. On a simulated signal sample for Run II on a single worker node at CERN, Long-Lived Tracking needs $O(5ms)$ to process an event. This time is not significant for the overall timing budget of HLT 2. Comparisons using callgrind show a decrease in processing time compared to Long-Lived Tracking predecessor of

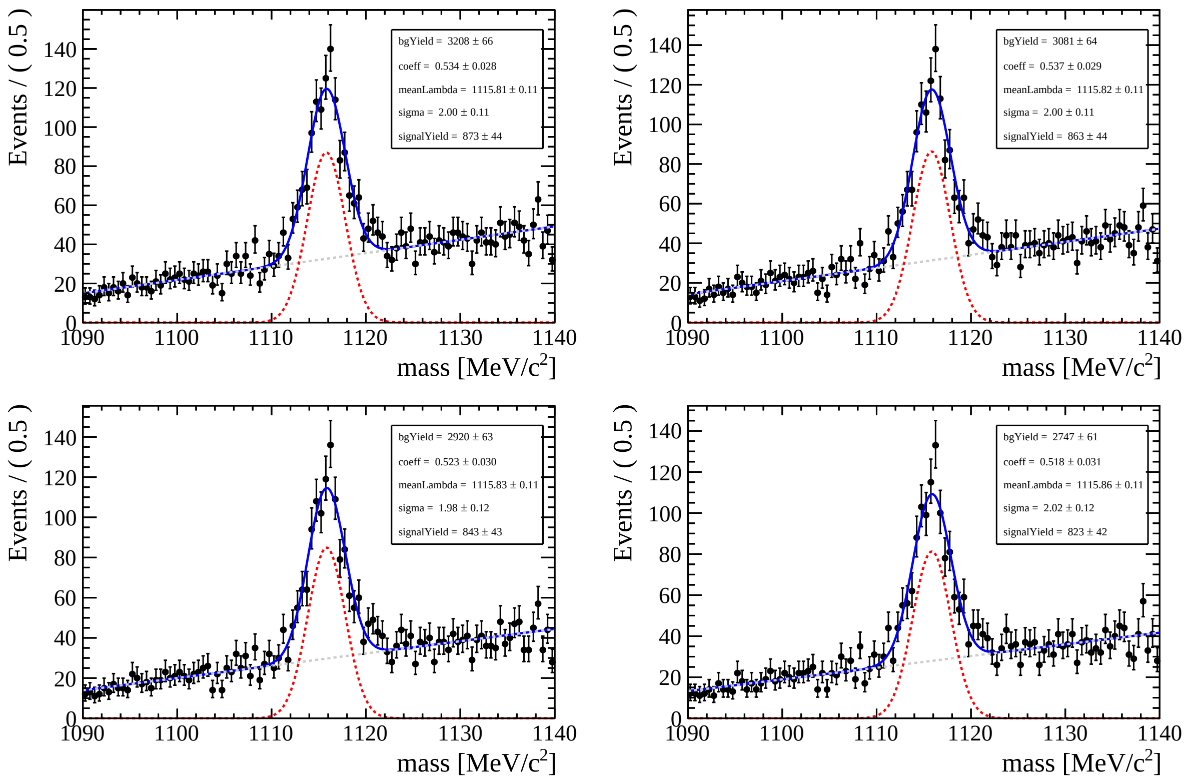


Figure 4.14.15: Invariant mass distribution of reconstructed $\Lambda_0 \rightarrow \pi^+ + \pi^-$ candidates. Each plot corresponds to the different values of the T-Seed classifier threshold. The left upper plot presents $Lambda_0$ invariant mass for a baseline (threshold $=$0.07), upper right corresponds to threshold $=$ 0.1, and the lower plots threshold 0.15 and 0.2 (right and left respectively). The statistical uncertainties are correlated here because the identical sample is used

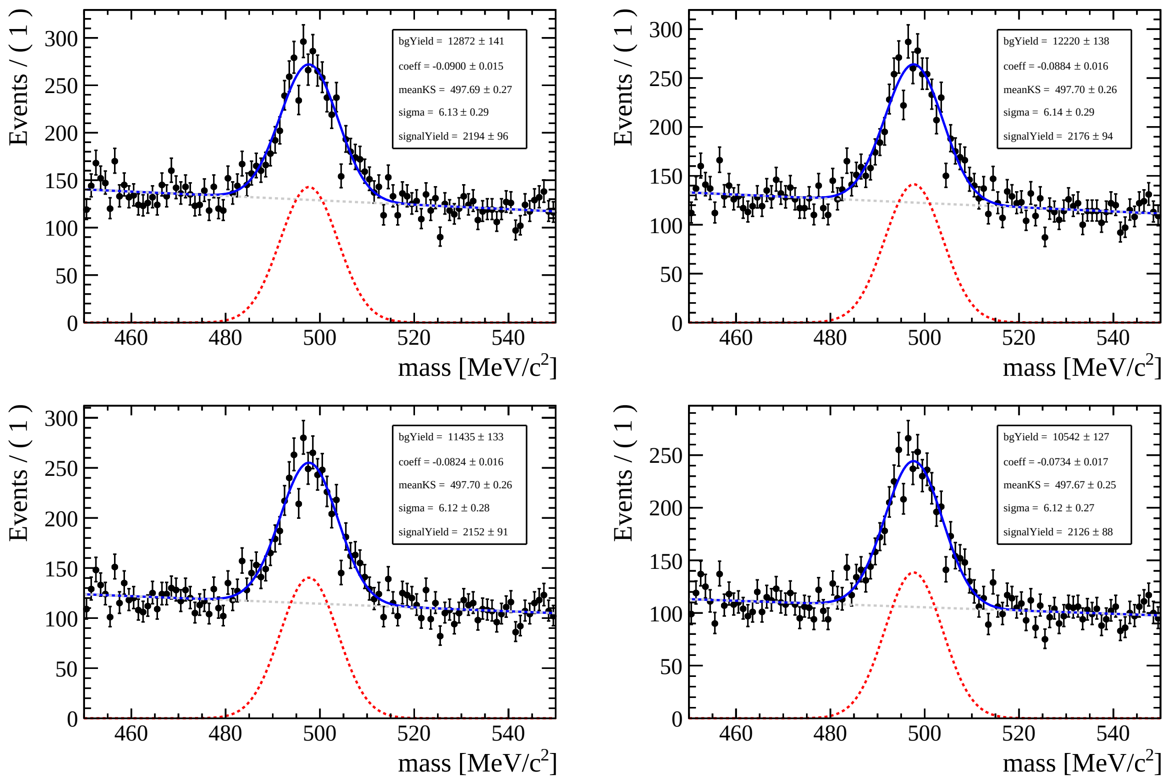


Figure 4.14.16: Invariant mass distribution of reconstructed $K_S \rightarrow \pi^+ + \pi^-$ candidates. Each plot corresponds to the different values of the T-Seed classifier threshold. The left upper plot presents Λ_0 invariant mass for a baseline (threshold =0.07), upper right corresponds to threshold = 0.1, and the lower plots threshold 0.15 and 0.2 (right and left respectively). The statistical uncertainties are correlated here because the identical sample is used.

Table 4.14.8: Evolution of the Long-Lived tracking algorithm K_s and Λ_0 events yield for change of the T-Seed classification threshold.

classification threshold	decay channel	signal	background	S/B [%]	Δ signal [%]	Δ background [
baseline 0.07	$K_S \to \pi^+ + \pi^-$	873	3208	27.21	0	0
0.1		863	3081	28.01	1.1	4
0.15		843	2920	28.87	3.4	9
0.2		823	2747	29.96	5.7	14.4
baseline 0.07	$\Lambda_0 \to \pi^+ + \pi^-$	2194	12872	17.04	0	0
0.1		2176	12220	17.81	0.8	5.1
0.15		2152	11435	18.82	1.9	11.2
0.2		2126	10542	20.17	3.1	18.1

about 45%.

4.15 Future work

This section is dedicated to present ideas, that can be leverages to solve a similar types of problems and due to the LongLived Tracking algorithm's submission deadline were never implemented. One of the project that can benefit from those ideas is Downstream Tracking reconstruction for the Upgraded LHCb.

That section contains a collection of concepts, that author would want to try or implement during the study on Downstream Tracking if he would have a time travel machine.

4.15.1 Recurrent Neural Network

The models that were implemented to classify T-Seeds and select tracks candidates use information that represents entire tracks, such as a track fit quality or total momentum. None of the models that were taken into consideration so far utilizes any additional information about the hits that constitute a given track. The hypothesis is that those hits may provide some hint of whether a particular track is a ghost. Moreover, the hits can be treated as a series and thus processed by the recurrent neural network (RNN). The hits representation produced by the RNN can be combined with the remainder of the track describing features.

A recurrent neural network can be thought of as multiple copies of the same network, each passing a message to a successor, and thus it is a promising model to approach a series problem. Figure 4.15.1 visualizes the idea of how the recurrent neural networks process the input sequential data.

From practical perspective, such a RNN is not be able to learn long range dependencies [26]. Therefore, it is recommended to use more robust implementation called Long-Short Term Memory (LSTM) [82].

One possible enhancement to the recurrent neural network is leveraging the idea of attention mechanism. Attention mechanism is the method that allows to explicitly focus on the part of the data that is relevant to the particular task. The explanation of how the model based on attention mechanism is beyond the scope of this thesis, the detailed explanation of this concept can be found in [139], however,

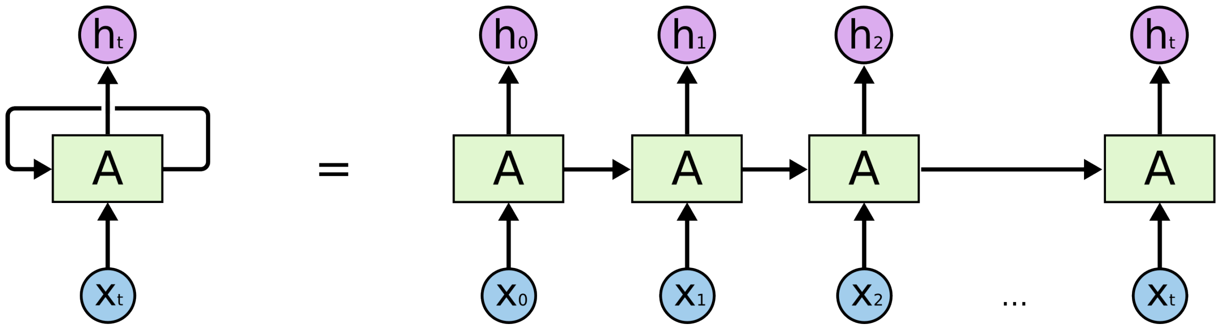


Figure 4.15.1: Visualization of an unrolled recurrent neural network. x_i may represent a i-th hit that constitute a track. Figure taken from [109].

it is important that attention-based models have been used successfully in a variety of tasks Natural Language Processing (NLP) and sequence modeling tasks.

4.15.2 Focal loss

The imbalanced classification problem occurs very often in the field of High Energy Physics. The amount of the background examples significantly exceeds the number of interesting signal events. There are a number of methods on how to approach that kind of problem. Thus this section is dedicated to presenting the one based on a modification of a loss function. [10] The proposed loss function is called **focal loss** and given by [97]:

$$FL(p_t) = -(1 - p_t)^{\gamma} log(p_t) \tag{4.27}$$

where:

$$p_t = \begin{cases} p & \text{when } y \text{ is true} \\ 1 - p & \text{otherwise} \end{cases} \tag{4.28}$$

and y specify the grand-true class and p is a model's predicted probability for class with label $y = 1$, and γ is a hyper-parameter that define how much the well classified examples are diminished. Figure 4.15.2 presents the plot of the Focal loss versus probability of a true example being classified as a true one.

The idea of Focal loss was proposed to improve one stage image detection model. Image detection is a type of problem in which the model needs to find a location (bounding box) of particular objects, for instance, find a face on a camera footage. In that kind of problem, the majority of regions are empty, and only a tiny number of them contain relevant objects. Thus this setup is very similar to the searching for an interesting physical signals. The Focal loss adds a factor $(1 - p_t)^{\gamma}$ to the standard cross-entropy criterion. Setting $\gamma > 0$ reduces the relative loss for well-classified examples ($p_t > 0.5$), putting more focus on hard, misclassified examples.

[10] The other method that is usually applied is dataset re-sampling. Two main approaches to randomly re-sampling an imbalanced dataset can be exclusively applied. One of them, called undersampling, drops examples from the majority class, and the second one, called oversampling, duplicates examples from the minority class.

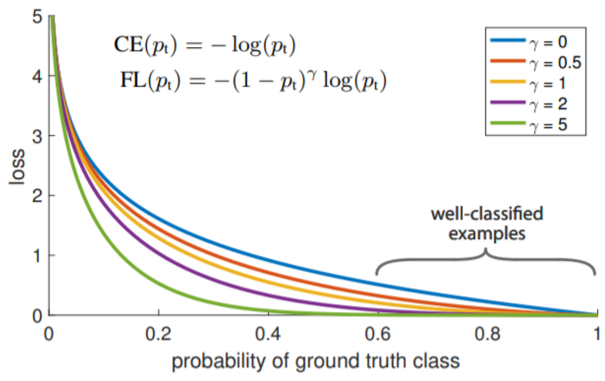


Figure 4.15.2: Visualization of the Focal loss (FL) versus the probability of true class, obtained for a different value of hyper-parameter γ. $\gamma = 0$ corresponds to the cross-entropy. Figure taken from [97].

In order to use Focal Loss as an xgboost objective function, one needs to implement its first and second derivative. Which can be done manually or using one of the automatic derivation packages like Scipy [7].

4.15.3 Workflow Management System

One of the parts of the study that can be enhanced is a pattern recognition validation and performance check. The current validation process requires a lot of manual interventions. The trained model needs to be extracted, then deployed within a Brunel application, after that Brunel output (DST file) is used as an input to the DaVinci which produces Ntuple. Those Ntuples can be used by the custom root scripts to reconstruct the invariant mass of the particle decaying into long-lived daughters to measure its width. Each of the listed above steps has to be executed painstakingly, which requires a lot of time and it is prone to errors. This processing chain can be automated using software called Workflow Management System.

That kind of tool organizes tasks into a Directed Acyclic Graph (DAG) [11], see an exemplary graph that visualizes possible automation of the pattern recognition algorithm performance validation in Figure 4.15.3. The scheduler that is dedicated to executing tasks organized into the DAG runs those tasks on an array of workers while following the specified dependencies. One of those programs, that can be easily used within the LHCb environment is an open-sourced Apache Airflow [4].

[11] DAG is a graph (collection of vertices and connecting them edges) that is directed and without cycles. This means that it is impossible to start from a given vertex v_{start} and follow any path (consistently directed sequence of edges) ends in the starting node v_{start}.

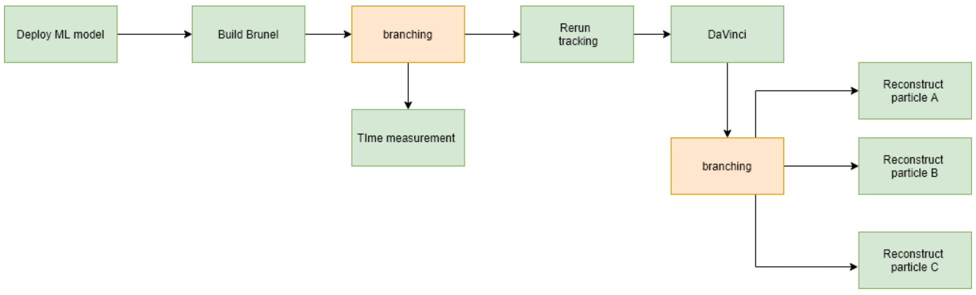


Figure 4.15.3: An example DAG that can be used to automate model performance checks.

5
Emulation and Monitoring of the Upstream Tracker RAW data

This chapter is dedicated to present the emulation and monitoring package for Upstream Tracker called TbUT. To provide a full picture of the data emulation process, this chapter starts by introducing the physics principles of signal formation. Within this section, the interaction of particles with matter and review of the principles of particle detection using silicon devices will be also shortly discussed. The TbUT package was designed to emulate all RAW Salt data processing algorithms, as well as to generate a number of performance and monitoring plots. This software was initially implemented in order to process all RAW data collected during the UT testbeams experiments. The described software will also play a crucial role in the future monitoring and calibration of the UT detector. The output of this software will help to ensure the high quality of the collected data.

5.1 Interaction of Particles with Matter

When a charged particle traverse through matter, it interacts with the material what results in its energy lose and multiple scattering. The energy deposition is caused mostly by inelastic collisions with atomic electrons and elastic scattering from the nuclei of the absorbing material along the particle trajectory. For particles which masses are well above the mass of an electron the mean loss of energy is described by the Bethe-Bloch-Formula:

$$-\left\langle \frac{dE}{dx} \right\rangle = Kz^2 \frac{1}{\beta^2} \frac{Z}{A} \left[\frac{1}{2} \ln \frac{2m_e c^2 \beta^2 \gamma^2 T_{max}}{I^2} - \beta^2 - \frac{\delta(\beta\gamma)}{2} \right] \tag{5.1}$$

where:

- $\frac{dE}{dx}$ is energy loss of the particle usually given in $\frac{eV}{g/cm^2}$
- K is a constant: $0.307\ MeV\, mol^{-1}\ cm^2$;
- Z and A are the atomic number and mass respectively (for silicon 14 and 28 respectively);
- c is the speed of light in vacuum;
- $\gamma = \frac{1}{\sqrt{1-\beta^2}}$ is the Lorenz Factor and $\beta = \frac{v}{c}$;
- I is a medium average ionisation energy;
- $\delta(\beta\gamma)$ describes the density effect correction for high energy particles;
- T_{max} is the maximum energy transferred to a free electron in a single collision, given by:

$$T_{max} = \frac{2m_e c^2 \beta^2 \gamma^2}{1 + 2\gamma \frac{m_e}{M} + (\frac{me}{M})^2} \tag{5.2}$$

The initial version of Formula 5.1 was proposed by Bethe [29] only three years after Schrodinger postulated non-relativistic Quantum Mechanic. The quantity $-\left\langle \frac{dE}{dx} \right\rangle$ can be interpreted as stopping power of a particular material. One should note, at this point, that this dependency is represented by the ratio of the atomic and mass numbers, and since the ratio is approximately constant for all elements, one can conclude that, in fact, the energy loss weakly depends on the material type. Figure 5.1.1 presents the stopping power for positive muons in copper as a function of $\beta\gamma$. It is worth mentioning that expressing the stopping power as a function of $\beta\gamma$ makes the energy loss curve approximately universal for any type of interacting charged particles.

For low energies term $\frac{1}{\beta^2}$ in Equation 5.1 is dominant and the stopping power decreases with increasing energy, reaching minimum when $(\beta\gamma) \approx 3$. A particle with an energy loss at the minimum is called a Minimum Ionizing Particle (MIP).

It is worth noticing that for a charged particle traversing a material, the number of collisions occurring and the amount of energy transferred in each collision is subject to statistical fluctuations. Formula 5.1 provides only the average value of energy loss, which due to the statistical nature of the process, may vary from the actual value. For a relatively thick material, the number of collisions will be large enough to justify modeling distribution of the energy loss by the Gaussian distribution [68]. Although for a thinner materials, which are widely used in the current tracking detectors, the average energy loss is small, so the large fluctuation in the deposited energy can be observed. This results in a largely asymmetric charge distribution, which can be parametrized by the long-tailed Landau distribution shown in Figure 5.1.2. This distribution is traditionally modelled by a Landau convoluted with a Gaussian to extract the value of generated charge (electron-hole paris), called the Most Probable Value (MPV). Since we usually operate the silicon devices in the proportional mode, we assume that the generated charge is

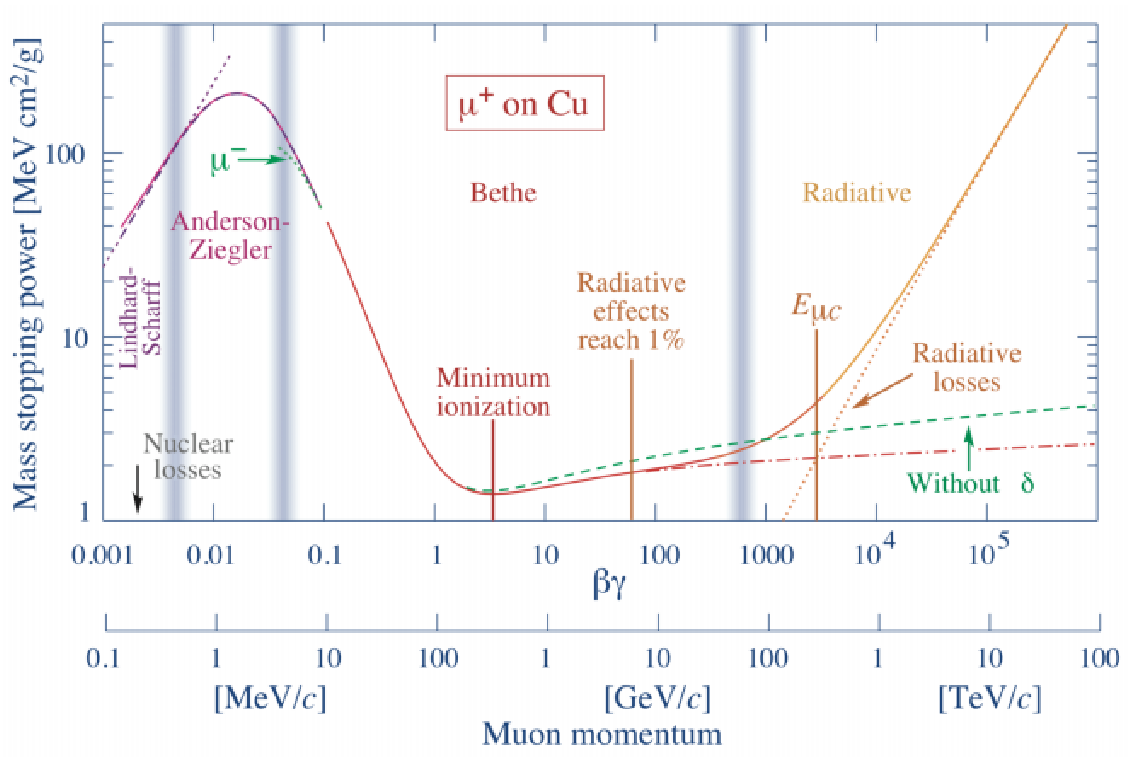


Figure 5.1.1: The stopping power $-\left\langle \frac{dE}{dx} \right\rangle$ as a function of $\beta\gamma$. The region that the Bethe-Bloch formula describes is $0.1 \leq (\beta\gamma) \leq 1000$ and is highlighted in red. Figure taken from [135].

proportional [1] to the energy deposited by a charged particle within its active volume. The Landau-Gauss convolution is given by:

$$f(x) = \frac{1}{\sqrt{2\pi}} \frac{\tau}{\sigma_{noise}\sigma_{width}} \int_{x_0+5\sigma}^{x_0-5\sigma} Landau(x, \tau, \sigma_{width}) \times Gauss(x, x_0, \sigma_{noise}) dx \tag{5.3}$$

where :

$$Landau(x, \tau) = \frac{1}{\pi\tau} \int_0^\infty e^{-t\ln t - \frac{t \cdot x}{\tau}} \sin(\pi t) dt \tag{5.4}$$

and τ is a Landau Most Provable Value (MPV), σ_{width} is a width of a Landau distribution and σ_{noise} is a Gaussian standard deviation. Landau Gauss convolution is calculated using numerical integration.

5.2 Operational principles of silicon detectors

As described in Chapter 1, several technologies are used for full event reconstruction at LHCb. However, to achieve the highest precision and resolution, silicon detectors are currently the best option. That kind of detector uses a minimal amount of material which particles need to traverse to be detected.

[1] In recent years the various studies on the new generation of radiation hard silicon devices led to the fabrication of sensors (LGAD and iLGAD) that exploit charge multiplication using elaborate structures.

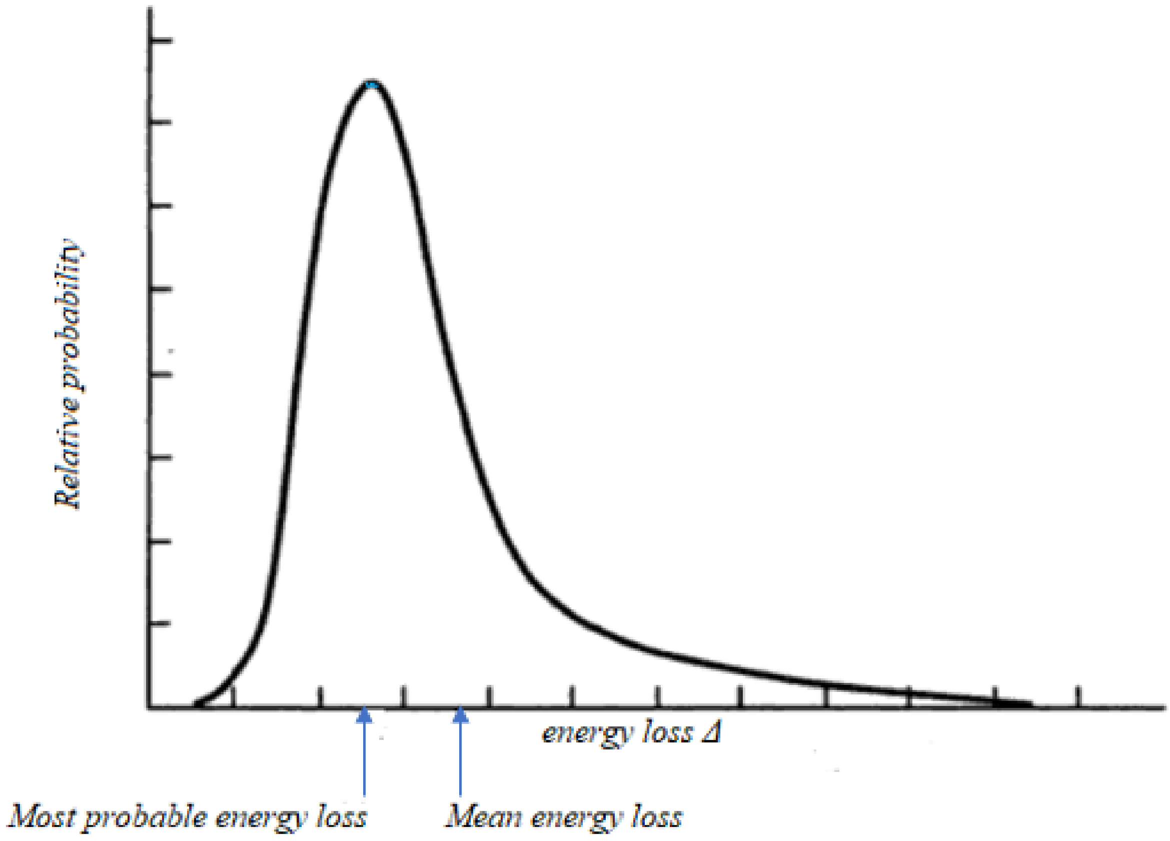


Figure 5.1.2: A Landau distribution. This distribution has a long tail which decreases non exponentially for large values of x. It is defined over the all set of positive real numbers and its mean and variance are undefined.

The only limitation of using this technology is its cost. Therefore, they are placed where the highest resolution is required, for instance, around the LHCb detector's interaction point. Most of the semiconducting detectors are made of silicon. This is because silicon is the second most abundant element on Earth and is widely used in industrial applications. Silicon has an atomic number of 14, so it has 14 electrons in three shells (2,8,4 arrangement). The electrons arranged in the outer shell are the valence electrons. Due to this electron arrangement, within the crystalline lattice structure, each silicon atom is surrounded by four neighbours, see Figure 5.2.1.

The periodic potential of the crystalline structure results in the formation of electron energy bands. The energy bounds can be interpreted as a collection of the individual energy levels of electrons surrounding each atom. The wavefunctions of the individual electrons are overlapped with those of electrons enclosed to neighbouring atoms, and due to the Pauli exclusion principle, the electron energy levels are prohibited from being the same so that one obtains a set of closely spaced energy levels, forming an energy band.

At a temperature of absolute zero, the highest energy electrons lie in the valence band. Those electrons are strongly bonded to the atoms and thus are not able to carry the charge. However, at the higher temperatures, electrons also occupy higher energy states known as the conduction band. The energy between the valence and conduction bands is called the band gap energy E_g. The value of the band

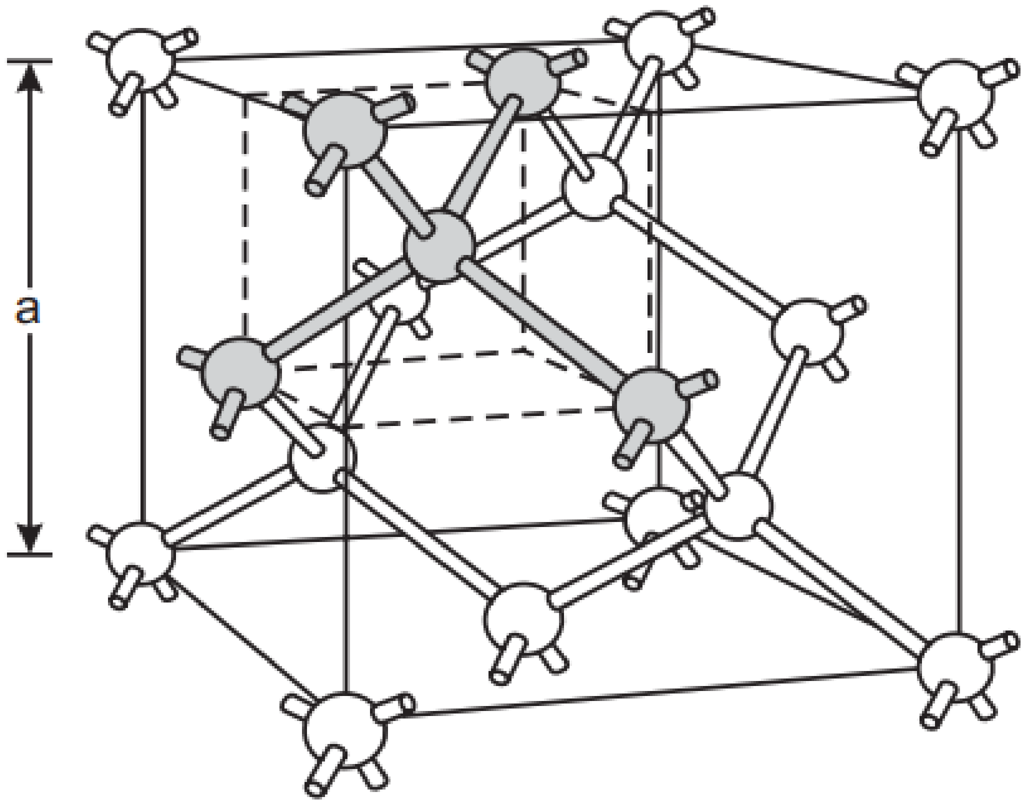


Figure 5.2.1: Lattice structure of silicon. The building block of the lattice is formed by a central atom bonded to four equidistant neighbors (indicated in gray). Figure taken from [130]

gap allows classifying materials into conductors, semiconductors, and insulators, which is shown in Figure 5.2.2.

One of the most important features of semiconductors is their ability to change electrical conductivity by introducing impurity atoms, called dopants. These dopants replace atoms of the original material at lattice sites. They typically have more (*n*-type donors) or fewer (*p*-type donors) valence electrons in their outer shell, depending on the specific type of dopant.

Most semiconductor detectors are made of $p - n$ junction, which manifests a diode characteristic. Such a silicon device is created by connecting two components of opposite dopping in good thermodynamical contact. The most important property of such a device is electrons' phenomena diffusing into the *p* region and holes into the *n* region where they recombined. Consequently, an electric field gradient is created, which eventually stops the diffusion process and establishes thermal equilibrium. Due to the electric field, there is a potential difference across the junction. The changing potential region is known as the depletion zone since it is almost fully depleted of all mobile charge carriers. This characteristic can be utilized for radiation detectors. Electron-hole pairs that are generated in the depletion zone by ionizing radiation will be separated by the electric field and can be detected if electrical contacts are

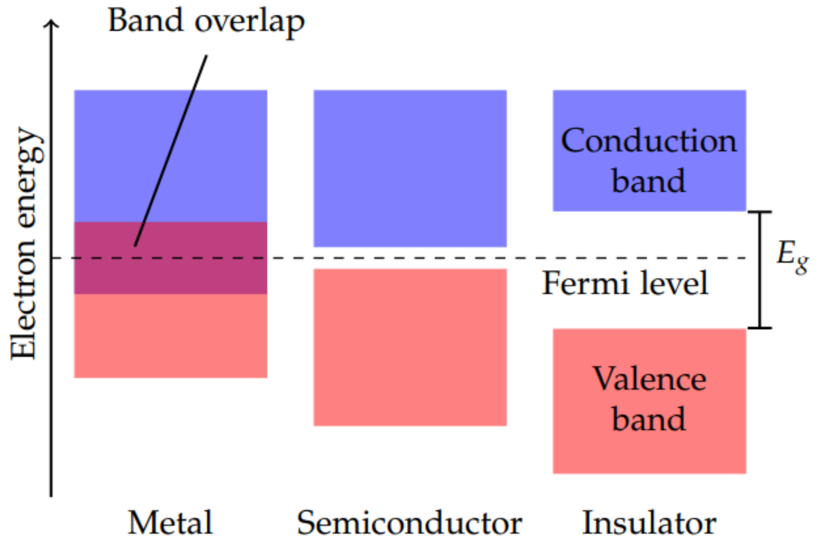


Figure 5.2.2: Classification of metals, semiconductors and insulators according to their band gap energy E_g. The Fermi level, denoted by the dashed line, is a hypothetical energy level which would have a 50% probability of being occupied at thermal equilibrium. Figure adapted from [77]

connected to the device, which is shown in Figure 5.2.3.

5.3 TbUT Emulation Software

The TbUT package is a software developed entirely by the author, and similarly to all of the official LHCb software, it is based on the Gaudi framework [23]. Gaudi identifies components with specific purposes and well-defined interfaces, interacting with each other to provide the complete functionality of an application. Gaudi provides to the application general-purpose components, like messaging and configuration services, see Figure 5.3.2. The configuration is performed via the customized python scripts. By design, the Gaudi framework decouples objects describing data from those describing algorithms.

The software was designed based on the Object-oriented Programming paradigm. This paradigm states that the software design is focused on data (also called **object**) rather than functions. An object can be defined as a data field that has unique attributes and behavior [124].

Each of the TbUT components inherits from one of the following classes:

- **Algorithms**, these objects inherit from *Gaudi::algorithm*, which is an essential part of the Gaudi application. They can read the input data, and via an appropriate tool process them, as well as they are capable of storing a data for the next processing steps within the TES.

- **Monitoring algorithm**, these object, similarly to the **Algorithms**, inherit from *Gaudi::algorithm*. Their job is to extract data from the TES and generate a set of monitoring histograms, which suppose to visualize the performance of the sensor. This component is a key to make a proper calibration.

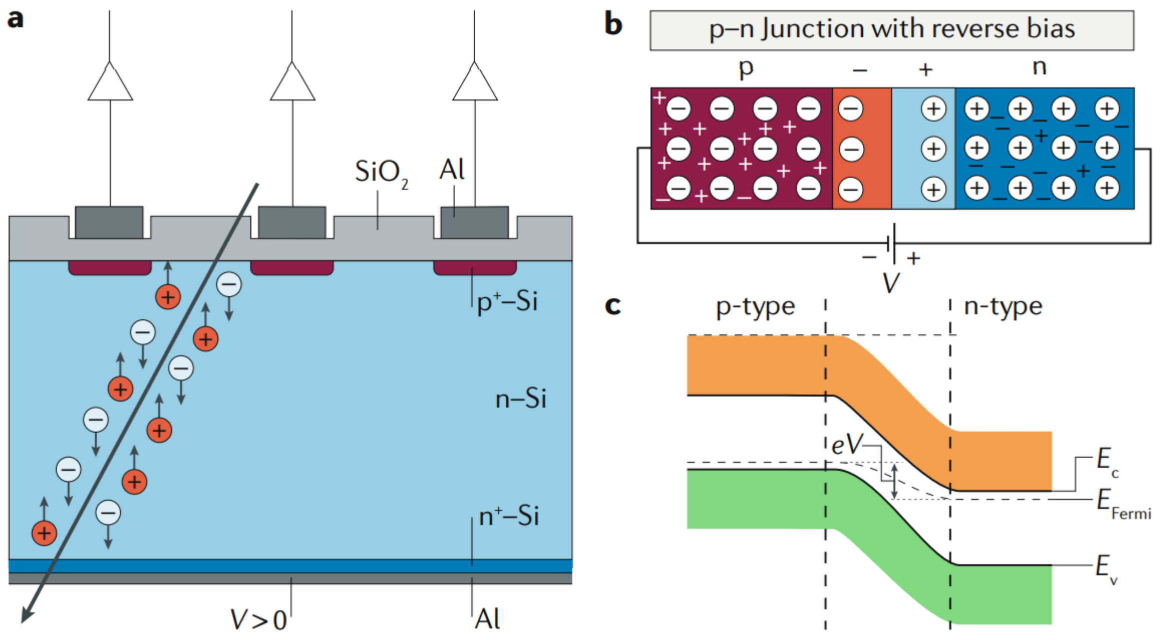


Figure 5.2.3: Visualization of principles of operating of silicon strip detector. The charged particle passing though a depleted detector creates electron-hole pairs that are separated in the electric field inducing a signal that is recorded by the dedicated electronics (panel a). Panels (b) and (c) present the development of the space charge region at the boundary between p-type and n-type silicon, which is enhanced by the application of a reverse-bias electric potential. Figure taken from [16]

- **Tools** These objects inherit from the pure virtual interface *IProcessingEngine*, and they were designed to perform the processing actions, for instance, pedestal removal.
- **Serializable data** this objects inherit from the *GaudiKernel::DataObject* and they are the created by the **Tools**. They can be stored inside the Transient Event Store (TES).
- **Tool's Factories** these types of objects were implemented as a concrete instances of a Factory Design Pattern. This concept was introduced in the Design Pattern book [67].

The first algorithm within the TbUT is dedicated to decoding the raw data acquired by the acquisition board, which is visualized in Figure 5.3.1. During this project's lifetime, the software has to able to process data acquired by the number of different DAQ systems, each of these producing the data in a completely different format. To achieve that kind of flexibility the tool, that is responsible for reading the input data was implemented using a Factory Design Pattern [67].

5.3.1 Pedestal Subtraction

Pedestals are the charge values measured by the read-out system in the absence of signal and noise. Each of the read-out channels can have a different ADC value of pedestal, and each pedestal's value depends on such environmental conditions as temperature, humidity, and operating conditions, like applied bias

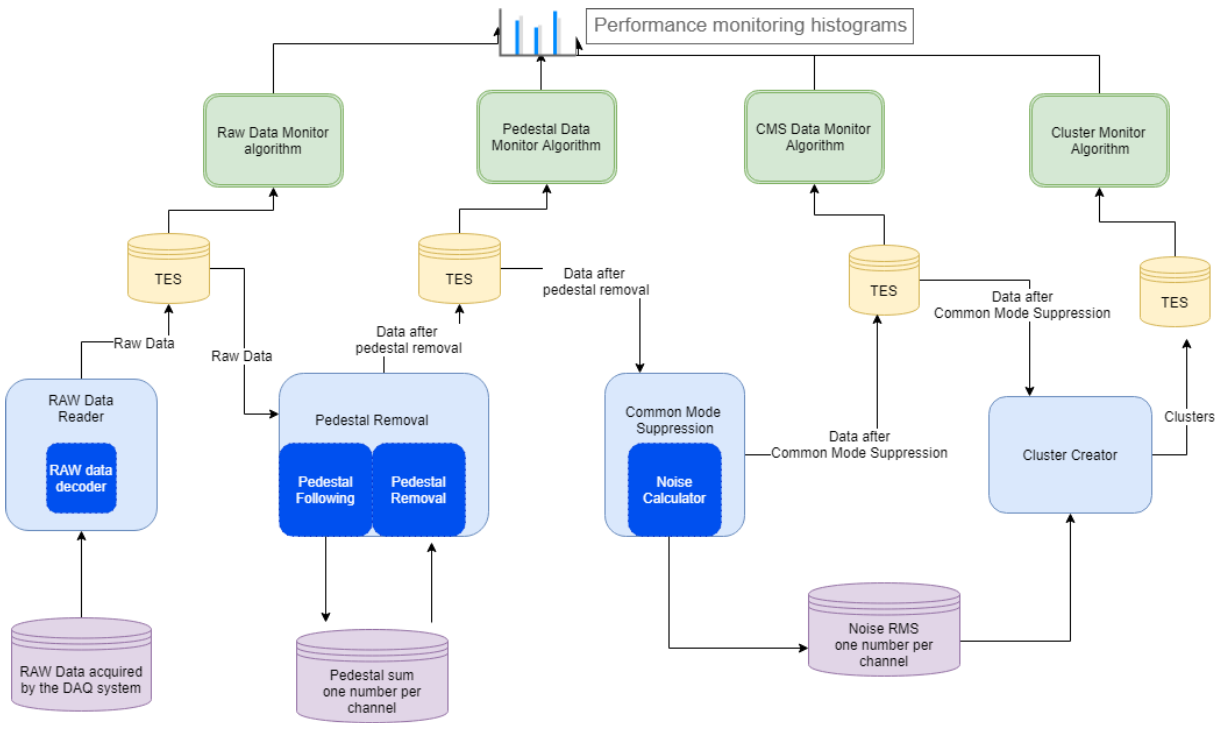

Performance monitoring histograms
Raw Data Monitor algorithm
Pedestal Data Monitor Algorithm
CMS Data Monitor Algorithm
Cluster Monitor Algorithm
TES
TES
TES
TES
Raw Data
Raw Data
Data after pedestal removal
Data after pedestal removal
Data after Common Mode Suppression
Data after Common Mode Suppression
Clusters
RAW Data Reader
RAW data decoder
Pedestal Removal
Pedestal Following
Pedestal Removal
Common Mode Suppression
Noise Calculator
Cluster Creator
RAW Data acquired by the DAQ system
Pedestal sum one number per channel
Noise RMS one number per channel

Figure 5.3.1: The diagram presents the main components of the TbUT package and interactions between them. The light blue components are algorithms, dark blue are tools, and the violet ones represent serializable data. The direction of the arrows indicates whether the data is an input or output to the algorithm.

voltage. To ensure proper pedestals are removed from the raw ADC data, prior to each data collection run, the non-signal measurements (pedestal runs) are taken. Usually, pedestal data are quick to collect since no trigger is required.

The pedestal subtraction algorithm has two phases. In first, the pedestal values are calculated. During the second one, the determined pedestal values are subtracted from the raw ADC data. Figure 5.3.3 presents Pedestal Subtraction algorithm sequence diagram. That kind of UML diagram [120] is dedicated to depicts an interaction between objects in sequential order. These diagrams are widely used by the enterprise and software developers to document and understand requirements for new and existing systems.

5.3.1.1 Pedestal following

From the mathematical point of view the calculation of the pedestal is a running average. In every training event then pedestal sum is updated. This update takes into account the previously calculated value of the pedestal sum and the current ADC count. The pedestal sum is calculated for each channel separately. To be more precisely, the pedestal sum, $p_i^{sum}(n)$, for channel i and event n can be expressed as follows:

$$P_i^{sum}(n+1) = P_i^{sum}(n) + \frac{\Delta_i(n+1)}{N} \tag{5.5}$$

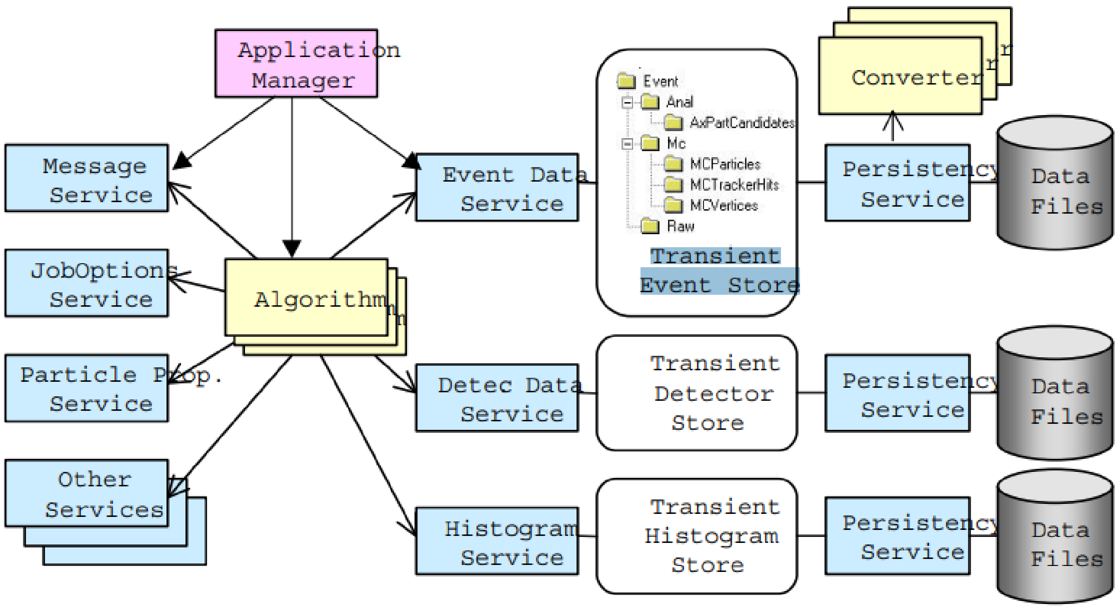


Figure 5.3.2: Object Diagram of the Gaudi architecture, figure taken from [23]

Where the $\Delta_i(n+1)$ is a event to pedestal correction. This correction is expressed as:

$$\Delta_i(n+1) = ADC_i(n+1) - P_i^{sum}(n) \tag{5.6}$$

In Equation 5.5, N is the weighting factor set by default to 1024.

To increase the suitability of the pedestal, and to remove potential outliers that may bias the pedestal value, the limit for correction is applied. Therefore the correction is given by:

$$\Delta_i(n+1) = \begin{cases} \Delta_i(n+1) & |\Delta_i(n+1)| < 15 \\ 15 \cdot sign\left(\Delta_i(n+1)\right) & \text{otherwise} \end{cases} \tag{5.7}$$

where $sign(x)$ is a function that extract the sign of a real number.

To determinate the pedestal values the pedestal sum should be normalized, so:

$$p_i = \frac{P_i^{sum}}{N} \tag{5.8}$$

The initial value of the pedestal sums is a mean value calculated for the first 100 pedestal events.

5.3.1.2 Pedestal subtraction algorithm

The second phase of the pedestal subtraction algorithm is removal of determined pedestal values from the raw ADC data. This procedure can be expressed as follows:

$$ADC_i = ADC_i^{RAW} - p_i \tag{5.9}$$

where: ADC_i is signal value after pedestal subtraction for event i, ADC_i^{RAW} is a raw data and the p_i is the pedestal value. Each of this quantities are in the unit of ADC counts. Figure 5.3.4 presents two

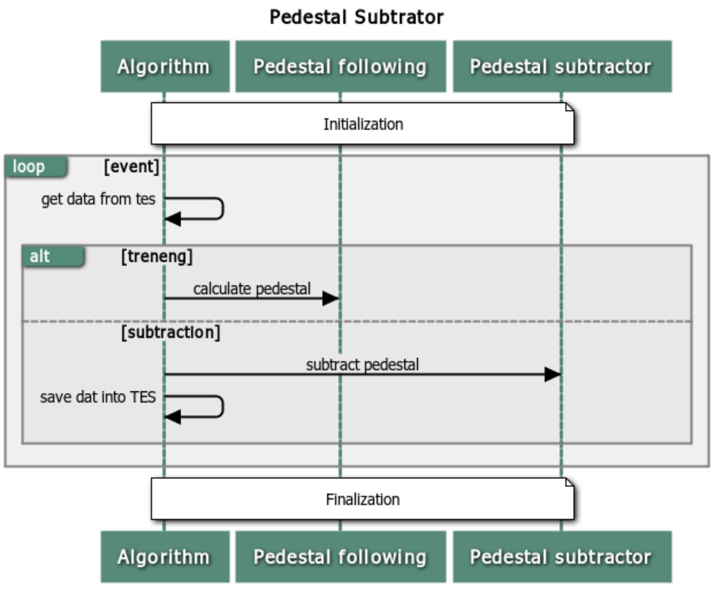


Figure 5.3.3: Pedestal Subtraction sequence diagram.

performance monitoring plots, the first one shows the raw data collected by the DAQ system, which is an input to the TbUT, and the second one is the data after pedestal removal algorithm.

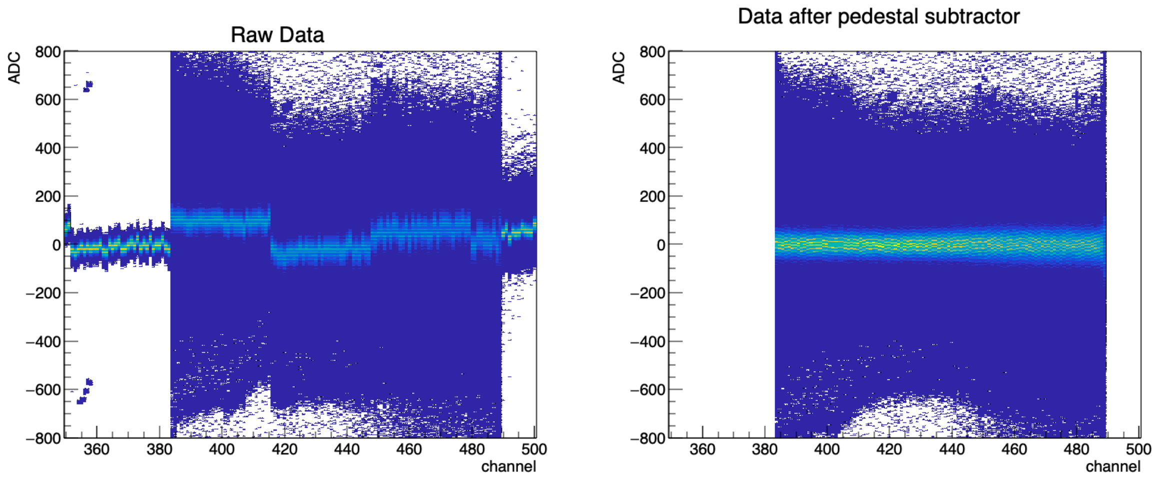


Figure 5.3.4: Exemplary Pedestal Subtraction Algorithm's monitoring plots. Typical raw data ADC values (left), Pedestal subtracted ADC values (right). The channels below 380 and above 480 were masked since they are not connected to any front-end chip.

5.3.2 Common mode subtraction

The next step within the TbUT processing chain is Common Mode Subtraction and the noise calculation. The noise, one floating-point number per channel, is then used to evaluate zero-suppression and clustering threshold value for the clusterization algorithm described in the next section.

The total noise affecting the signal measurement consists of two main components. The first one affects every read-out channel independently. The second one is common to the group of consecutive channels. This type of noise is called Common Mode, and it may originate from grand from loops in the power supplies, or read-out strips might pick some environmental noise.

The Common Mode Suppression algorithm consists of two phases. The first one was implemented to calculate the average pedestal-subtracted ADC value of the channels in 32-channel groups. This value is calculated for each event independently. Using this mean value, a search for particle hits is performed for each channel. All channels with hits are masked, and a new mean value is calculated for each 32-channel group. The channel is masked when its ADC value fulfils the following condition:

$$|ADC_i(n) - CMS_i^{corr}| > \alpha\sigma_i^{RMS} \tag{5.10}$$

where $ADC_i(n)$ is a pedestral-subtracted charge collected in channel n in event i, σ_i^{RMS} is a channel ADC Root Mean Square, CMS_i^{corr} is a correction, and α is a tunable parameter, that specifies the required distance between noise and signal. After the initial testbeam studies, this parameter was set to $\alpha = 4$. The previously calculated mean value is then used to correct the ADC values in all channels of the 32-channel group.

The second phase of the CMS algorithm is calculation of the noise per channel n, according to the following formula:

$$\sigma_i = \sqrt{\frac{\sum_{n=1}^{N}(ADC_i(n) - \mu_i)^2}{N}} \tag{5.11}$$

where $\mu_i = \sum_{n=1}^{N} ADC_i(n)$ is a mean ADC value per channel.

Figure 5.3.5 presents the one-dimensional projection of the data after pedestal and Common Mode subtraction. It is clearly visible, based on the noise values, that the Common Mode removal step is necessary to reduce the noise level and thus increase the Signal-to-Noise (S/N) ratio.

A few typical events after all corrections applied are shown in Figure 5.3.7. Here, no requirement is made on the number of tracks. Strips with large ADC counts are indicative of the passage of a beam particle through the detector. The other channels show roughly Gaussian fluctuations about zero, typical of incoherent detector noise. Signals generally stand out significantly above the noise. These examples were selected to visualize a couple of cases that may occur, such as double strip clusters (event 14) and multiple clusters per event (event 83).

5.3.3 Cluster finding algorithm

The final algorithm is executed to reconstruct clusters (or hits) using previously proceeded ADC values and noise per channel, which plays the clusterization threshold. The additional algorithm's input parameters are the value of the low and high thresholds. Clusters in the DUT are built up by searching for

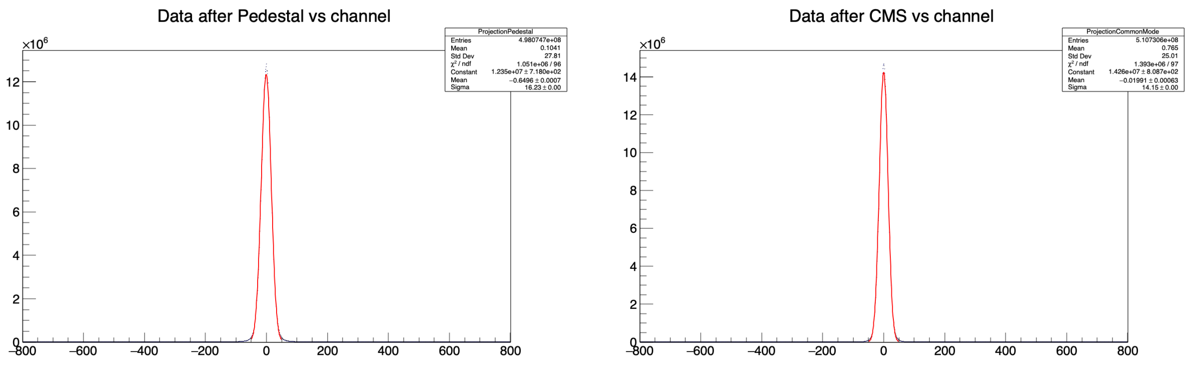


Figure 5.3.5: Projection of data after pedestal subtraction (left) and CMS (right)

a seed strip with a collected charge exceeding a high threshold value, that was evaluated as a multiply [2] of the noise measured on this channel. The outcome of this subroutine is a list of cluster seeds. The next subroutine is dedicated to removing the seeds that belong to the same clusters. Such a situation occurs when the cluster has more than one strip, and all of them have collected charges value higher than the high threshold. Moving away from the seed strip, the adjacent side strips with the ADC count exceeding the low threshold are added to the cluster seed. The cluster reconstruction is terminated when a side strip charge is below a low threshold value. The maximum number of strips that can constitute a cluster is limited to 5. When there are multiple strips in the cluster, the position is computed using linear charge weighting:

$$x_{cluster} = \frac{\sum_{i=1}^{S} x_i q_i}{\sum_{i=1} S q_i} \tag{5.12}$$

where x_i and q_i are the positions and charges of the strips in the cluster, and S is the number of strips that constitute the cluster. Thus, the cluster position is represented by a floating point number.

```
Data: ADC data after Common Mode Suppression (1...$N$ events),
      clusterization thresholds (one floating-point number per channel),
      two integer values to obtain low and high threshold values.

Result: vector of reconstructed clusters

for event in 1...$N$ do :
   find cluster seeds;
   remove seeds belonging to the same clusters;
   extend cluster seeds by adding adjacent strips;
   calculate cluster position
end
```

Algorithm 5.1: Cluster Creator algorithm

The analysis using the data reconstructed by this clusterization algorithm can be used to investigate the quality of the collected data by the UT detector. The first example of a very important plot is a his-

[2]The multiplication factor was a tunable paramter, that was determined using a dedicated calibration procedure. Two strategies were employed for this purpose. The first one relied on a signal-to-noise cut and the second one was based on the observed noise hit rate. Both of them yielded a similar results.

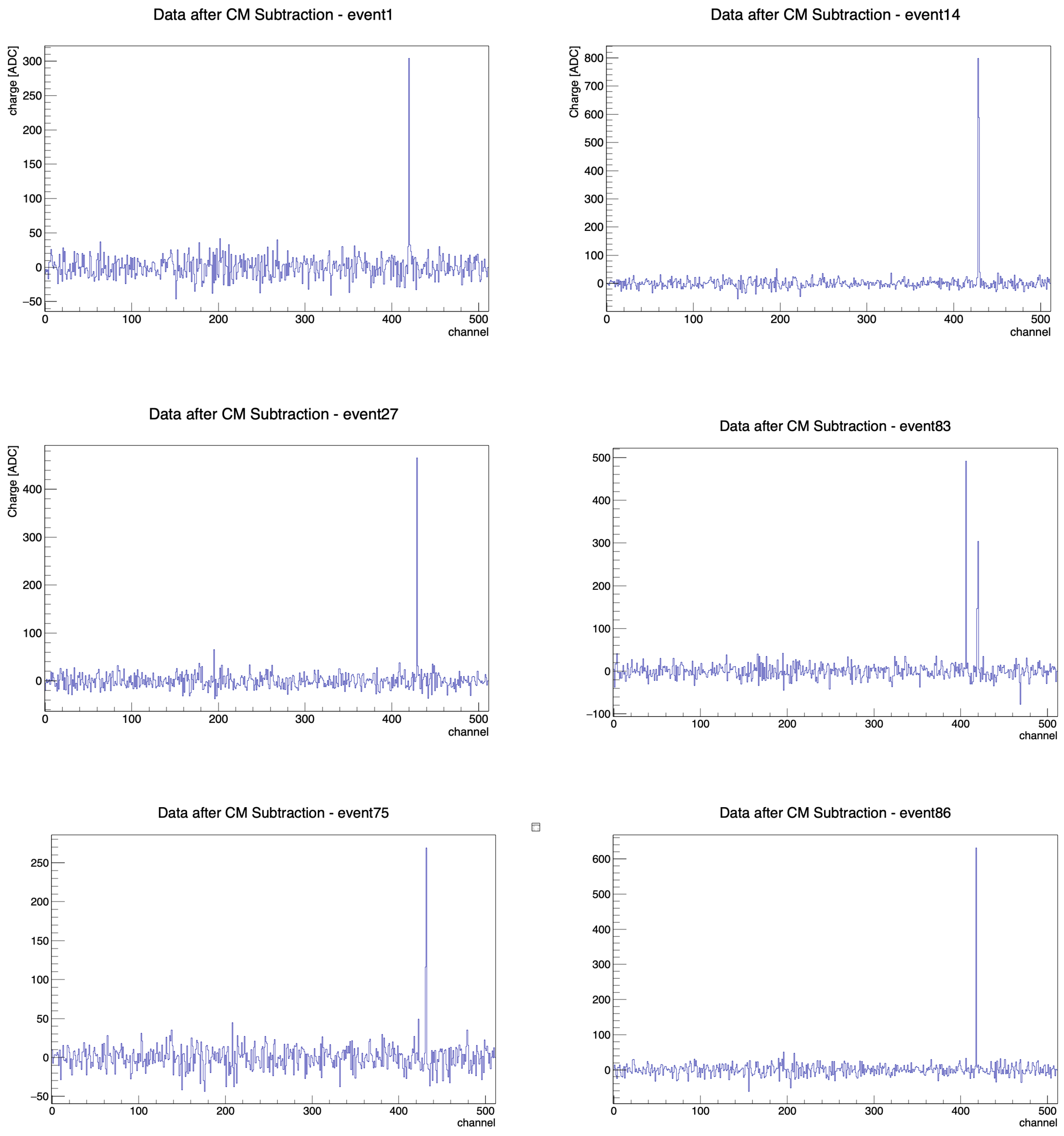


Figure 5.3.6: A few examples of a single events after pedestal and common mode removal.

togram presenting the distribution of the cluster charge. Based on it one can get physics quantities describing the sensor's performance, the cluster charge data by fitted with the Landau Gauss convolution model. The parameter of this fit allows determining the Most Provable Value and the width of the signal. Figure 5.3.8 presents a collection of cluster charge distribution histograms. Those histograms were created on top of the data collected during the one of the testbeam. They were used to show the sensor performance versus the bias voltage applied to them. Those studies were conducted to investigate the effect of a drop in the collected charge when sensor irradiation increases, which was seen previously [19]. The collected charge drop may lead to decreasing the signal-to-noise ratio.

Another example of monitoring plot is the cluster size spectrum. That plot allows better understanding of how the resolution of the sensors changes as a function of the angle between the incident particle

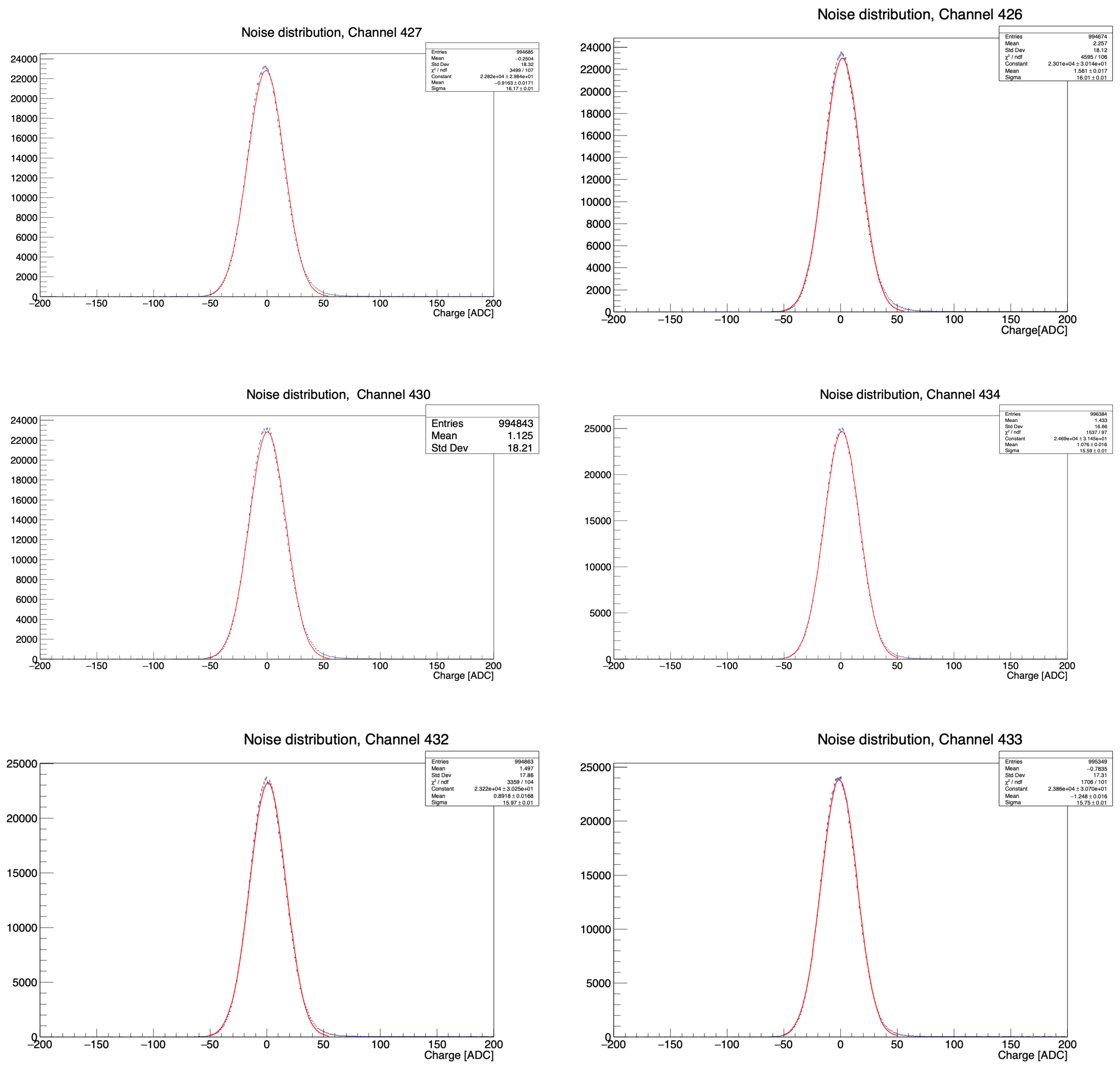


Figure 5.3.7: Noise distributions after Common Mode Suppression for selected channels within the beam region. The red curves show a Gaussian fit to the core of the distribution.

and the normal to the sensor. It can also be used to infer charge sharing properties of the sensor, which in turn, has significant impact to improving the spatial resolution of a single hit. Figure 5.3.9 presents a cluster size as a function of the sensor rotation angle obtained using data collected during the 2015 testbeam.

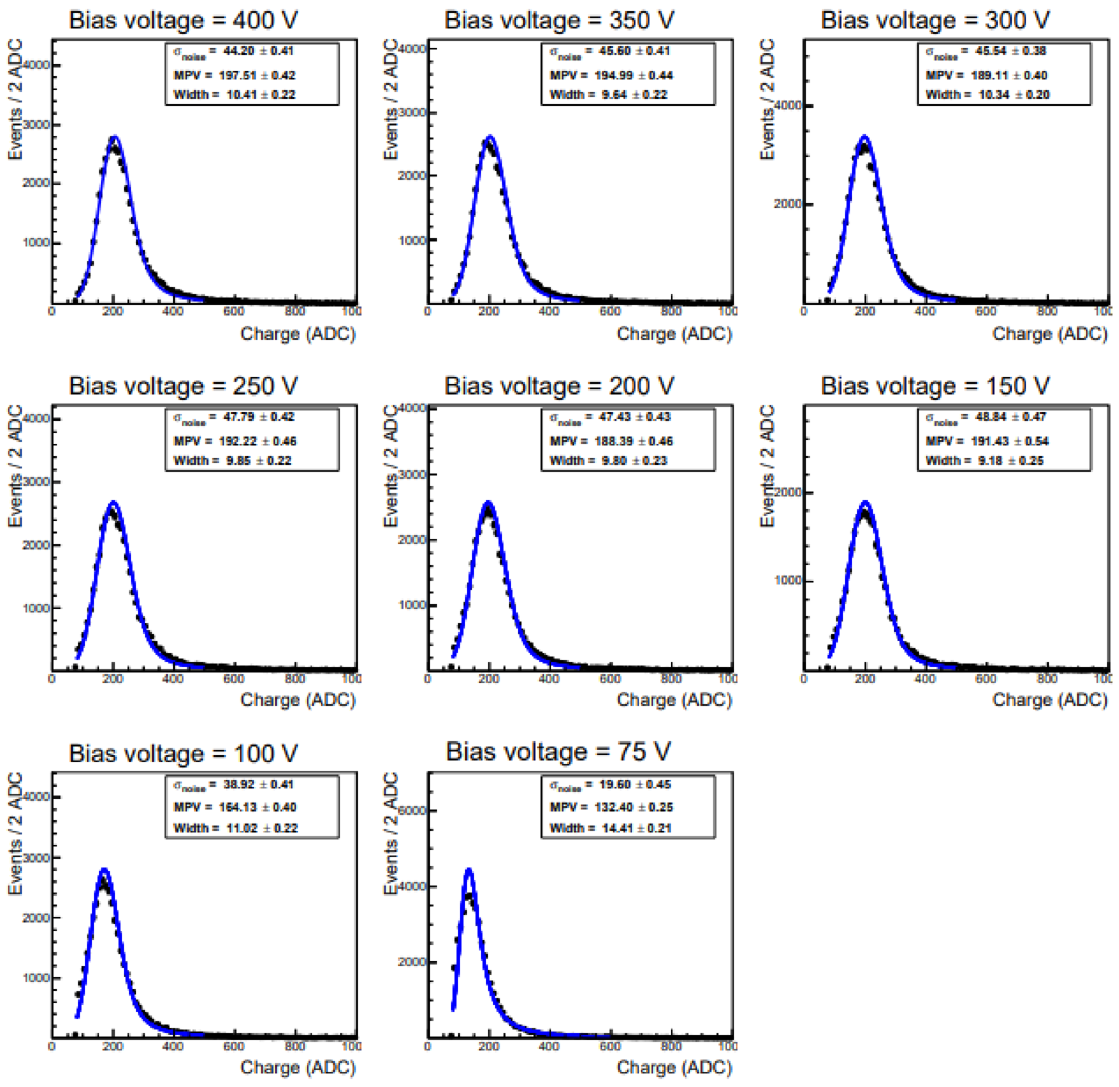


Figure 5.3.8: Cluster charge distributions. The data are fit to a Landau convoluted with a Gaussian resolution function, and the fit is shown (solid blue line). Figures taken from [11].

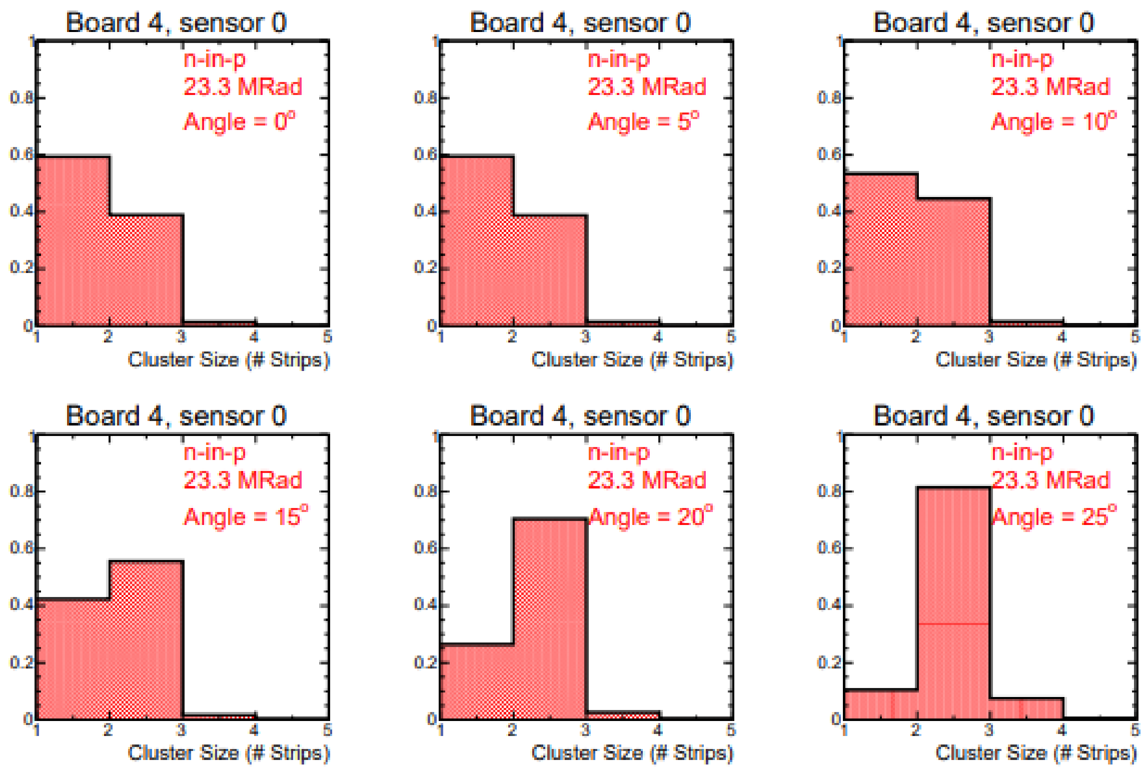


Figure 5.3.9: Cluster size distributions. Figures taken from [11].

Quality is never an accident; it is always the result of intelligent effort.

John Ruskin.

6 UT testbeam analysis - measurement of the charge sharing in planar silicon sensors

This chapter is dedicated to describing a selected analysis performed using the testbeam data collected using prototypes of silicon microstrip sensors for the UT tracking detector. The chapter starts by presenting the testbeam experiments' general idea, followed by a section on an experimental setup. The final section describes the study on charge sharing phenomena and the proposed correction.

6.1 Experimental setup

6.1.1 The beam

The tests were performed at CERN SPS North Area, see Figure 1.2.1. The beam consisted of secondary particles produced by the interaction of high intensity 450 GeV/c primary proton beam with a fixed target made of beryllium and lead. The beam had average energy of $180 GeV$. The beam typically produced four spills/minute. The spill lasted about 4 seconds, and it was intermittent by 40 seconds window with no beam. For most of the data taking, the beam size was collimated down to about 0.5 cm in diameter.

6.1.2 Timepix3 telescope

The essential tool that allows making many silicon sensor performance studies is the TimePix3 telescope, presented in Figure 6.1.1.

This device consists of 8 active planes of 300 μm thick p-on-n sensors bump bonded to the Timepix3 ASIC, divided equally between two arms. The position of each plane along the z-axis is adjustable,

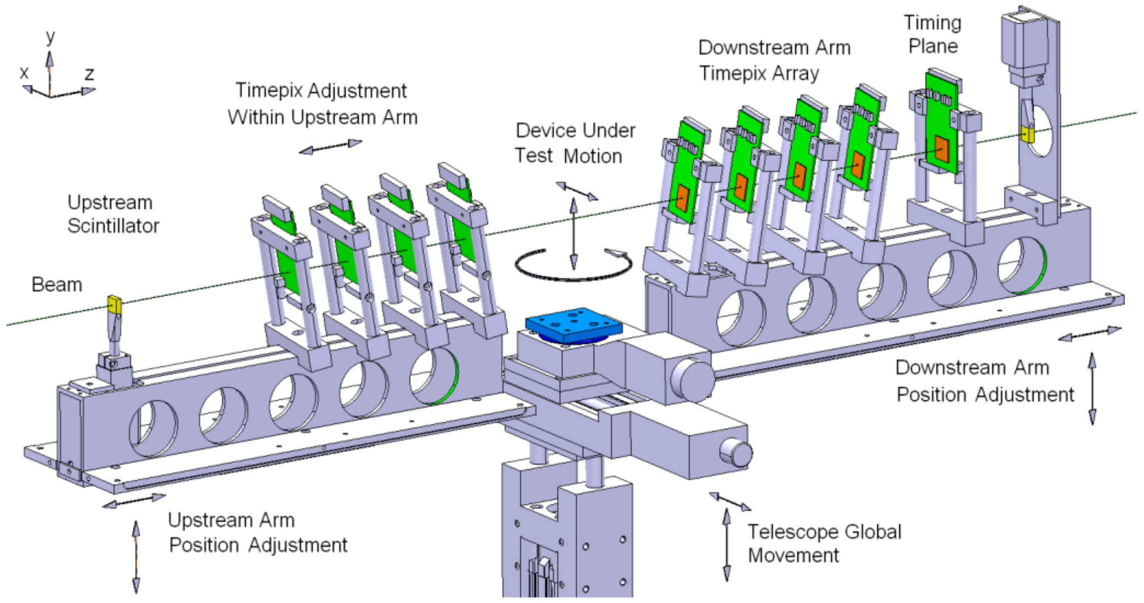

y
x
z
Timepix Adjustment Within Upstream Arm
Upstream Scintillator
Beam
Device Under Test Motion
Downstream Arm Timepix Array
Timing Plane
Downstream Arm Position Adjustment
Upstream Arm Position Adjustment
Telescope Global Movement

Figure 6.1.1: Layout of the Timepix Telescope mechanics, pixel planes and scintillators with respect to the beam axis. Figure taken from [15]

typically the distance between each plate is set to be approximately 30 mm, and plates are rotated to an angle of 9 deg in both x and y-axis to improve position reconstruction. There is a 200 mm gap between two telescope arms, which is used to place Device Under Test (DUT), see photography 6.1.2 taken during one of the testbeam experiments. The DUT is housed inside a metal box designed to fit into the gap and provides an airtight dry environment cooled via a Peltier device. The DUT was installed on a motion stage allowing angular rotations, and x and y translations.

To allow the DUT acquisition to trigger two scintillators are placed upstream and downstream of the telescope. The telescope acquisition system does not require any trigger signal since it works in a so-called data-driven read-out mode in which the data package of each pixel hit is sent immediately after Time-over-Threshold (ToT) conversion (note, that the hit is sent out only if it passes an analogue threshold, what allows to discriminate the noise hits). To synchronize DUT clusters with associated Telescope tracks, the information about the trigger timestamp is added to the data recorded by the telescope. To leverage the tracking information, the algorithm to match the UT clusters with corresponding telescope track was implemented.

6.1.2.1 TimePix3 Telescope Tracking

To find the position where the particle interacts with the DUT sensor, the particle trajectory needs to be reconstructed using the information provided by all Telescope sensors. This procedure starts by finding clusters. The cluster is a collection of neighboring pixels in which the measured signal exceeds a certain threshold, and such a measurement is called a hit. To add the hit to the cluster, its timestamp must lie within the 100 nanosecond window surrounding the timestamp of the seed hit. The timestamp of the cluster is a minimum of the timestamps associated with each of the hits belonging to the cluster.

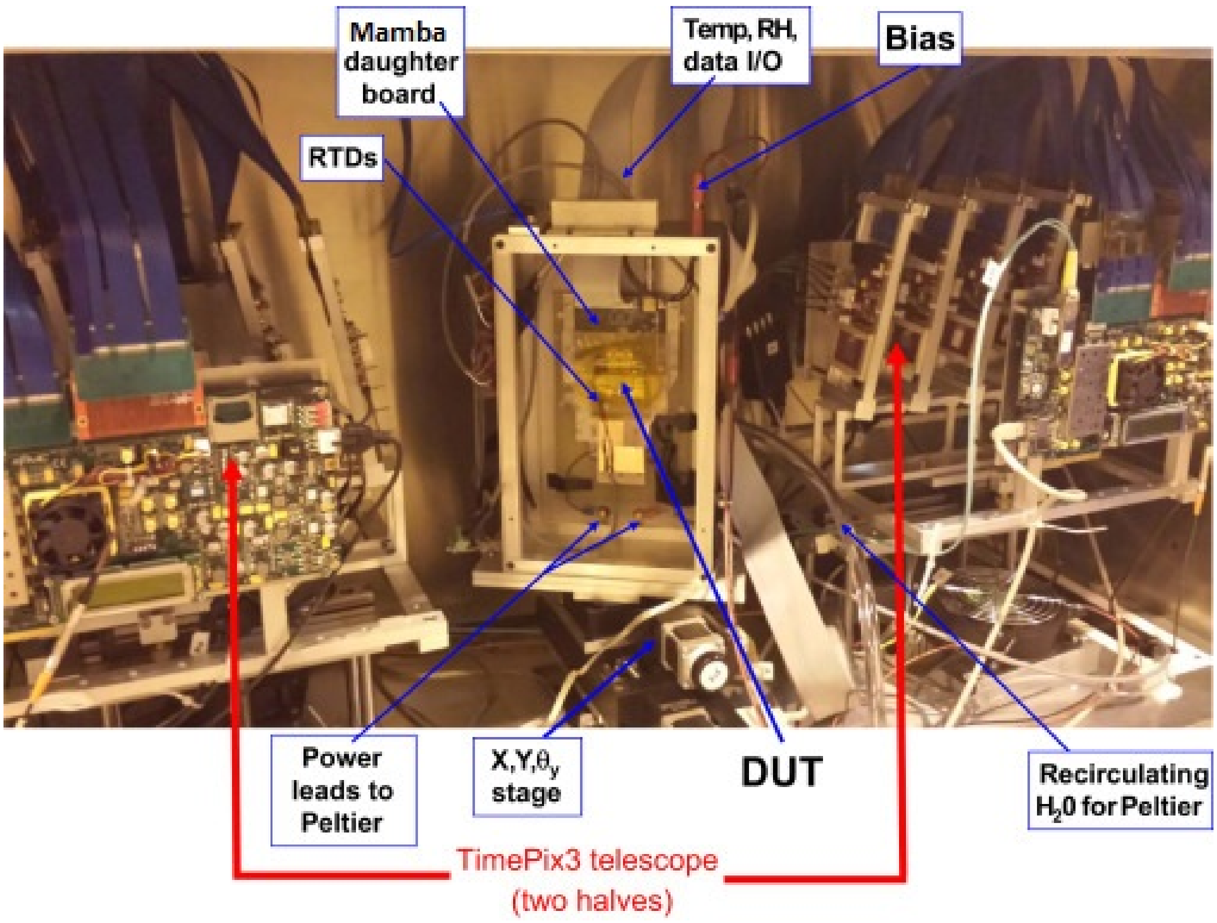


Figure 6.1.2: The photo of the Timepix Telescope and the DUT inside the box.

The cluster charge is a sum of charges of the constituent hits. The x and y positions of the cluster are calculated using the center of gravity method:

$$\{x, y\}_{cluster} = \frac{\sum_{i=0}^{n} \{x, y\}_i S_i}{\sum_{i=0}^{n} S_i} \tag{6.1}$$

where $\{x, y\}_i$ is the x or y coordinates of ith pixel and S_i its signal.

The clusters are then used as input for the tracking algorithm, which is based on the timing information of the clusters. The tracking algorithm starts by taking a cluster from the first plane and then looks for the matching cluster on the second plane. The hits are considered matched when the maximal time difference between them is within ten nanoseconds window. These two clusters constitute a track seed which is then extrapolated to the next plane, excluding the device under test, looking for a cluster within a cone with an opening angle of 0.01 radians and the ten nanoseconds time window. The procedure ends when all planes have been considered. Figure 6.1.3 shows four examples of Timepix3 tracks.

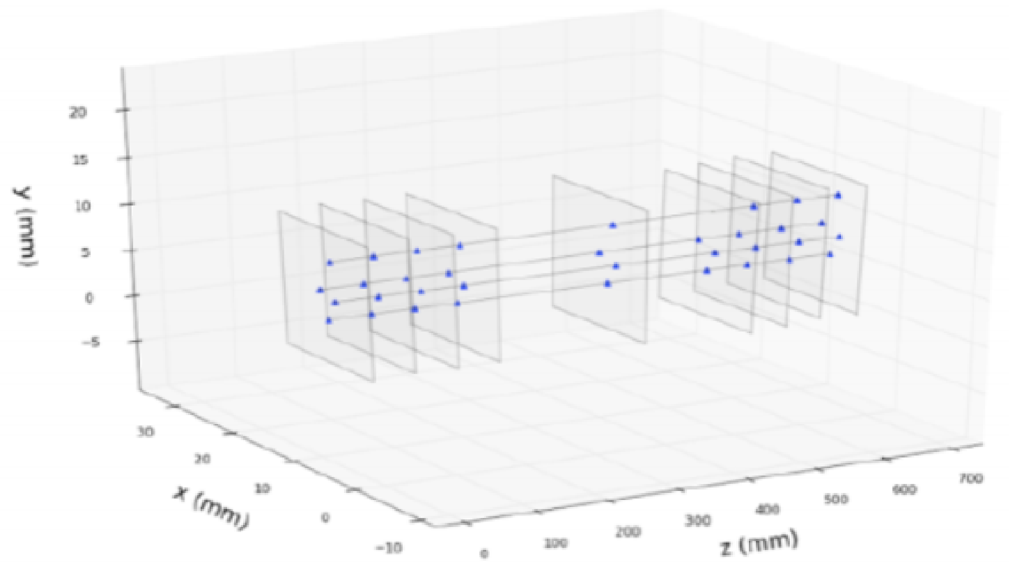


Figure 6.1.3: Four exemplary tracks reconstructed in the Timepix3 telescope. Figure taken from [117].

6.1.3 Read-out electronic

During all of the testbeam experiments conducted from 2015 till 2017, the SALT ASIC was not ready. Therefore, the sensor was read-out by the Beetle chip, which is the ASIC used as a front-end chip in Velo and TT during Run 1 and Run 2. The Beetle chip can read-out the signals from 128 channels and returns semi-Gaussian analog pules. The detailed description of the Beetle chip data processing can be found in [102]. In contrary to the SALT ASIC, the Beetle chip has no digital signal processing module. Thus, data have to be digitalized and processed by the external components. The analog data read-out by the Beetle chip was digitalized by the Mamba board designed by INFN in Milan and produced by Nuclear Instruments [3]. The processing of the digitalized data was done by the software described in chapter 5.

6.2 Testbeam studies and the results

In the course of the last five years, the LHCb UT Upgrade team has organized a number of testbeam campaigns to test new prototyped sensors. Figure 6.2.1 presents the photography of a sensor that was used during the testbeam with equipment which allowed to cool them and collect the data.

During this testbeams, the following detailed measurements were performed:

- Landau distribution as a function of bias voltage. This test scenario was executed to measure the influence of radiation fluence on a Signal to Noise ratio (S/N). For each sensor, the data was collected at bias voltages ranging from well below the full depletion region (about 50V) to the region well above it (usually 500V).

- Cluster size vs. bias voltage and track angle. This study was performed to understand the effect of charge sharing in high irradiated sensors.

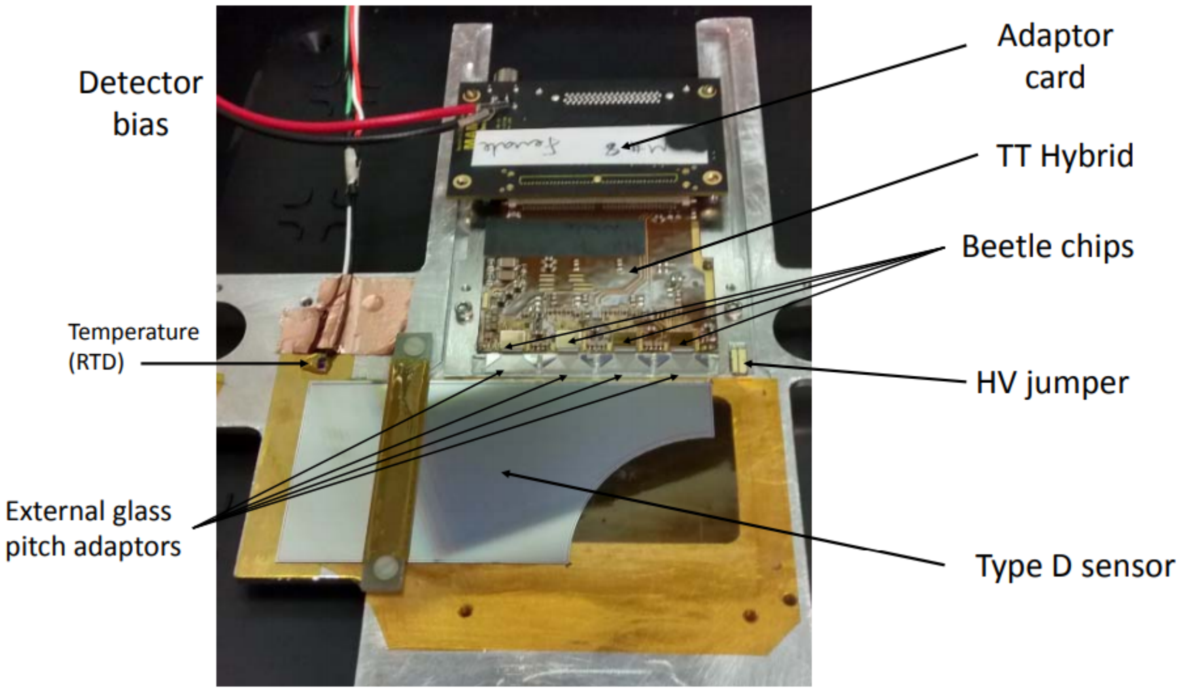


Figure 6.2.1: Photo of the D-type sensor that were studied during 2015 testbeam.

- Cluster charge vs. cluster size and inter-strip position. In this section, the team investigates the dependence of the collected charge on the cluster size. With no radiation effects, one would expect that the collected charge to be independent of the cluster size. However, radiation effects can lead to a difference which was investigated.
- Cluster size and resolution vs. angle. Studies of the detector performance were also carried out at angles ranging from normal incidence to 30 o (with respect to the normal to the sensor).
- Charge collection near the quarter-circle region. One type of sensor that will constitute the UT sensor has a quarter-circle region where there are no strips. This study aimed to investigate whether there is any drop in charge collected as a cluster approaches the quarter-circle radially.

The results of this studies lead to the following publications [11] [9] [10]. To get a broader picture of the UT testbeam studies, one can also refer to these PhD thesis [98] and [89].

6.3 Cross-talk correction

One of the effects that may influence sensor's performance, that was observed during testbeam, was an asymmetric cross-talk between channel that was attributed to their capacitative couplings. This effect can be measured by studying the signal asymmetry between $(N-1)$-th and $(N+1)$th channel, where N-th channel is a cluster seed strip position (so called cluster centre strip). The effect was studied separately for odd and even channels since it was observed, that this cross-talk effect is stronger for even channels, see Figure 6.3.1.

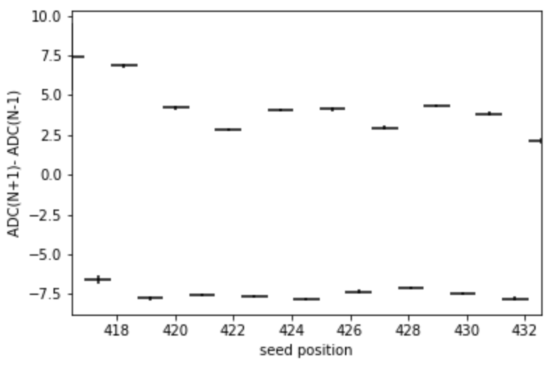


Figure 6.3.1: The charge collection difference $(ADC(N+1) - ADC(N-1))$ as a function of the strip position. The statistical uncertainty, which were calculated for each bin independently, are shown as one standard deviation.

To get a better understanding of the signal cross-talk effect on a collected data, the difference between $ADC(N-1)$ and $ADC(N+1)$ as a function of a cluster charge was studied. Figure 6.3.2 shows that this difference is strongly dependent on a cluster charge. Thus any proposed correction should take this fact into consideration. The idea to correct this effect is to create a two-dimensional map with a strip position and cluster charge as indices and a correction as a value. The correction can be modeled using Gaussian distribution since it was empirically discovered that the charge difference is normally distributed. Therefore the correction algorithm splits the cluster data into bins according to strip position N and the cluster charge, then for each of the bin, the correction is calculated by fitting Gaussian distribution to the charge difference $ADC(N-1)$ and $ADC(N+1)$. Figure 6.3.5 present this correction calculation procedure for cluster seed charge between 300 and 400 ADC, and Figure 6.3.3 shows the entire 2D correction map.

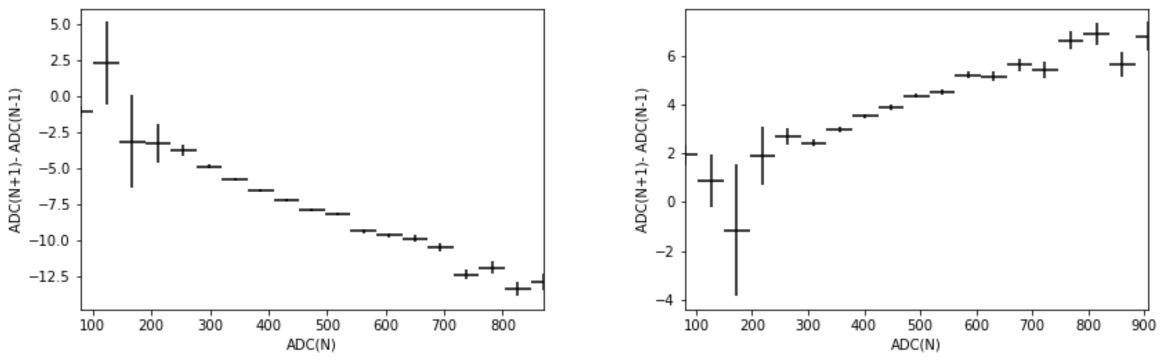


Figure 6.3.2: The measured charge difference $(ADC(N+1) - ADC(N-1))$ as a function of the cluster charge collected on channel N. The left plot shows the difference for even strips while right plot for odd ones

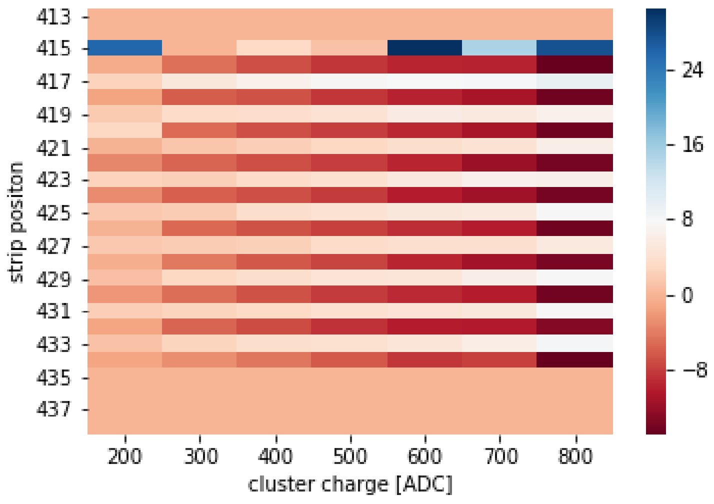


Figure 6.3.3: Cross-talk correction map. The color indicates value of the correction obtained from the Gaussian fit to the charge difference $(ADC(N+1) - ADC(N-1))$.

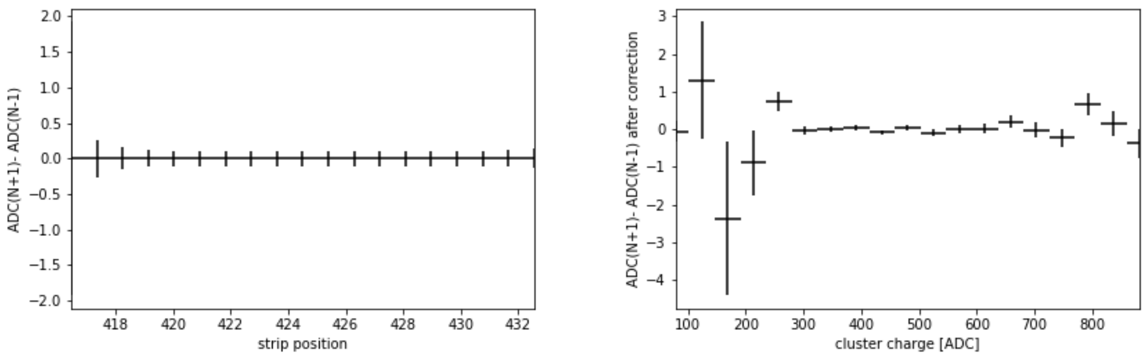


Figure 6.3.4: The charge difference $(ADC(N+1) - ADC(N-1))$ after 2D correction as a function of the cluster charge collected in channel N (left) and cluster charge (right).

After application of this correction the cross-talk was removed successfully, since the residual difference is below 2 ADC count, see Figure 6.3.4. All DUTs have similar odd–even cross-talk corrections. The even/odd cross-talk correction was similar for all of the DUTs, but not identical. Therefore, each sensor had a unique cross-talk correction to remove this bias.

6.4 Charge sharing

Charge sharing is manifested when a particle passing between two strips and generates charge cloud encompassing them. In this case, the charge can be collected on both strips, see Figure 6.4.1. The theoretical explanation of this effect can be seen as a transverse diffusion of the holes (or electrons) during the drift towards the strip implant [130].

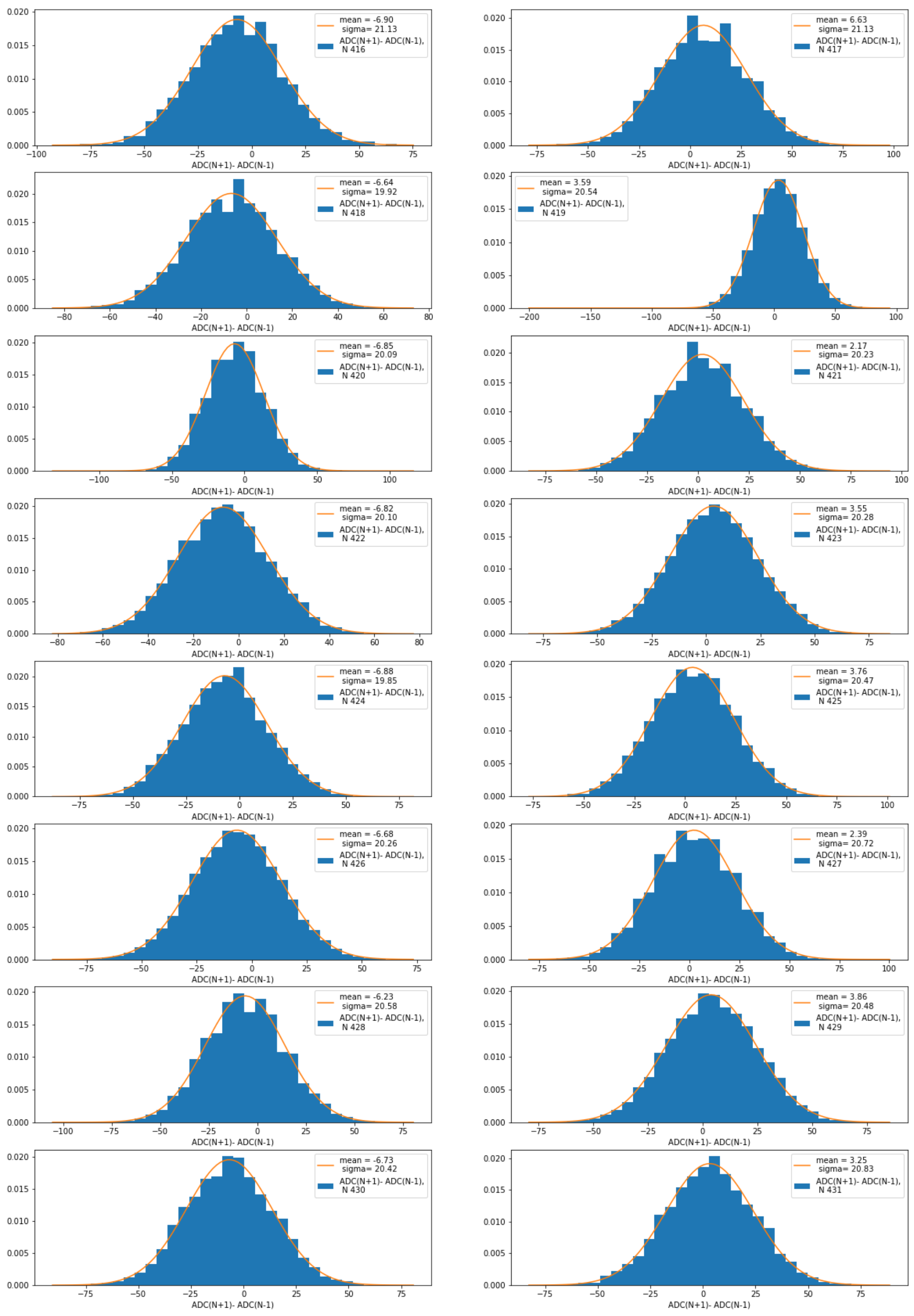


Figure 6.3.5: The cross-talk correction procedure visualization. Left plots shows charge difference for odd channels while the right plots presents same distribution for even channels. The data was fitted with a Gaussian distribution.

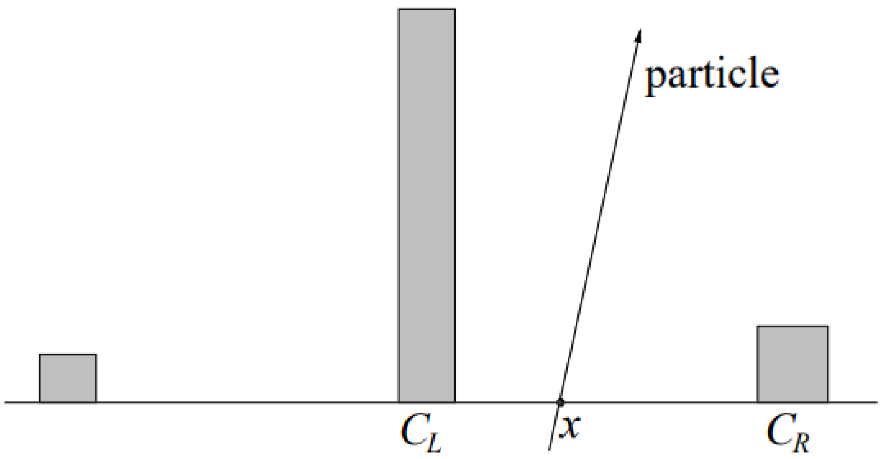


Figure 6.4.1: Cartoon of the charge deposited by a particle passing through the silicon. The charge, $C_{\{L,R\}}$, is deposited on the strips left and right of the particle's impact point x. On the far-left strip some charge is measured as well due to capacitative coupling. Figure adapted from [138]

This phenomenon can be quantified via parameter η, proposed in [24], given by:

$$\eta = \frac{Q_{left}}{Q_{left} + Q_{right}} \tag{6.2}$$

The effect of charge sharing were analyzed with respect to the cluster inter-strip position and the rotation angle. Each distribution was fitted with an error function model given by:

$$f(x;\sigma) = a\left(1 - erf(\frac{x-\mu}{\sqrt{2}\sigma})\right) + b \tag{6.3}$$

where a, b are constants, μ is a mean inter-strip position, σ is a width and erf is the Gauss error function given by:

$$erf(x) = \frac{2}{\pi}\int_x^0 e^{-t^2}dt \tag{6.4}$$

This non-linear charge sharing model can be interpreted as a smeared step function in which the width of the charge sharing (σ) equals the width of the Gaussian smearing. The full derivation of this formula can be found in [138]. The purpose of this analysis was to measure the dependence of charge sharing η versus the track angle. Different incidence angles were obtained by rotating the stage where the DUT box was mounted around the y axis.

The data that were used within this analysis were collected during the May 2016 testbeam. This section presented studies related to one selected unirradiated mini sensor produced by Hamamatsu Photonics [2]. These data were not used in any publications. The sensor is $p^+ - on - n$ and has a thickness of 320μm and a constant strip pitch of 190μm. The clusters that were considered during the presented have to lie within the beam region and be matched with a telescope tracks. Moreover, the reconstruction quality of those matched tracks should also be satisfactory ($\chi^2 ndf < 10$). Finally the cross-talk correction described in section 6.3 was applied.

Figure 6.4.2 present the η distributions integrated overall connected and not masked channels of the mini sensor. Each subfigure is dedicated to present the η for a different particle incidence angle. The charge sharing seems to be symmetric, thus no further correction was recommended.

To get a better understanding of the charge sharing, its dependence versus the cluster inter-strip position for various rotation positions around the sensor's y-axis is shown in Figure 6.4.3. Figure 6.4.4 presents same distribution but with model 6.3 fitted to the data. One can find that the effect of clockwise and anticlockwise rotations around the y-axis is the same. The σ as a function of the particle incident angle have a minimum at zero degrees and is symmetrically distributed around the minimum. The rotation around the y-axis was used to increase the charge sharing between neighbouring strips, which can be quantified using the value of σ obtained from the fit. The overall behaviour of the η function is exactly as we would expect. For the small rotation angles the curve is very close to the step function, which corresponds to a very moderate charge sharing, i.e., particle must cross the sensor very close to the mid-position between the strips to deposit charge on both strips and form multi-strip clusters. As the rotation angle of the DUT sensor increases we observe that the section of the curve that exhibits linear behaviour grows. Still, quite large strip pitch significantly confines the linear charge sharing region, that in case of the largest tested angle ($10\ ^{\circ}$), extends in the interval $\pm$ 0.2 about the inster-strip position 0. Since the typical track angle expected at LHCb is approximately $8\ ^{\circ}$ one can expect very moderate charge sharing, and thus, low rate of multiple-strip clusters. As a consequence, the single hit spatial resolution registered at UT detector may be very close to the binary resolution.

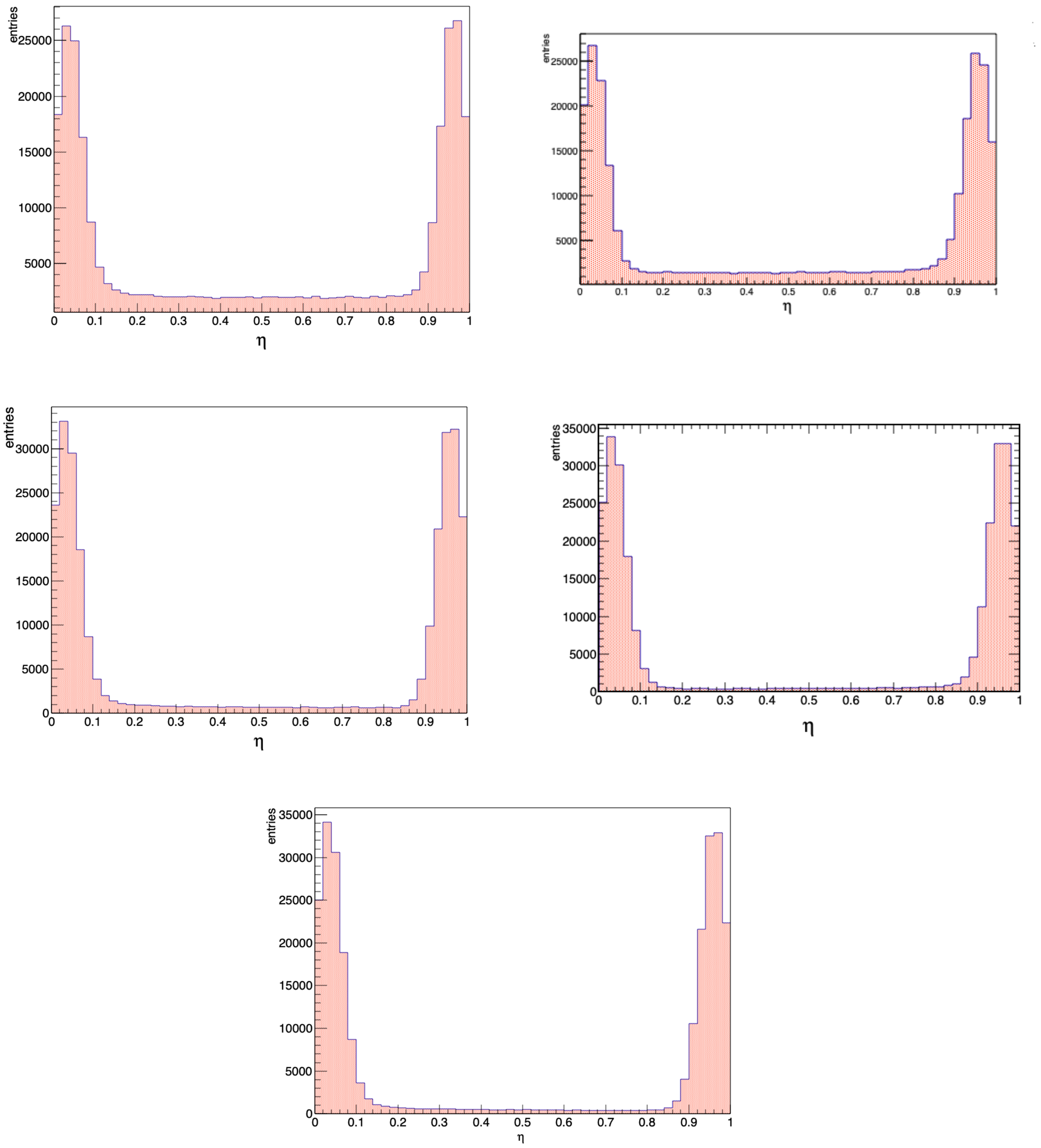


Figure 6.4.2: Distribution of a charge sharing η obtained for particles incident at a different angle. Upper row 10° and -10°, middle row -2° and 2°, bottom row 0° - perpendicular to the x axis.

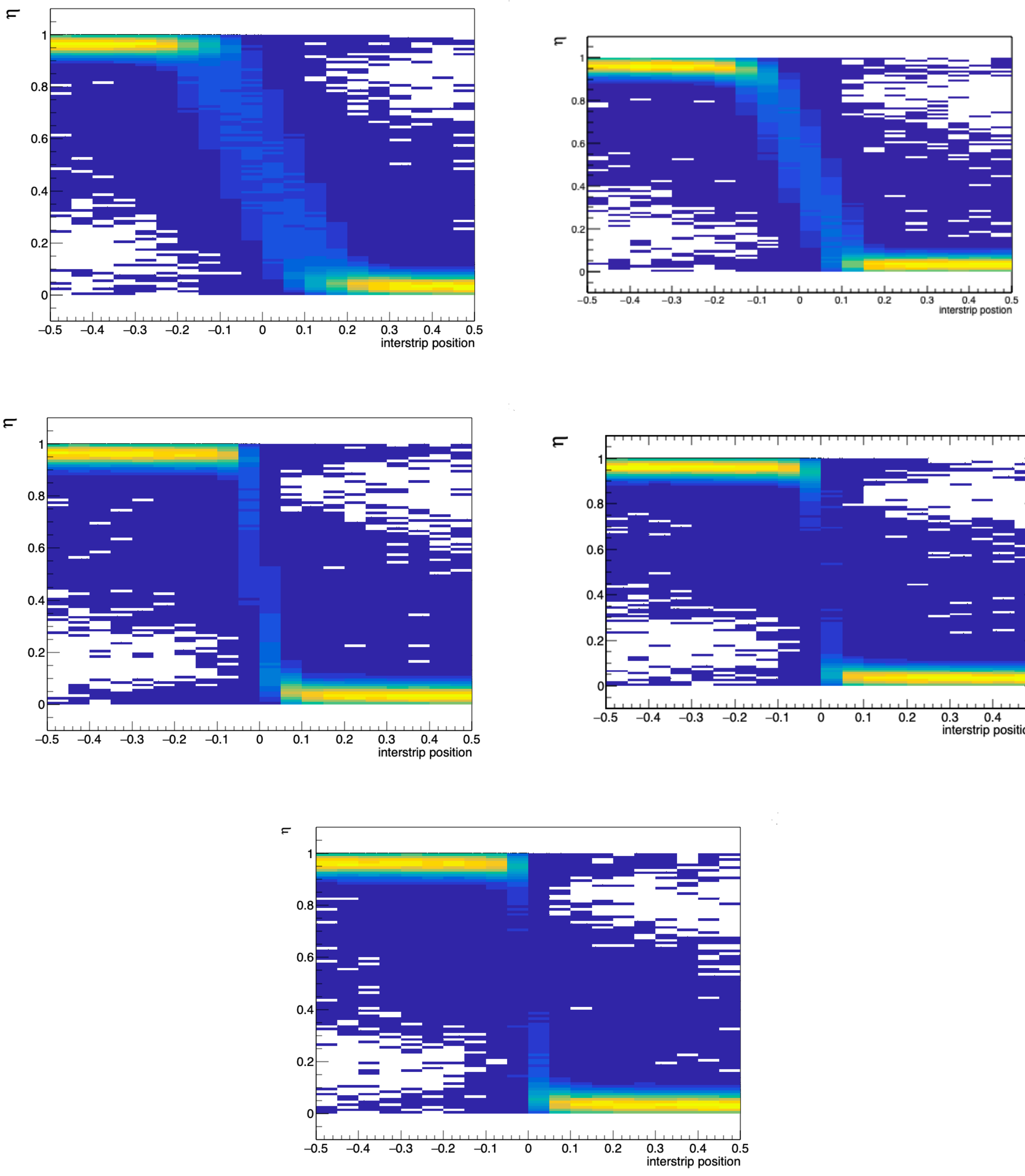


Figure 6.4.3: Distribution of a charge sharing η versus inter-strip position obtained for particles incident at a different angles. Upper row 10° and -10°, middle row -2° and 2°, bottom row 0° - perpendicular to the x axis.

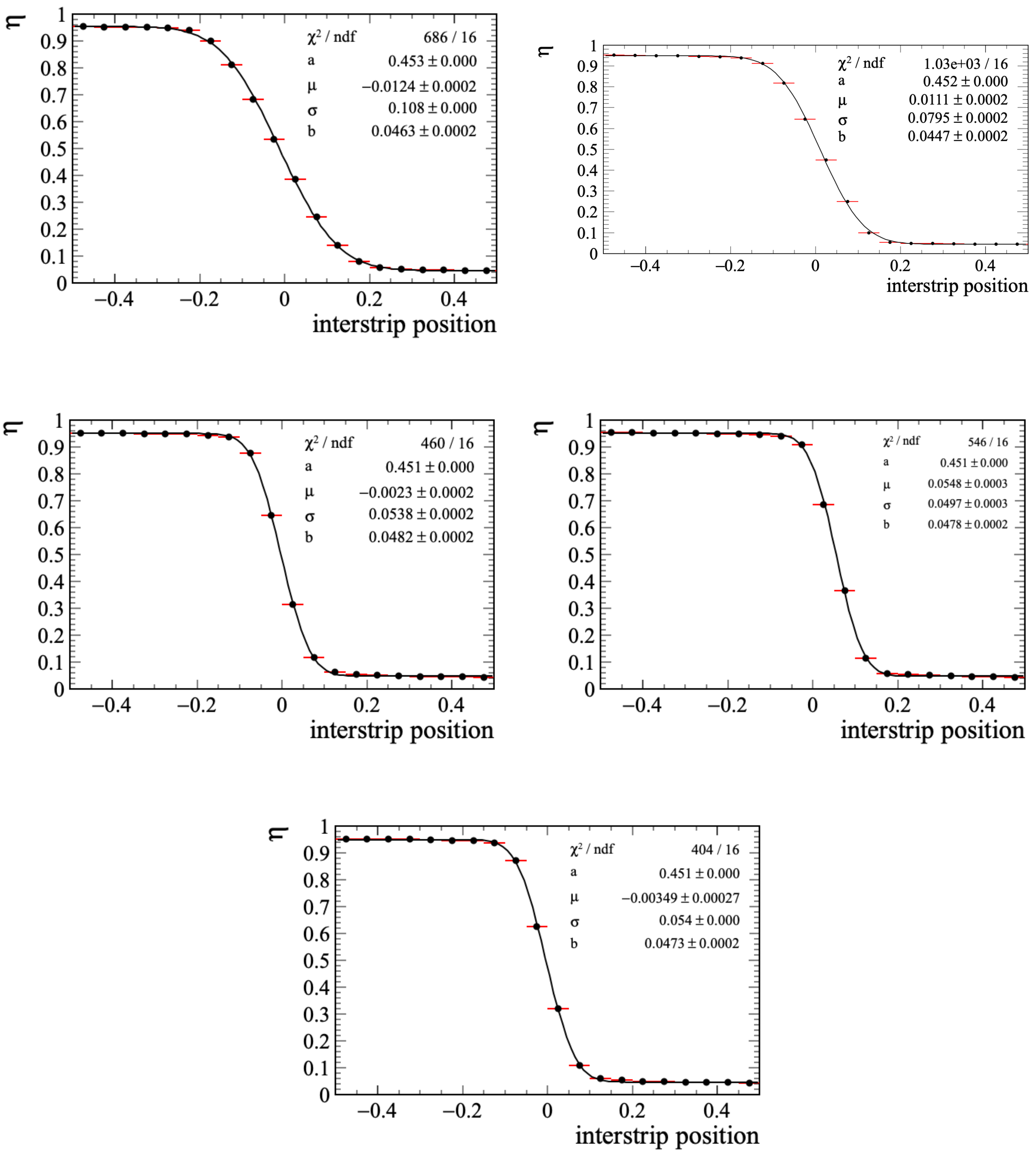


Figure 6.4.4: Charge sharing η as a function of the cluster inter-strip position obtained for different rotation angles around y axis. Upper row 10° and -10°, middle row -2° and 2°, bottom row 0° - perpendicular to the x axis. The black solid line is a charge sharing model 6.3 fitted to the data.

I'm smart enough to know that I'm dumb.

Richard Feynman

7 Summary and Outlook

This thesis discusses two projects, which were performed during the author's work for the LHCb Collaboration.

The first one was related to the design and improvement of the algorithm dedicated to reconstructing long-lived particles, such as K_s or Λ hadrons in the LHCb experiment. This algorithm is used to reconstruct the particles that decay outside of the Velo detector and depends on the input from the tracking station situated downstream from the Velo. Thus, it is called Downstream tracking. This reconstruction algorithm's time budget is minimal because it is executed as a part of a real-time LHCb trigger system. However, due to the number of particles created during each beam crossing, the previous implementation of the tracking procedure a considerable amount of so-called ghost tracks. Those tracks are the ones that do not represent a trajectory of a real particle. To significantly reduce such cases, two machine learning classifiers were trained and deployed. Within this project, a number of models, including Logistic Regression, *k*NN, Boosted Decision Trees, and deep neural networks, were tested. That kind of enhancement of the tracking reconstruction has never been deployed before. Each of the presented models was carefully tested, and its hyperparameters were optimized using various strategies, including Bayesian Optimization. Within the process of model validation, two novel methods for model prediction interpretation were proposed.

As a result of this work, the track reconstruction efficiency of the Downstream tracking was improved from 81.4% to 85.3 %, while the rate of misreconstructed tracks was lowered by a relative 18 %. Those numbers were obtained using a simulated sample of $B_0 \rightarrow J/\Psi K_s^0$. These changes were integrated into the LHCb software and are used in the official reconstruction of LHCb data. Also, the new algorithm reduces the runtime of the pattern recognition by 45%, with respect to its predecessor.

The Downstream tracking algorithm was also tested using the real data. The performance of the algorithm was evaluated via reconstruction of the $K_S \rightarrow \pi^+ + \pi^-$ and $\Lambda_0 \rightarrow p^+ + \pi^-$. The new algorithm allows reconstructing 6.5 % more signals than the previous algorithm's implementation. Furthermore, the tuning of the classification threshold value can decrease the background by 18% while sacrificing only 3% of the signal.

The presented within this thesis Downstream reconstruction algorithm is conceptually similar to the one that will be used to collect the data after the LHCb Upgrade. Within this thesis, a couple of further enhancements were proposed. Some of them are related to the new neural network architectures and one to the new loss function, which takes into consideration the imbalanced class problem.

The second part of this thesis was related to the design and implementation of the Upstream Tracker raw data emulation and performance monitoring software platform. The result of this study an application TbUT has been created. This software was essential for processing the data collected during the testbeam experiments and obtained results were essential for a number of published papers by the UT group. In the future, this software will be used to perform calibration of the UT processing algorithms as well as for performance diagnostics of the UT detector. The final section presents a selected analysis of the testbeam data.

References

[1] Mirror worlds. https://steemit.com/life/@pjheinz/mirror-worlds. Accessed: 2020-04-20.

[2] Hamamatsu photonics. URL https://www.hamamatsu.com/jp/en/index.html.

[3] Mamba board: Multi beetle chip read out system. URL http://it.rapidcast.it/mamba-board-multi-beetle-chip-read-out-system/.

[4] Apache airflow official website. URL http://airflow.apache.org.

[5] Scribe: Transporting petabytes per hour via a distributed, buffered queueing system. URL https://engineering.fb.com/data-infrastructure/scribe/.

[6] The why and how of differential signaling. URL https://www.allaboutcircuits.com/technical-articles/the-why-and-how-of-differential-signaling.

[7] SciPy 1.0 Contributors. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature Methods*, 17(3):261–272, Mar 2020. ISSN 1548-7105. doi: 10.1038/s41592-019-0686-2. URL https://doi.org/10.1038/s41592-019-0686-2.

[8] Martín Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

[9] Andrea Abba et al. Testbeam studies of pre-prototype silicon strip sensors for the LHCb UT upgrade project. Technical Report LHCB-PUB-2015-015. CERN-LHCb-PUB-2015-015. LHCb-PUB-2015-015, CERN, Geneva, May 2015. URL http://cds.cern.ch/record/2019712. 25 pages, 20 figures.

[10] Andrea Abba et al. Studies of pre-prototype sensors for the UT Upgrade project. Technical Report LHCb-PUB-2015-006. CERN-LHCb-PUB-2015-006, CERN, Geneva, Mar 2015. URL https://cds.cern.ch/record/2002512.

[11] Andrea Abba et al. Study of prototype sensors for the Upstream Tracker Upgrade. Technical Report LHCb-PUB-2016-007. CERN-LHCb-PUB-2016-007, CERN, Geneva, Mar 2016.

[12] Yaser S. Abu-Mostafa, M. Magdon-Ismail, and H.T. Lin. *Learning from Data: A Short Course*. AMLBook.com, 2012. ISBN 9781600490064. URL https://books.google.pl/books?id=iZUzMwEACAAJ.

[13] M. Adinolfi et al. Performance of the LHCb RICH detector at the LHC. *Eur. Phys. J.*, C73:2431, 2013. doi: 10.1140/epjc/s10052-013-2431-9.

[14] S. Agostinelli et al. GEANT4: A Simulation toolkit. *Nucl. Instrum. Meth. A*, 506:250–303, 2003. doi: 10.1016/S0168-9002(03)01368-8.

[15] Kazuyoshi Akiba, , et al. The Timepix Telescope for High Performance Particle Tracking. *Nucl. Instrum. Methods Phys. Res., A*, 723(arXiv:1304.5175):47–54. 14 p, Apr 2013. doi: 10.1016/j.nima.2013.04.060. URL https://cds.cern.ch/record/1543139. Comments: 14 pages, 9 figures.

[16] Philip Allport. Applications of silicon strip and pixel-based particle tracking detectors. *Nature Reviews Physics*, 1(9):567–576, Sep 2019. ISSN 2522-5820. doi: 10.1038/s42254-019-0081-z. URL https://doi.org/10.1038/s42254-019-0081-z.

[17] A. Augusto Alves, Jr. et al. The LHCb Detector at the LHC. *JINST*, 3:S08005, 2008. doi: 10.1088/1748-0221/3/08/S08005.

[18] Edgar Anderson. The species problem in iris. *Annals of the Missouri Botanical Garden*, 23:457–509, 1936. URL https://www.biodiversitylibrary.org/part/4079.

[19] Marina Artuso. Silicon sensors implemented on p-type substrates for high radiation resistance applications. *Nucl. Instrum. Meth. A*, 582:835–838, 2007. doi: 10.1016/j.nima.2007.07.106.

[20] Barbosa-Marinho and Others. *LHCb outer tracker: Technical Design Report*. Technical Design Report LHCb. CERN, Geneva, 2001. URL https://cds.cern.ch/record/519146.

[21] Barbosa-Marinho and Others. *LHCb VELO (VErtex LOcator): Technical Design Report*. Technical Design Report LHCb. CERN, Geneva, 2001. URL https://cds.cern.ch/record/504321.

[22] and others Barbosa-Marinho. *LHCb muon system: Technical Design Report*. Technical Design Report LHCb. CERN, Geneva, 2001. URL http://cds.cern.ch/record/504326.

[23] G. Barrand et al. GAUDI - A software architecture and framework for building HEP data processing applications. *Comput. Phys. Commun.*, 140:45–55, 2001. doi: 10.1016/S0010-4655(01)00254-5.

[24] E. Belau et al. The Charge Collection in Silicon Strip Detectors. *Nucl. Instrum. Meth.*, 214:253, 1983. doi: 10.1016/0167-5087(83)90591-4.

[25] I Belyaev, T Brambach, N H Brook, N Gauvin, G Corti, K Harrison, P F Harrison, J He, C R Jones, M Lieng, G Manca, S Miglioranzi, P Robbe, V Vagnoni, M Whitehead, and J Wishahi and. Handling of the generation of primary events in gauss, the LHCb simulation framework. *Journal of Physics: Conference Series*, 331(3):032047, dec 2011. doi: 10.1088/1742-6596/331/3/032047. URL https://doi.org/10.1088%2F1742-6596%2F331%2F3%2F032047.

[26] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

[27] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13:281–305, 2012. URL http://dblp.uni-trier.de/db/journals/jmlr/jmlr13.html#BergstraB12.

[28] R P Bernhard and Others. The LHCb Silicon Tracker. Technical Report LHCb-2007-126. CERN-LHCb-2007-126, CERN, Geneva, Nov 2007. URL https://cds.cern.ch/record/1070312.

[29] H. Bethe. Zur theorie des durchgangs schneller korpuskularstrahlen durch materie. *Annalen der Physik*, 397(3):325–400, 1930. doi: 10.1002/andp.19303970303. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/andp.19303970303.

[30] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[31] L. Breiman, J. Friedman, C.J. Stone, and R.A. Olshen. *Classification and Regression Trees*. The Wadsworth and Brooks-Cole statistics-probability series. Taylor & Francis, 1984. ISBN 9780412048418. URL https://books.google.pl/books?id=JwQx-WOmSyQC.

[32] Eric Brochu, Vlad M. Cora, and Nando De Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *CoRR*, abs/1012.2599, 12 2010.

[33] J. Brownlee. *Statistical Methods for Machine Learning: Discover how to Transform Data into Knowledge with Python*. Machine Learning Mastery, 2018. URL https://books.google.pl/books?id=386nDwAAQBAJ.

[34] R. Brun and F. Rademakers. ROOT: An object oriented data analysis framework. *Nucl. Instrum. Meth. A*, 389:81–86, 1997. doi: 10.1016/S0168-9002(97)00048-X.

[35] Sz. Bugiel, R. Dasgupta, M. Firlej, T. Fiutowski, M. Idzik, M. Kuczynska, J. Moron, K. Swientek, and T. Szumlak. SALT, a dedicated readout chip for high precision tracking silicon strip detectors at the LHCb upgrade. *Journal of Instrumentation*, 11(02):C02028–C02028, feb 2016. doi: 10.1088/1748-0221/11/02/c02028. URL https://doi.org/10.1088%2F1748-0221%2F11%2F02%2Fc02028.

[36] I S Bulbakov, E V Atkin, and A G Voronin. High speed SLVS transmitter and receiver. *Journal of Physics: Conference Series*, 675(4):042035, feb 2016. doi: 10.1088/1742-6596/675/4/042035. URL https://doi.org/10.1088%2F1742-6596%2F675%2F4%2F042035.

[37] T.T. Böhlen, Francesco Cerutti, Motofumi Chin, Alberto Fasso, Alfredo Ferrari, Pablo G. Ortega, Andrea Mairani, Paola Sala, George Smirnov, and Vasilis Vlachoudis. The fluka code: Developments and challenges for high energy and medical applications. *Nuclear Data Sheets*, 120: 211–214, 06 2014. doi: 10.1016/j.nds.2014.07.049.

[38] Nicola Cabibbo. Unitary Symmetry and Leptonic Decays. *Phys. Rev. Lett.*, 10:531–533, 1963. doi: 10.1103/PhysRevLett.10.531.

[39] Nicola Cabibbo. Unitary Symmetry and Leptonic Decays. *Phys. Rev. Lett.*, 10:531–533, 1963. doi: 10.1103/PhysRevLett.10.531.

[40] O Callot. Downstream Pattern Recognition. Technical Report LHCb-2007-026. CERN-LHCb-2007-026, CERN, Geneva, Mar 2007. URL https://cds.cern.ch/record/1025827.

[41] O. Callot and M. Schiller. PatSeeding: A standalone track reconstruction algorithm. 8 2008.

[42] Angel Camacho Rozas, A.J. Rozas, Pertti Aarnio, Habtamu Abie, P. Abreu, A. Zucchini, and Gianni Zumerle. The delphi detector at lep. *Nuclear Instruments and Methods in Physics Research Section A Accelerators Spectrometers Detectors and Associated Equipment*, A303:223–276, 11 1990.

[43] J. Charles, A. Höcker, H. Lacker, S. Laplace, F. R. Le Diberder, J. Malclés, J. Ocariz, M. Pivk, and L. Roos. Cp violation and the ckm matrix: assessing the impact of the asymmetric b factories. *The European Physical Journal C*, 41(1):1–131, May 2005. ISSN 1434-6052. doi: 10.1140/epjc/s2005-02169-1. URL http://dx.doi.org/10.1140/epjc/s2005-02169-1.

[44] S. Chatrchyan et al. The CMS Experiment at the CERN LHC. *JINST*, 3:S08004, 2008. doi: 10.1088/1748-0221/3/08/S08004.

[45] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4232-2. doi: 10.1145/2939672.2939785. URL http://doi.acm.org/10.1145/2939672.2939785.

[46] J. H. Christenson, J. W. Cronin, V. L. Fitch, and R. Turlay. Evidence for the 2π decay of the k_2^0 meson. *Phys. Rev. Lett.*, 13:138–140, Jul 1964. doi: 10.1103/PhysRevLett.13.138. URL https://link.aps.org/doi/10.1103/PhysRevLett.13.138.

[47] LHCb collaboration. Trigger schemes. URL http://lhcb.web.cern.ch/lhcb/speakersbureau/html/TriggerScheme.html.

[48] LHCb Collaboration. LHCb VELO Upgrade Technical Design Report. Technical Report CERN-LHCC-2013-021. LHCB-TDR-013, Nov 2013. URL https://cds.cern.ch/record/1624070.

[49] LHCb Collaboration. LHCb Tracker Upgrade Technical Design Report. Technical Report CERN-LHCC-2014-001. LHCB-TDR-015, Feb 2014. URL https://cds.cern.ch/record/1647400.

[50] LHCb Collaboration. Angular analysis of the $B^0 \to K^{*0}\mu^+\mu^-$ decay using 3 fb^{-1} of integrated luminosity. *JHEP*, 02(CERN-PH-EP-2015-314. CERN-PH-EP-2015-314. LHCB-PAPER-2015-051):104. 92 p, Dec 2015. doi: 10.1007/JHEP02(2016)104. URL https://cds.cern.ch/record/2115087. All figures and tables, along with any supplementary material and additional information, are available at https://lhcbproject.web.cern.ch/lhcbproject/Publications/LHCbProjectPublic/LHCb-PAPER-2015-051.html.

[51] LHCb Collaboration. Observation of $j/\psi p$ resonances consistent with pentaquark states in $\Lambda_b^0 \to j/\psi K^- p$ decays. *Phys. Rev. Lett.*, 115:072001, Aug 2015. doi: 10.1103/PhysRevLett.115.072001. URL https://link.aps.org/doi/10.1103/PhysRevLett.115.072001.

[52] The ATLAS Collaboration. The ATLAS experiment at the CERN large hadron collider. *Journal of Instrumentation*, 3(08):S08003–S08003, aug 2008. doi: 10.1088/1748-0221/3/08/s08003. URL https://doi.org/10.1088%2F1748-0221%2F3%2F08%2Fs08003.

[53] P Cortese et al. ALICE: Physics performance report, volume I. *J. Phys. G*, 30:1517–1763, 2004. doi: 10.1088/0954-3899/30/11/001.

[54] Gloria Corti, Marco Cattaneo, Philippe Charpentier, Markus Frank, Patrick Koppenburg, Pere Mato, Florence Ranjard, Stefan Roiser, Ivan Belyaev, and Guy Barrand. Software for the lhcb experiment. *Nuclear Science, IEEE Transactions on*, 53:1323 – 1328, 07 2006. doi: 10.1109/TNS.2006.872627.

[55] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[56] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, Dec 1989. ISSN 1435-568X. doi: 10.1007/BF02551274. URL https://doi.org/10.1007/BF02551274.

[57] Adam Davis, Michel De Cian, Adam Mateusz Dendek, and Tomasz Szumlak. PatLongLivedTracking: a tracking algorithm for the reconstruction of the daughters of long-lived particles in LHCb. Technical Report LHCb-PUB-2017-001. CERN-LHCb-PUB-2017-001, CERN, Geneva, Jan 2017. URL https://cds.cern.ch/record/2240723.

[58] Adam Dendek. utorch: a minimal implementation of autograd and pytorch, 2021. URL https://github.com/adendek/utorch.

[59] Denis Derkach et al. Machine-learning-based global particle-identification algorithms at the lhcb experiment. *Journal of Physics: Conference Series*, 1085:042038, 09 2018. doi: 10.1088/1742-6596/1085/4/042038.

[60] Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *Ann. Statist.*, 32(2):407–499, 04 2004. doi: 10.1214/009053604000000067. URL https://doi.org/10.1214/009053604000000067.

[61] "Christian Elsässer". "$\bar{b}b$ production angle plots". URL https://lhcb.web.cern.ch/lhcb/speakersbureau/html/bb_ProductionAngles.html.

[62] Lyndon Evans and Philip Bryant. LHC machine. *Journal of Instrumentation*, 3(08):S08001–S08001, aug 2008. doi: 10.1088/1748-0221/3/08/s08001. URL https://doi.org/10.1088%2F1748-0221%2F3%2F08%2Fs08001.

[63] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7 (7):179–188, 1936.

[64] Mike Lamont for the LHC team. LHC Report: Level best. Dernières nouvelles du LHC : à son meilleur niveau. (BUL-NA-2012-345. 47/2012):4, Nov 2012. URL https://cds.cern.ch/record/1493444.

[65] Brendan Fortuner. Can neural networks solve any problem? URL https://towardsdatascience.com/can-neural-networks-really-learn-any-function-65e106617fc6.

[66] R. Fruhwirth. Application of Kalman filtering to track and vertex fitting. *Nucl. Instrum. Meth. A*, 262:444–450, 1987. doi: 10.1016/0168-9002(87)90887-4.

[67] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional Computing Series. Pearson Education, 1994. ISBN 9780321700698. URL https://books.google.pl/books?id=6oHuKQe3TjQC.

[68] RS Gilmore. *Single particle detection and measurement*. Taylor Francis Group, United Kingdom, 1992. ISBN 085067550.

[69] V.V. Gligorov and Mike Williams. Efficient, reliable and fast high-level triggering using a bonsai boosted decision tree. *JINST*, 8(arXiv:1210.6861):P02013, Oct 2012. doi: 10.1088/1748-0221/8/02/P02013. URL https://cds.cern.ch/record/1490195. Comments: 10 pages, 2 figures.

[70] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[71] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR. URL http://proceedings.mlr.press/v15/glorot11a.html.

[72] Yoav Goldberg. A primer on neural network models for natural language processing, 2015.

[73] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014. URL http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

[74] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. The MIT Press, 2016. ISBN 0262035618, 9780262035613.

[75] Griffiths, D. *Introduction to Elementary Particles*. Physics textbook. Wiley, 2008. ISBN 9783527618477. URL https://books.google.pl/books?id=Wb9DYrjcoKAC.

[76] Charles R Harris and Others. Array programming with numpy. *Nature*, 585(7825):357–362, Sep 2020. ISSN 1476-4687. doi: 10.1038/s41586-020-2649-2. URL https://doi.org/10.1038/s41586-020-2649-2.

[77] Jonathan Robert Harrison. Radiation damage studies in the LHCb VELO detector and searches for lepton flavour and baryon number violating tau decays, 2014. URL https://cds.cern.ch/record/1712972. Presented 16 05 2014.

[78] H. He and Y. Ma. *Imbalanced Learning: Foundations, Algorithms, and Applications*. Wiley, 2013. ISBN 9781118646335. URL https://books.google.pl/books?id=YqQJngEACAAJ.

[79] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, 2015.

[80] R.D. Heuer and H. Schopper. *LEP - The Lord of the Collider Rings at CERN 1980-2000: The Making, Operation and Legacy of the World's Largest Scientific Instrument*. Springer Berlin Heidelberg, 2009. ISBN 9783540893011. URL https://books.google.pl/books?id=4Sysug8oMkUC.

[81] Jerry L. Hintze and Ray D. Nelson. Violin plots: A box plot-density trace synergism. *The American Statistician*, 52(2):181–184, 1998.

[82] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9: 1735–80, 12 1997. doi: 10.1162/neco.1997.9.8.1735.

[83] Andreas Hocker et al. TMVA - Toolkit for Multivariate Data Analysis with ROOT: Users guide. TMVA - Toolkit for Multivariate Data Analysis. Technical Report physics/0703039, CERN, Geneva, Mar 2007. URL https://cds.cern.ch/record/1019880.

[84] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Netw.*, 2(5):359–366, July 1989. ISSN 0893-6080.

[85] Sergey Ioffe and Christian Szegedy. "batch normalization: Accelerating deep network training by reducing internal covariate shift", 2015.

[86] Ian Jolliffe and Jorge Cadima. Principal component analysis: A review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374:20150202, 04 2016. doi: 10.1098/rsta.2015.0202.

[87] Rudolph Emil Kalman and Others. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[88] Andrej Karpathy. A recipe for training neural networks. URL http://karpathy.github.io/2019/04/25/recipe/.

[89] Matthew Jordan Kelsey. Measurements of Beauty Production Cross Sections and Fragmentation Fraction Ratios using the LHCb Detector, Apr 2018. URL http://cds.cern.ch/record/2670646. Presented 2018.

[90] Vardan Khachatryan et al. Observation of the rare $B_s^0 \to \mu^+\mu^-$ decay from the combined analysis of CMS and LHCb data. *Nature*, 522(CERN-PH-EP-2014-220. CERN-PH-EP-2014-220. CMS-BPH-13-007. LHCB-PAPER-2014-049):68–72. 46 p, Nov 2014. doi: 10.1038/nature14474. URL https://cds.cern.ch/record/1970675. Comments: Submitted to Nature. Correspondence should be addressed to cms-and-lhcb-publication-committees@cern.ch.

[91] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[92] M Kobayashi and T Maskawa. Cp-violation in the renormalizable theory of weak interaction. *Progr. Theor. Phys. (Kyoto), v. 49, no. 2, pp. 652-657*, 2 1973. doi: 10.1143/PTP.49.652.

[93] M. Laine, G. Nardini, and K. Rummukainen. First order thermal phase transition with 126 gev higgs mass, 2013.

[94] D.J. Lange. The EvtGen particle decay simulation package. *Nucl. Instrum. Meth. A*, 462:152–155, 2001. doi: 10.1016/S0168-9002(01)00089-4.

[95] Christiane Lefèvre. The CERN accelerator complex. Complexe des accélérateurs du CERN. Dec 2008. URL https://cds.cern.ch/record/1260465.

[96] LHCb Collaboration LHCb Experiment. General Photo, May 2016. URL https://cds.cern.ch/record/2151262.

[97] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection, 2017.

[98] Federica Lionetto. Measurements of Angular Observables of $B^0 \to K^{*0}\mu^+\mu^-$ and $B^0 \to K^{*0}e^+e^-$ Decays and the Upgrade of LHCb, Feb 2018. URL http://cds.cern.ch/record/2624938. Presented 22 Mar 2018.

[99] Anthony C. Little, Benedict C. Jones, and Lisa M. DeBruine. Facial attractiveness: evolutionary based research. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, 366(1571):1638–1659, Jun 2011. URL https://doi.org/10.1098/rstb.2010.0404.

[100] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf.

[101] Scott M. Lundberg, Gabriel G. Erion, and Su-In Lee. Consistent individualized feature attribution for tree ensembles. *CoRR*, abs/1802.03888, 2018. URL http://dblp.uni-trier.de/db/journals/corr/corr1802.html#abs-1802-03888.

[102] S Löchner and M Schmelling. The Beetle Reference Manual - chip version 1.3, 1.4 and 1.5. Technical Report LHCb-2005-105. CERN-LHCb-2005-105, CERN, Geneva, Nov 2006. URL https://cds.cern.ch/record/1000429.

[103] Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

[104] Dougal Maclaurin. *Modeling, inference and optimization with composable differentiable procedures*. PhD thesis, 2016.

[105] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, Dec 1943. ISSN 1522-9602. doi: 10.1007/BF02478259. URL https://doi.org/10.1007/BF02478259.

[106] Rukmani Mohanta and Anjan Giri. Flavour physics and cp violation. *Pramana - Journal of Physics*, 74:675–703, 05 2010. doi: 10.1007/s12043-010-0091-y.

[107] Guido Montúfar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks, 2014.

[108] I. Newton. *Philosophiae naturalis principia mathematica*. J. Societatis Regiae ac Typis J. Streater, 1687. URL https://books.google.pl/books?id=-dVKAQAAIAAJ.

[109] Christopher Olah. Understanding lstm networks. URL https://colah.github.io/posts/2015-08-Understanding-LSTMs.

[110] Adam Paszke et al. Pytorch: An imperative style, high-performance deep learning library, 2019.

[111] Sagar Patel, Prachi Talati, and Saniya Gandhi. Design of i2c protocol. 03 2019.

[112] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[113] P Rodríguez Pérez. The LHCb VERTEX LOCATOR performance and VERTEX LOCATOR upgrade. *Journal of Instrumentation*, 7(12):C12008–C12008, dec 2012. doi: 10.1088/1748-0221/7/12/c12008. URL https://doi.org/10.1088%2F1748-0221%2F7%2F12%2Fc12008.

[114] Foster Provost and Tom Fawcett. *Data Science for Business: What You Need to Know about Data Mining and Data-Analytic Thinking*. O'Reilly Media, Inc., 1st edition, 2013. ISBN 1449361323.

[115] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005. ISBN 026218253X.

[116] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144, 2016.

[117] Sophie Elizabeth Richards. Characterisation of silicon detectors for the LHCb Vertex Locator Upgrade, Nov 2017. URL https://cds.cern.ch/record/2626889. Presented 15 Dec 2017.

[118] E Rodrigues. Tracking definitions. Technical Report LHCb-2007-006. CERN-LHCb-2007-006, CERN, Geneva, Feb 2007. URL https://cds.cern.ch/record/1016937. revised version submitted on 2007-03-28 09:34:37.

[119] Sebastian Ruder. An overview of gradient descent optimization algorithms, 2016.

[120] James Rumbaugh, Ivar Jacobson, and Grady Booch. *Unified Modeling Language Reference Manual, The (2Nd Edition)*. Pearson Higher Education, 2004. ISBN 0321245628.

[121] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, Oct 1986. ISSN 1476-4687. doi: 10.1038/323533a0. URL https://doi.org/10.1038/323533a0.

[122] A.D. Sakharov. Violation of CP Invariance, C asymmetry, and baryon asymmetry of the universe. *Sov. Phys. Usp.*, 34(5):392–393, 1991. doi: 10.1070/PU1991v034n05ABEH002497.

[123] Julian Schwinger. The theory of quantized fields. vi. *Phys. Rev.*, 94:1362–1384, Jun 1954. doi: 10.1103/PhysRev.94.1362. URL https://link.aps.org/doi/10.1103/PhysRev.94.1362.

[124] M.L. Scott. *Programming Language Pragmatics*. Programming Language Pragmatics. Morgan Kaufmann, 2000. ISBN 9781558604421. URL https://books.google.pl/books?id=To3xpkvkPvMC.

[125] Andrew W. Senior et al. Improved protein structure prediction using potentials from deep learning. *Nature*, 577(7792):706–710, Jan 2020. ISSN 1476-4687. doi: 10.1038/s41586-019-1923-7. URL https://doi.org/10.1038/s41586-019-1923-7.

[126] Claude Elwood Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 7 1948. doi: 10.1002/j.1538-7305.1948.tb01338.x. URL https://ieeexplore.ieee.org/document/6773024/.

[127] Lloyd S Shapley. A value for n-person games. *Contributions to the Theory of Games*, 2(28):307–317, 1953.

[128] David Silver et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, jan 2016. ISSN 0028-0836. doi: 10.1038/nature16961.

[129] Torbjörn Sjöstrand, Stephen Mrenna, and Peter Skands. A brief introduction to pythia 8.1. *Computer Physics Communications*, 178(11):852–867, Jun 2008. ISSN 0010-4655. doi: 10.1016/j.cpc.2008.01.036. URL http://dx.doi.org/10.1016/j.cpc.2008.01.036.

[130] H. Spieler and Knovel (Firm). *Semiconductor Detector Systems*. Oxford science publications. OUP Oxford, 2005. ISBN 9780198527848. URL https://books.google.pl/books?id=qIXjBAAAQBAJ.

[131] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

[132] Erik Strumbelj and Igor Kononenko. An efficient explanation of individual classifications using game theory. *Journal of Machine Learning Research*, 11:1–18, 01 2010. doi: 10.1145/1756006.1756007.

[133] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998. URL http://www.cs.ualberta.ca/~sutton/book/the-book.html.

[134] Gerard 't Hooft. Symmetry Breaking Through Bell-Jackiw Anomalies. *Phys. Rev. Lett.*, 37:8–11, 1976. doi: 10.1103/PhysRevLett.37.8.

[135] M. Tanabashi et al. Review of particle physics. *Phys. Rev. D*, 98:030001, Aug 2018. doi: 10.1103/PhysRevD.98.030001. URL https://link.aps.org/doi/10.1103/PhysRevD.98.030001.

[136] CERN The LHCb Collaboration. Upgrade Software and Computing. Technical Report CERN-LHCC-2018-007. LHCB-TDR-017, CERN, Geneva, Mar 2018. URL https://cds.cern.ch/record/2310827.

[137] S Tolk, J Albrecht, F Dettori, and A Pellegrino. Data driven trigger efficiency determination at LHCb. Technical Report LHCb-PUB-2014-039. CERN-LHCb-PUB-2014-039, CERN, Geneva, May 2014. URL https://cds.cern.ch/record/1701134.

[138] J van Tilburg. Studies of the Silicon Tracker resolution using data. Technical Report LHCb-PUB-2010-016. CERN-LHCb-PUB-2010-016, CERN, Geneva, Aug 2010. URL http://cds.cern.ch/record/1286299.

[139] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[140] Lincoln Wolfenstein. Parametrization of the kobayashi-maskawa matrix. *Phys. Rev. Lett.*, 51:1945, 1983. doi: 10.1103/PhysRevLett.51.1945.

[141] Zhongxing Zhang et al. Exploring the clinical features of narcolepsy type 1 versus narcolepsy type 2 from european narcolepsy network database with machine learning. *Scientific Reports*, 8, 12 2018. doi: 10.1038/s41598-018-28840-w.