Dissertation

submitted to the

Combined Faculty of Mathematics, Engineering and Natural Sciences

of Heidelberg University, Germany

for the degree of

Doctor of Natural Sciences

Put forward by

Christian Hans-Jörg Sonnabend

born in Wiesbaden, Germany

Oral examination: 29.04.2026

Neural network cluster finding for the ALICE TPC online GPU processing

Referees:

Prof. Dr. Silvia Masciocchi

apl. Prof. Dr. Monica Dunford

Be curious enough to start and stubborn enough to finish.

Abstract

The ALICE Time Projection Chamber (TPC) is one of the main tracking and particle identification detectors of the ALICE experiment at CERN, designed for the study of Quantum Chromodynamics (QCD) and the Quark–Gluon Plasma (QGP). The upgrade of the TPC readout from Multiwire Proportional Chambers to Gas Electron Multipliers enables continuous readout at interaction rates up to 50 times higher than previously achievable. While being essential for the physics program of Run 3 and beyond, this upgrade introduces new challenges for both reconstruction algorithms and the online computing infrastructure.

The TPC reconstruction chain benefits significantly from GPU-based hardware acceleration, which also provides an ideal environment for modern machine learning techniques such as neural networks. This thesis develops a novel neural-network-based cluster reconstruction algorithm designed for online deployment in Run 4 and beyond. Two neural networks are trained: a classification network aimed at data-size reduction and improved performance in high-density tracking environments, and a regression network for charge deconvolution and the precise determination of cluster properties.

For the training and benchmarking of these algorithms, an ideal cluster finder, based on simulated data, is developed. The performance of the neural-network-based reconstruction is evaluated in Monte Carlo studies covering a wide range of track densities and detector occupancies, with continuous emphasis on the balance between physics performance and computational feasibility for online operation. The results are compared to the current reconstruction framework throughout the analysis.

The algorithm is subsequently validated on reconstructed real data from Pb–Pb collisions at interaction rates of 30–38 kHz. A cluster reduction of up to 18% is achieved while improving the track-fit quality (χ^2/NDF) and the $\mathrm{d}E/\mathrm{d}x$ separation power, with maintained particle identification performance. Finally, the scope of this work exceeds the original deployment timeline with a first successful commissioning in online proton–proton data taking at 507 kHz, where the data-size reduction is confirmed and shared cluster contributions to tracks are reduced.

Zusammenfassung

Die ALICE Time Projection Chamber (TPC) ist einer der wichtigsten Detektoren zur Spurrekonstruktion und Teilchenidentifikation des ALICE Experiments am CERN und wurde für die Untersuchung der Quantenchromodynamik (QCD) und des Quark-Gluon Plasmas (QGP) konzipiert. Das Upgrade der TPC Ladungsverstärkungskammern von Multiwire Proportional Chambers (MWPCs) auf Gas Electron Multipliers (GEMs) ermöglicht eine kontinuierliche Auslese bei Wechselwirkungsraten, die bis zu 50-mal höher sind als zuvor erreichbar. Obwohl dieses Upgrade für das Physikprogramm von Run 3 und darüber hinaus essenziell ist, führt es sowohl für die Rekonstruktionsalgorithmen als auch für die Online-Recheninfrastruktur zu neuen Herausforderungen.

Die TPC-Rekonstruktionskette profitiert erheblich von parallelisierten Berechnungen auf Grafikprozessoren, die zugleich eine ideale Umgebung für moderne Methoden des machinellen Lernens wie neuronalen Netzen bietet. In dieser Arbeit wird ein neuartiger, auf neuronalen Netzen basierender Clusterrekonstruktionsalgorithmus entwickelt, der für die Echtzeitdatenaufnahme in Run 4 und darüber hinaus ausgelegt ist. Es werden zwei neuronale Netze trainiert: Ein Klassifikationsnetzwerk mit dem Ziel der Datenmengenreduktion und einer verbesserten Leistung in dichten Tracking-Umgebungen sowie ein Regressionsnetzwerk zur besseren Ladungstrennung zwischen Clustern und zur präzisen Bestimmung der charakteristischen Clustereigenschaften.

Für das Training und Benchmarking dieser Algorithmen wird ein idealer Cluster Finder entwickelt, der auf simulierten Daten basiert. Die Leistung der auf neuronalen Netzen basierenden Rekonstruktion wird in Monte Carlo Studien untersucht, die ein breites Spektrum an Spurendichten abdecken, wobei sowohl die physikalische Leistungsfähigkeit als auch die Umsetzbarkeit für den Online-Betrieb im Foklus stehen. Die Ergebnisse werden im gesamten Analyseprozess mit dem aktuellen Rekonstruktionsframework verglichen.

Der Algorithmus wird anschließend anhand rekonstruierter, echter Daten aus Pb–Pb Kollisionen bei Wechselwirkungsraten von 30–38 kHz validiert. Dabei wird eine Clusterreduktion von bis zu 18% erreicht, während sich die Qualität des Spur-Fits (χ^2/NDF) sowie die dE/dx-Trennschärfe verbessern, bei gleichzeitig erhaltener Leistungsfähigkeit der Teilchenidentifikation. Schließlich überschreitet diese Arbeit den ursprünglichen Einsatzzeitplan durch eine erste erfolgreiche Inbetriebnahme während der Online-Rekonstruktion von Proton–Proton Daten bei 507 kHz, bei der die Datenmengenreduktion bestätigt und die Beiträge geteilter Cluster zu einzelnen Spuren reduziert werden.

Table of Contents

Motivation and outline of the thesis **1**

I Experimental measurements and theoretical concepts **3**

1.1 CERN and the ALICE experiment . . . 3

1.1.1 Electromagnetic interactions for detection . . . 5

1.1.2 Neutron capture process . . . 6

1.1.3 ITS - Inner Tracking System . . . 7

1.1.4 TPC - Time Projection Chamber . . . 8

1.1.5 TPC readout and cluster-finding . . . 15

1.1.6 Tracking and downstream processing . . . 16

1.2 Neural networks and hardware acceleration . . . 18

1.2.1 Hardware acceleration - CPU vs. GPU . . . 18

1.2.2 Fully connected feed-forward networks - Introduction . . . 20

1.2.3 Backpropagation and gradient descent . . . 21

1.2.4 Convolutional neural networks . . . 22

1.2.5 Open problems in the field of machine learning . . . 23

II Algorithmic approaches to cluster finding **25**

2.1 Monte Carlo cluster finding . . . 25

2.1.1 Looping tracks and occupancy tagging . . . 28

2.1.2 Assignment strategy and ambiguity resolution . . . 31

2.1.3 Performance of the MC clustering algorithm . . . 32

2.2 The ALICE TPC online cluster finder algorithm . . . 34

2.2.1 Working principles of the heuristic cluster finder . . . 34

2.3 Neural network cluster finder . . . 35

2.3.1 Working principle and input data . . . 36

2.3.2 Training data and selection strategy . . . 39

2.3.3 Investigated model architectures . . . 43

III Analysis and performance on Monte-Carlo data **45**

3.1 Reconstruction performance metrics . . . 45

3.2 Cluster and track reconstruction on simulated data . . . 45

3.3 Pb–Pb simulation without space-charge distortions . . . 48
3.3.1 Cluster residuals . . . 48
3.3.2 Impact of cluster resolution on tracking performance . . . 57
3.4 Pb–Pb simulation with space-charge distortions . . . 58
3.4.1 Cluster matching . . . 62
3.4.2 Impact of model size . . . 67
3.4.3 Results on track reconstruction level . . . 68
3.5 Extensions of this work . . . 80
3.5.1 Training on real data . . . 80
3.5.2 Momentum vector estimation . . . 81
3.5.3 Application to reconstructed clusters . . . 83
3.5.4 Cluster split-flag setting . . . 84
3.6 Prelude on the analysis on real data . . . 87

IV Particle identification and analysis on real data 89
4.1 Practical introduction to particle identification . . . 89
4.2 Clean sample selection and calibration of the Bethe-Bloch parameterization . 91
4.2.1 Particle identification on V0 decays and γ conversions . . . 92
4.2.2 $\mathrm{d}E/\mathrm{d}x$ resolution and separation power . . . 94
4.2.3 Bethe-Bloch parameter optimization . . . 97
4.3 Secondary corrections for particle identification . . . 98
4.3.1 Neural networks for the TPC PID response correction . . . 101
4.4 Invariant mass distributions: K^0_S, Λ, D^0 . . . 105
4.4.1 K^0_S and Λ mass resolution . . . 105
4.4.2 Dependence of number of clusters and tracks on the neural network threshold . . . 108
4.4.3 D^0 mass resolution . . . 112
4.5 Commissioning runs and online reconstruction . . . 115
4.5.1 Analysis of commissioning runs and comparison with standard reconstruction . . . 118
4.6 Summary . . . 121

V Computing performance of the neural network cluster finding algorithm 123
5.1 CPU-based computation . . . 123
5.2 Hardware accelerated computing . . . 127
5.2.1 Computational limits for online processing . . . 127
5.2.2 GPU kernel optimizations . . . 131
5.2.3 Single vs. half precision: Accuracy and resource usage . . . 132
5.2.4 Convolutional versus fully connected layers . . . 135

5.2.5 Compute speed evaluations in online processing . . . 138
5.2.6 Compute time evaluations of individual processing steps . . . 139
5.3 Loss landscape of the neural network . . . 140
5.4 The economic aspect . . . 147
5.5 Summary . . . 148

VI Summary and conclusion **149**

A Appendix **153**
A Algorithmic approaches to cluster finding . . . 153
B Monte Carlo . . . 153
B.1 Analysis on pp data . . . 154
B.2 Cluster residuals without space-charge distortion effects . . . 157
B.3 Efficiency, clone and fake-rates with space-charge distortion effects . . . 159
B.4 Studies with different peak finder algorithm and 3D convolutional networks . . . 162
C Real data analysis . . . 163
C.1 Real data analysis: LHC24ar . . . 163
C.2 D^0 analysis . . . 164
C.3 Online runs . . . 166
D Computing . . . 167
E Reproducibility . . . 170

B Bibliography **173**

C Acknowledgment **177**

Motivation and outline of the thesis

Machine learning (ML) is the field of study that gives computers the ability to learn without being explicitly programmed[1]. In recent years it has had a profound impact on every day life and has gained substantial popularity across all fields of science and research.
Parallelizable algorithms such as neural networks (NN) can make full use of dedicated hardware support such as graphics processing units (GPU), field-programmable gate arrays (FPGA) and application-specific integrated circuits (ASIC). In recent years, high performance computing (HPC) centers have shifted their focus from purely CPU-based systems to heterogeneous architectures (CPU + hardware accelerators) to meet the large data processing and analysis demands. The four major experiments at the European Organization for Nuclear Research (CERN), located around the Large Hadron Collider (LHC) make use of cutting-edge technological innovations in computing and machine learning. They combine the challenge of high data rates (terabyte-per-second (TB/s) scale) with the fundamental research conducted to gain insight to the best known and tested theory to date, describing the atomic and subatomic regime, the Standard Model of particle physics.

Data collected at these experiments is intrinsically high-dimensional and is processed in a parallelizable fashion. If the available hardware supports parallel execution, machine learning can be deployed optimally for data processing. Due to their computationally lightweight training and execution demands, boosted decision trees have a long tradition in data analysis within the LHC experiments. However, with the more general applicability to different problem sets and typically superior performance, neural networks become an uprising tool, both in analysis and calibration. The field of detector and data reconstruction has yet to experience the large scale application of machine learning. It is the goal of this thesis to explore such new endeavors in the ALICE (A Large Ion Collider Experiment) reconstruction process. The work is in parts inspired by previous analyses [1].

Large scale high energy physics experiments typically consider two processing steps for the collected data. The first stage is online processing, which handles data selection, compression, reconstruction, and calibration in real time, matching the data rates of the individual detectors. After that comes the offline processing, which encompasses any processing steps performed after the online processing is completed. The ALICE experiment at CERN has shifted the

[1] Arthur Samuel, 1959

online processing of its largest detector in terms of data size, the Time Projection Chamber (TPC), to a fully GPU-based reconstruction. It is therefore the motivation for this thesis to explore novel, neural-network-based algorithms for the online processing of the ALICE TPC data. Specifically, this thesis will consider the task of cluster finding, which represents the first step on the online processing farm and is a crucial part for any further operations, such as tracking and calibrations. It is a concern that the current cluster finding algorithm falls short on the aspect of charge deconvolution and accurate cluster reconstruction in high-density environments encountered in the ALICE TPC in Run 4 and beyond. This motivates the choice of topics for this thesis and the two tasks which will be its primary focus:

1. **Cluster classification** - The identification of whether a cluster is useful for downstream processing or physics performance or not. Several physics aspects, such as looping tracks, noise charges and cluster overlap will be discussed. The effect on tracking and reconstruction by rejecting certain types of clusters will be analyzed and demonstrated. Besides data quality, a major impact of this algorithm is the reduction in final data size.

2. **Cluster regression** - This concerns the estimation of the center-of-gravity position, width and total charge of a cluster. A comparison with current heuristic cluster finding algorithms is made and the impact on the final particle identification performance will be shown. Additionally, a novel momentum vector estimate is introduced, which is not present in the current cluster finding algorithm.

The validation of the new cluster finding algorithm on both simulated and real data is considered a crucial step for this project. The investigation on Monte Carlo (MC) data is conducted in chapter (III). An analysis on real data follows in chapter (IV). As this is one of the pioneer projects in applying neural networks in online, GPU-based processing in all LHC experiments and the first of its kind in the ALICE experiment, a major focus is put on its computing performance. Not only does the algorithm need to satisfy the physics criteria, but also processing speed and memory limitations must be taken into consideration to fit into the online reconstruction scheme. This is dealt with in detail in chapter (V). The scope of this project spans from physics validation and compute performance benchmarking to a first application of this algorithm in a commissioning run conducted at the ALICE experiment (October 2025), with a final deployment forseen in Run 4 of LHC.

I Experimental measurements and theoretical concepts

1.1 CERN and the ALICE experiment

Humanity's quest to uncover the fundamental laws of nature is a combination of reasoning (theory) and experience (experiment). Deductions and testable conclusions are inferred from observing real-world phenomena and dedicated experiments, sparking ideas about new ways to probe nature. It is the task of physicists around the world to carry out such measurements with ever-increasing precision and give explanations as well as reproducible setups for quantities of interest, called observables. In modern times, the pinnacle of this endeavor is a collaboration of thousands of physicists united in their pursuit to understand the laws governing the atomic and subatomic regime. One such collaboration manifested itself in 1954 near Geneva, where the European Organization for Nuclear Research (CERN) was founded. Central to its operation nowadays is the LHC, the largest circular particle accelerator in the world. Protons, heavy ions and most recently (2025) light ions are accelerated and brought to collision at the four major experiments (ALICE, ATLAS, CMS, LHCb) located around the accelerator ring. One of them, named A Large Ion Collider Experiment (ALICE) is a dedicated experiment to research the state of matter formed in heavy-ion collisions, called the Quark-Gluon Plasma (QGP) [2].

The regime of strongly interacting particles is described by quantum chromodynamics (QCD), the gauge theory of the strong interaction. QCD is based on the non-Abelian gauge symmetry $SU_C(3)$ under which color charge arises. Unlike electric charge, color charge is associated with the fundamental representation of $SU_C(3)$ and comes in three states (labeled red, green and blue). Consequently, the emerging massive, spin-$\frac{1}{2}$ particles called quarks carry a color charge, which is invariant under gauge transformations of the group (hence the subscript C). The interaction of color charge is mediated by massless, spin-1 bosons, which emerge as the generators of the Lie algebra of $SU_C(3)$. There are eight such bosons in QCD, called gluons, which differ by their charge composition (color-octet) and can, in contrast to the bosons of quantum electrodynamics (photons), self-interact. The interaction strength is characterized by the coupling constant of the strong force $\alpha_s(k) = 1/(\beta_0 \ln(k^2/\Lambda^2))$ (where k is the energy of the process involved and Λ the QCD energy scale parameter) and limits the effective range

of interactions of QCD to the (sub-)atomic, femtometer scale. This coupling constrains color interactions to so-called flux-tubes, which govern the behavior at low local energy densities (non-perturbative regime). As the distance between quarks in a bound state increases, so does the energy density of the flux tube between them. At sufficiently high energies, a new quark-antiquark pair is created from vacuum, recombining with the original quarks. For this reason color charge can not be observed independently at ordinary conditions, a behaviour called confinement [3]. With an increase in energy density of the system, k^2 and hence α_s decreases. This effect is referred to as the running of the coupling, allowing color charges to exist in a deconfined state at sufficiently high energy densities, known as asymptotic freedom. At this stage, quarks and gluons exist independently without bound states. The energy densities of heavy-ion collisions at the LHC achieve such a deconfined state of matter, a phase known as the QGP.

The ultra-relativistic momenta carried by the gluons lead to gluon splitting and an overall domination of the QGP by gluonic degrees of freedom at the initial stages. Gluon-gluon interactions lead to recombination and a build-up of pressure gradients within the medium, forcing an expansion and rapid cooling. The two nearly spherical collision partners create an elliptical region of interaction, with the remaining non-interacting nucleons (spectators) continuing in their direction of flight. The asymmetry of the medium, due to typically non-central interactions of the initial collision partners, imprints itself in the momentum distribution of final state particles. Near the crossover temperature ($\approx$ 155 MeV) [4]–[6], the medium has cooled through its expansion to a point where the local energy density is insufficient to produce particles thermally. This region at which the strong coupling between quarks becomes dominant again, effectively freezing the thermal production of final state particle yields, is labeled as the chemical freeze-out. After further expansion, the spatial separation of particles leads to the kinetic freezeout, where the momentum distributions are fixed and particles are freely streaming towards the detector layers. Due to their high masses charm and beauty quarks cannot be produced in intermediary interactions. At analysis level, particles containing charm and beauty quarks are categorized as the heavy-flavor sector and are of particular relevance for the ALICE experiment as they must emerge from the primary collision, therefore experiencing the full evolution of the QGP. In contrast, mesons and baryons containing up, down and strange quarks can be produced kinematically and are categorized in the light-flavor sector. Unstable compositions in both the heavy and light flavour sector typically decay before or within the first detector layers into long-lived, stable particles which are measured in the detectors of the experiment. The analysis requires an accurate measurement and reconstruction of particle tracks, but also a reliable particle identification. In the ALICE experiment, designated detectors for such tasks are cylindrically arranged around the interaction point (primary vertex) in what is commonly referred to as the central barrel. The primary detectors used in reconstruc-

tion allow physics measurements within momentum ranges of ~10 MeV/c up to ~100 GeV/c.

In the following, the fundamental electromagnetic interactions with the detector material and the most important detectors of the central barrel reconstruction of ALICE will be illustrated in more detail. Particular emphasis will be put on the reconstruction capabilities and the individual aspects each detector design can contribute. Strengths and weaknesses of each design show why each of the detection systems is crucial for the track reconstruction process and where trade-offs between accuracy and quantity of measurements are made.

1.1.1 Electromagnetic interactions for detection

At ordinary scales of particle detectors, electromagnetic interactions of charged particles are dominantly used for particle detection. Several choices of different detector designs exploit different interactions that these particles undergo when traversing the sensitive material. Three processes are most common for electromagnetic interactions of charged particles: Ionization, Coulomb scattering and Bremsstrahlung. The most common type of interaction is ionization, where a charged particle traversing the detector material interacts with the shell electrons of an atom or a molecule of the material, releasing a charge carrier (electron) from a bound state to a freely moving state. Figure (1.1.1) illustrates this process for an incoming particle (orange) on a stationary nucleus.

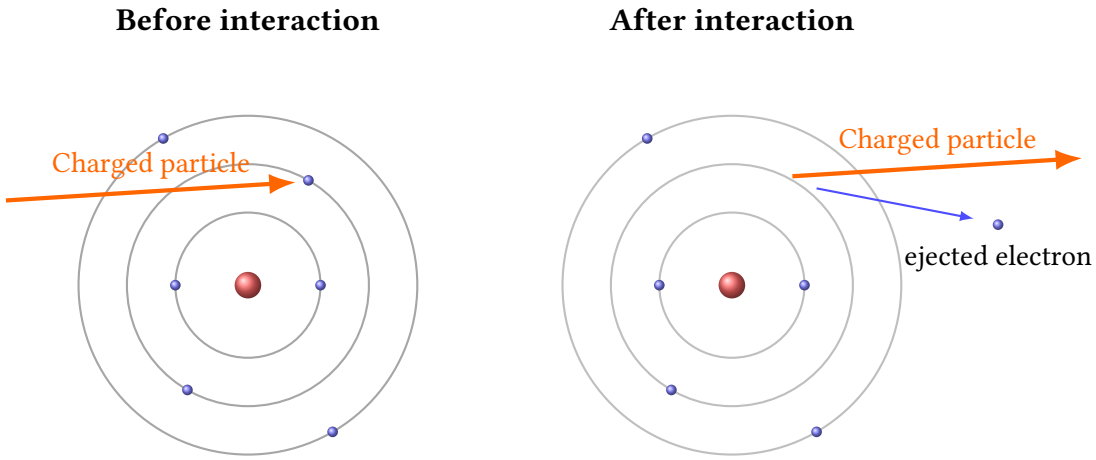


Fig. 1.1.1: Illustration of charged particle interaction with a molecule of detector material and release of an electron via the process of ionization.

Furthermore, if the energy transfer is sufficiently high, ionized electrons can themselves ionize molecules, releasing other electrons, so-called δ-rays. The released charges are then freely moving with a comparably small remaining fraction of the momentum exchange from the interaction. In comparison to the process of ionization, Coulomb interactions are not inelastic, but rather describe the process of elastic scatterings upon interactions with atomic Coulomb

(electric) fields. This process can occur several times within the layers of dense materials, as interactions become statistically more likely and is therefore referred to as multiple scattering. Reducing the material budget of detection layers results in a lower transfer of energy to the material, leading to more precise measurements in subsequent detection layers. This is of particular relevance for low momentum particles and is therefore one of the key design challenges for high energy particle physics experiments. The final process of interaction is the release of photons, as particles interact with the Coulomb field of material, predominantly of nuclei. This process is referred to as Bremsstrahlung and leads to a reduction of the particle momentum without the direct production of free charge carriers. The Bremsstrahlung photons can release charges as a secondary effect via the three mechanisms of photoelectric absorption, the Compton effect or pair production. For the incident particle, the parallel ($p_{\parallel}$) and orthogonal ($p_{\perp}$) components of the momentum vector scale with γ^4 and γ^6 in an electric field (where γ is the Lorentz factor ($\gamma = 1/\sqrt{1-(v/c)^2}$, v the particle velocity, c the speed of light in vacuum). Since the relativistic energy of a particle is equal to $E = \gamma mc^2$, the components scale with $p_{\parallel} \sim m^{-4}$ and $p_{\perp} \sim m^{-6}$ [7]. Therefore, the particle species which will experience the highest energy losses by Bremsstrahlung are particles with low masses, typically electrons ($m = 511\ \mathrm{keV}/c^2$). Bremsstrahlung then becomes the dominant effect for the energy loss of light particles at high momenta [8].
Additional interactions with the nuclei of atoms can lead to showers in the material, particularly common in dense media. This process is exploited to detect particles with electromagnetic (for charged particles and photons) and hadronic calorimeters (for charged and neutral particles), by measuring the particles total energy deposition. Since these detection systems require dense materials and are typically of low spatial resolution, they are only used at large radii and do not influence detectors closer to the primary vertex. Such detection systems will not be considered in this thesis.

1.1.2 Neutron capture process

The physics program of HEP experiments mainly focuses on the detection of tracks from primary and secondary particles from particle decays. One effect which is not of interest for the current physics program of ALICE, but creates a significant contribution to the datasize and number of tracks, are charged particles emerging from secondary electromagnetic processes, following the deexcitation of nuclear states. These are electrons emerging from photon conversions ($\gamma \rightarrow e^+e^-$) released from excited nuclear states after the absorption of a primary neutron into nuclei of the detector material. The released electrons are typically of very low momentum ($p \sim O(10\ \mathrm{MeV}/c)$). The entire process is referred to as the neutron capture process.

To detect the released charges in drift and pixel detectors relevant for this thesis, external electric fields apply a Coulomb force on the released charge carriers, guiding them towards a sensitive readout. On the sensitive readout the particles induce a voltage difference or current, leading to a charge measurement. The final goal of every detector is the measurement of current / voltage changes, that are converted into digital values and used for further computations. The three interactions (ionization, Coulomb scattering and Bremsstrahlung) discussed above are effects exploited to measure incident particles and are the most common electromagnetic interactions encountered in high-energy particle physics experiments. While mainly ionization is exploited in innermost detector layers, all three processes contribute to the final momentum distribution of particles and are a key part of consideration for the optimization strategy of the individual detectors described in the following.

1.1.3 ITS - Inner Tracking System

The Inner Tracking System (ITS) is the first detector in radial direction from the interaction point (IP) [9]. Its primary task is the separation of primary and secondary (from particle decays) vertices as well as tracking, both necessitating good spatial resolution. Seven layers of silicon pixel sensors surround the IP in a cylindrical arrangement. In Run 3 of the LHC, monolithic active pixel sensors (MAPS) with a granularity of 30×30 μm^2 are used. The resulting resolution of $\sim 5\mu$m allows precise particle tracking and primary vertex resolution, better than 100μm. The silicon strip and silicon drift layers employed throughout Run 2 were replaced by pure silicon pixel layers for Run 3. This removes the possibility for particle identification through specific energy loss in the material but increases the momentum and angular track resolution [10]. Particle identification is then performed by the detectors following outwards in radial direction, most notably the ALICE TPC. Figure (1.1.2) shows the ITS as it is operated in ALICE throughout Run 3 of the LHC.

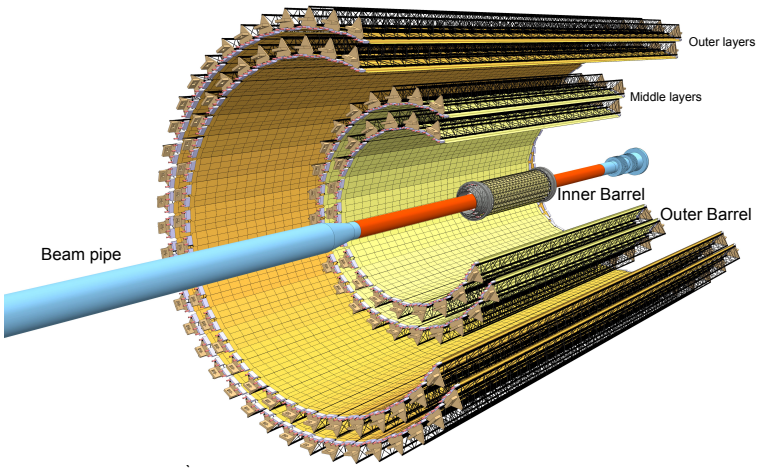


Fig. 1.1.2: Illustration of the ITS detector as it is used in ALICE throughout Run 3 [9].

1.1.4 TPC - Time Projection Chamber

The Time Projection Chamber (TPC) is the major particle identification detector of the ALICE experiment [11]. It is cylindrically arranged around the interaction point and located outside the ITS. With 5 meters in length, an inner radius of 83 cm and an outer radius of 2.5 meters, the ALICE TPC is the largest TPC ever built with a gas volume of 88 m^3. Its main purposes are particle tracking and identification. An applied electric field of 100 kilovolts (kV) between the central electrode and the anode plane accelerates the released electrons towards a sensitive readout at the end caps. After a drift time of $\approx$ 100μs (for electrons released close to the central electrode) the charges reach an amplification stage. For Run 3 of the LHC, the TPC readout was upgraded with Gas Electron Multipliers (GEMs) to amplify the arriving charges, resulting in a readout interval of 200 ns (50 kHz) (compared to 1 kHz in Run 2 with Multi-Wire Proportional Chambers) [12]. A schematic of the ALICE TPC is shown in figure (1.1.3).

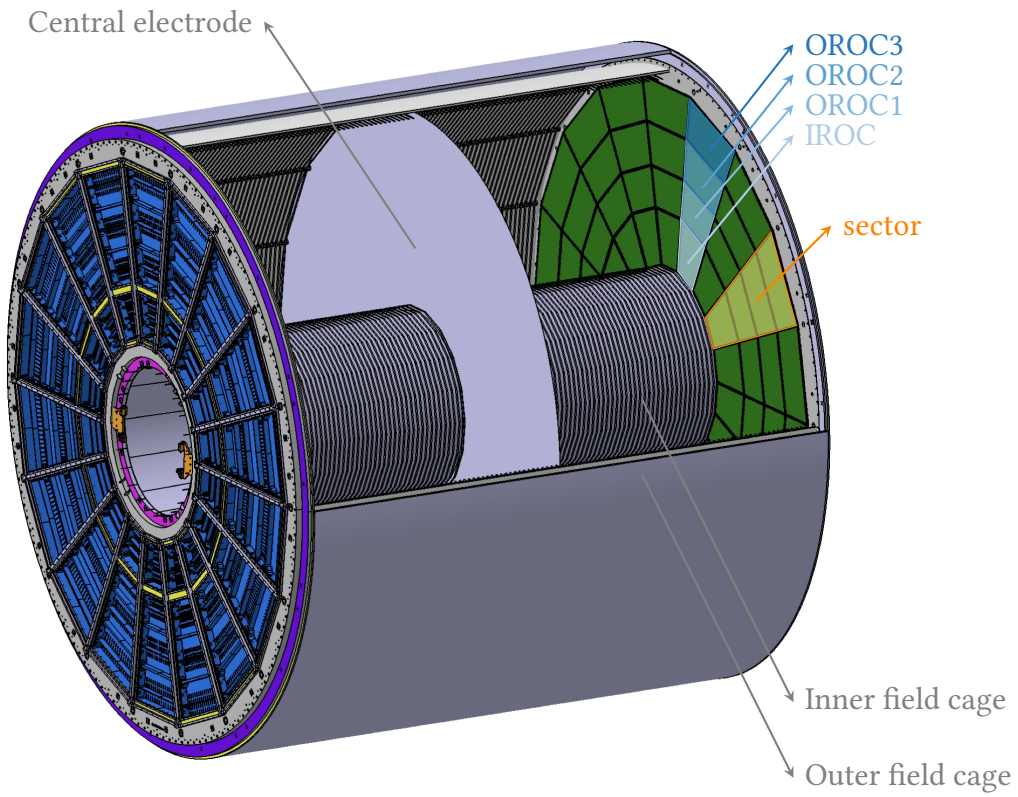


Fig. 1.1.3: Schematic of the ALICE Time Projection Chamber with highlighted central, high-voltage electrode, readout sector, inner and outer readout chambers (IROC, OROCs) and field cage [12].

Coordinate system, pad-plane and GEM stacks

The two half-sides, divided by the central electrode, are cylindrically arranged around the beam pipe and the interaction point at $|z| = 250$ cm. The end caps of each side are referred to as the pad plane. Each plane is divided into 18 sectors, each sector into 152 rows and each row

into $O(100)$ individual readout pads. The number of pads and their size varies over different rows. A pad represents the smallest readout unit of the TPC, commonly referred to as a voxel. The global and local x and y coordinates are illustrated in figure (1.1.4).

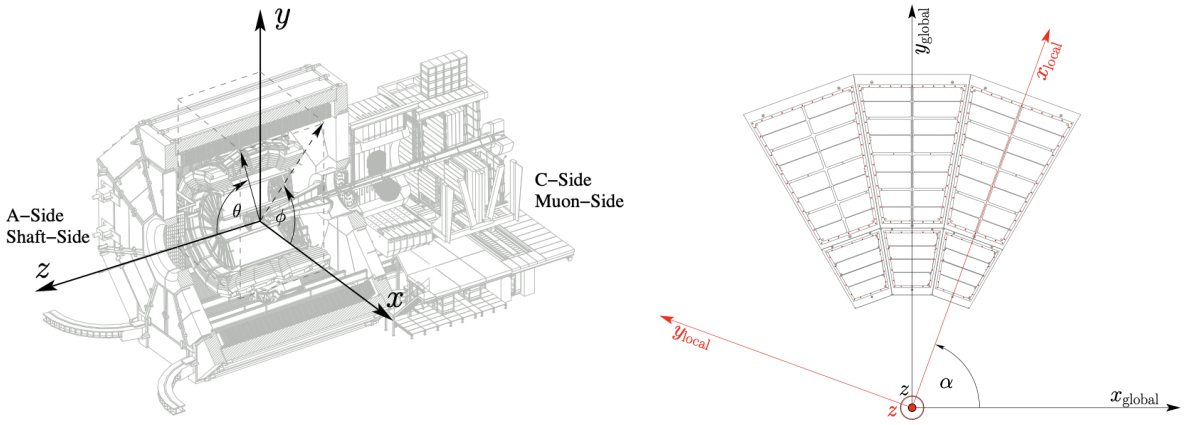


Fig. 1.1.4: The ALICE coordinate system (left: global, right: local) [12].

A direct, linear transformation between the (x, y) and (sector, row, pad) coordinate system exists, which will be frequently used in this thesis. Arriving charges are measured with their geometric position in the xy plane and their time of measurement. The pad geometry of the OROCs is shown in figure (1.1.5).

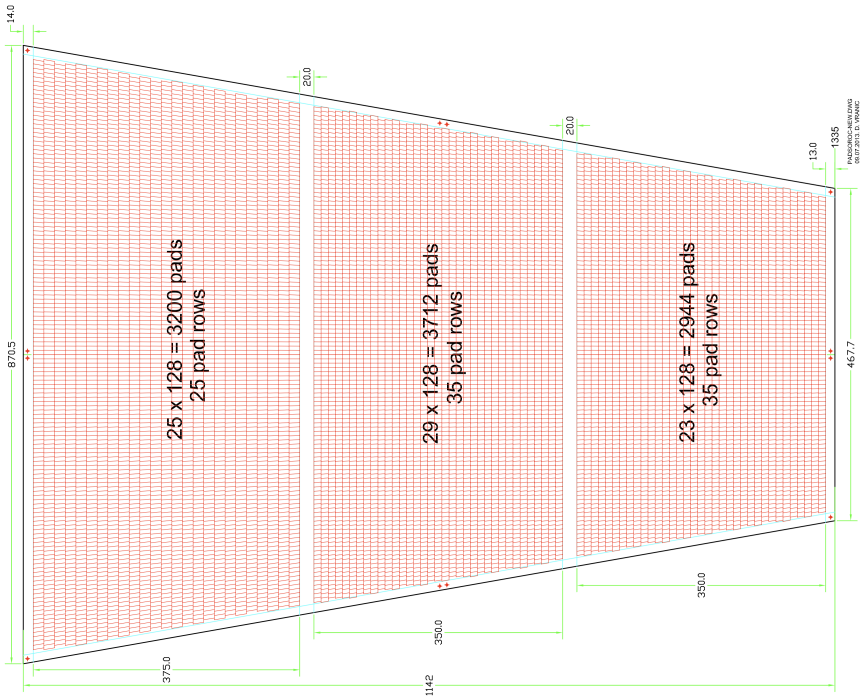


Fig. 1.1.5: Geometric orientation of the voxels of the outer readout chambers (OROCs), rotated counter-clockwise by 90° [12].

Prior to their arrival at the readout pads, the arriving charges are amplified in four GEM foils. Each foil is made of 50μm thin Polymide foil with a $2-5\mu$m thin copper-coated surface [12]. Symmetrically arranged, double-conical holes of 50μm in (inner) diameter are imprinted on the GEM foil. Electric fields of $O(50)$ kV/cm are applied to the GEM foils to accelerate arriving charges passing through the holes. This acceleration causes avalanches and charge amplification (gain) factors of $10^3 - 10^4$ [12]. The hole orientation, spacing of the GEM foils and applied electric fields follows the principles of maximizing charge amplification with minimal ion backflow ($O(1\%)$). The GEM stacks and the charge amplification process are illustrated in (1.1.6).

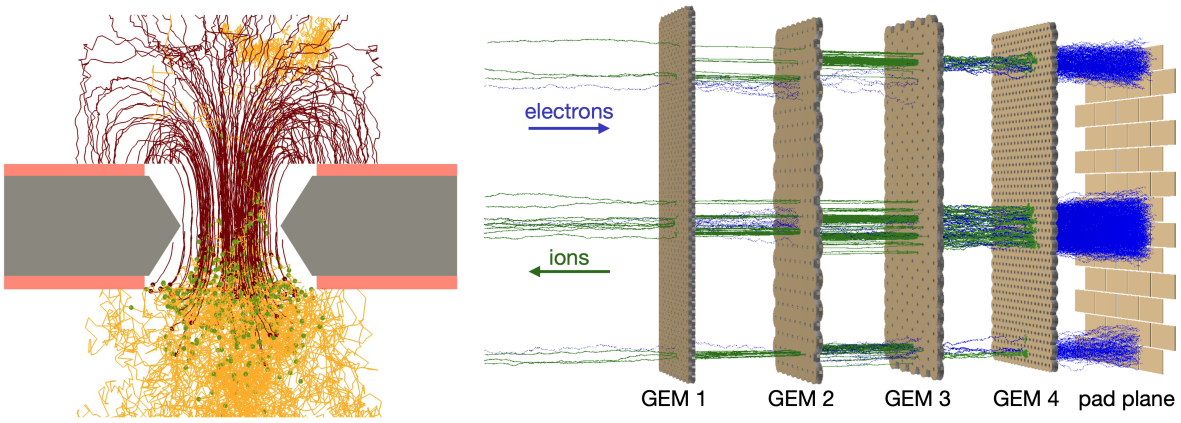


Fig. 1.1.6: Charge amplification of individual electrons at a single GEM hole (left) [12] and charge amplification over the four GEM stacks at the readout plane (right) [13].

Particle identification and the Bethe-Bloch formula

In the TPC, particles are identified by their specific energy loss per unit distance $\mathrm{d}E/\mathrm{d}x$. The energy loss of the particle is proportional to the deposited charge in the detector material. This makes the $\langle \mathrm{d}E/\mathrm{d}x \rangle$ for a given track a directly measurable quantity using the measured charges and track length across a padrow. A theoretical prediction for this quantity for a given material, characterized by its electron density n and mean excitation energy I, is given by the Bethe-Bloch formula [3], [14], [15]

$$\left\langle -\frac{dE}{dx} \right\rangle = \frac{4\pi n z^2}{m_e c^2 \beta^2} \cdot \left(\frac{e^2}{4\pi\epsilon_0} \right)^2 \cdot \left(\frac{1}{2} \ln \left(\frac{2m_e c^2}{I} \beta^2 \gamma^2 \right) - \beta^2 - \frac{\delta(\beta\gamma)}{2} \right) \tag{1.1.1}$$

for a particle with charge z, velocity v (and the Lorentz factors $\beta = v/c$, $\gamma = 1/\sqrt{1-(v/c)^2}$ and the relation $p = m\beta\gamma$). This function only depends on material properties and $\beta\gamma$ of the particle track. While material properties are studied in advance in a laboratory framework, the kinetic property $\beta\gamma$ needs to be determined for each track passing the detector gas. However, this quantity is not experimentally accessible. The measurement is restricted to the momentum p, where $\beta\gamma = p/m$. In collider experiments, an applied magnetic field, with field lines

parallel to the beam axis, curves charged particle tracks. Using the Lorentz force equation, the transverse momentum is extracted from the curvature radius of the track. Inclination of the track then allows the momentum determination. The particle identity is then assigned by matching the $\mathrm{d}E/\mathrm{d}x$ and momentum to the prediction made by the Bethe-Bloch function.

Formula (1.1.1) is derived for particles with masses much higher than the electron mass. Several overlapping physical effects create the characteristic shape of the Bethe-Bloch curve. For a particle with mass m at low velocities, longer interaction times with atomic electrons and larger momentum transfers govern the energy loss, which decreases with $1/\beta^2$. With increasing velocities, the electric field of the traversing particle is subject to Lorentz contraction in the direction of flight and an extension in transverse direction. This results in electromagnetic interactions with distant molecules. A relativistic treatment with the assumption $2\gamma m_e/m \ll 1$ leads to the term proportional to $\ln(\beta^2\gamma^2)$. This term is referred to as the relativistic rise and leads to an increase in energy loss with increasing velocity of the particle. However, at even higher velocities, polarization of the medium leads to a partial shielding of the electromagnetic field. This density effect is described by

$$\delta \rightarrow 2\ln(\hbar\omega_p/I) + 2\ln(\beta\gamma) - 1 \tag{1.1.2}$$

with $\hbar\omega_p = \sqrt{\rho\langle Z/A\rangle} \cdot 28.816$ eV the plasma energy of the medium [8]. With a maximum possible energy transfer for a particle with mass m to a shell electron of mass m_e ($m \gg m_e$) of

$$W_{\max} = \frac{2m_e c^2\beta^2\gamma^2}{1 + 2\gamma m_e/M + (m_e/M)^2}, \tag{1.1.3}$$

the Bethe-Bloch function approaches $\ln(W_{\max})$ rather than $\ln(W_{\max}\beta^2\gamma^2)$. This results in a complete cancellation of the relativistic rise due to the density effect and an asymptotic behavior for the Bethe-Bloch function at high $\beta\gamma$ [3].

In practice, the calculation of the $\mathrm{d}E/\mathrm{d}x$ value for a given track is done by dividing the track-attached cluster charges by the measured track length during the reconstruction process. However, the individual $\mathrm{d}E/\mathrm{d}x$ values along the particle trajectory are not Gaussian distributed, but rather follow a Landau distribution [16]. This results in non-finite mean and higher order moments of the final $\mathrm{d}E/\mathrm{d}x$ distribution per track. Using a Gaussian approach for a direct calculation of the mean value is therefore unsuitable, as it leads to large fluctuations due to the asymmetry of the distribution. To mitigate this problem a *truncated mean* calculation is performed, where only the lowest 50-70% of the measured $\mathrm{d}E/\mathrm{d}x$ values of a track are used. The fractions of contributing $\mathrm{d}E/\mathrm{d}x$ measurements to a track are calculated for maximum separation power between the minimum ionizing region and the Fermi plateau. The

resulting distribution follows a Gaussian distribution to good approximation, which allows the calculation of a mean $\mathrm{d}E/\mathrm{d}x$ value as

$$\langle dE/dx \rangle \coloneqq \frac{1}{\alpha N} \sum_{i=1}^{\lceil \alpha N \rceil} \left(\frac{Q_{\mathrm{tot},i}}{\Delta x_i} \right) \tag{1.1.4}$$

where $Q_{\mathrm{tot},i}$ is the total charge of cluster i along the track and Δx_i the distance to the next cluster. This value is assigned to the track and used in all further downstream processing. Due to this truncated mean estimation, the theoretical Bethe-Bloch function can not be used in practice. A parameterization of the Bethe-Bloch formula is fitted to the resulting $(p, \mathrm{d}E/\mathrm{d}x)$ distributions as will be described in more detail in chapter (IV). The resulting spectra with the fitted function are shown in figure (1.1.7).

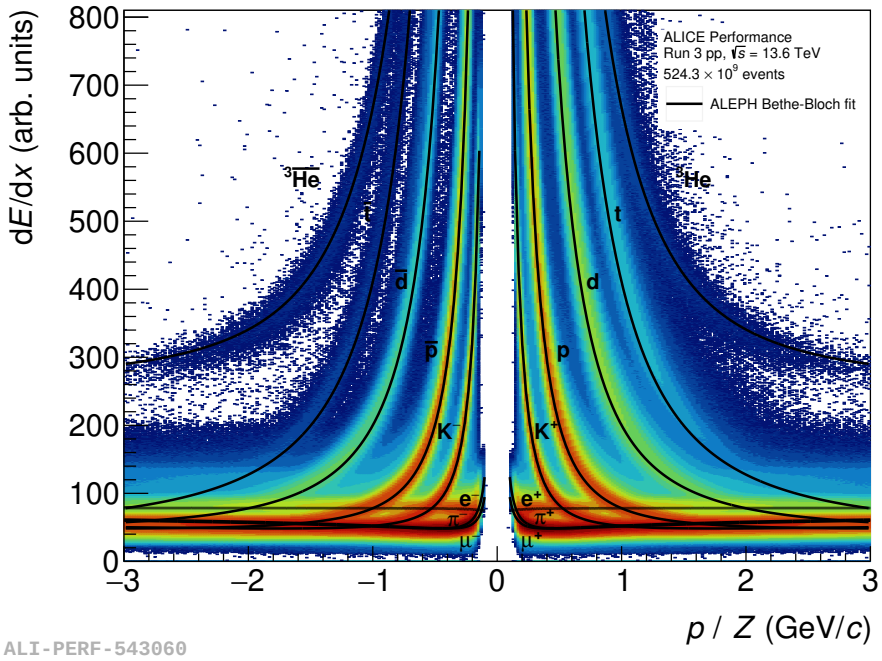


Fig. 1.1.7: $\mathrm{d}E/\mathrm{d}x$ spectrum resulting from all merged 2022 proton-proton high interaction rate runs with fitted Bethe-Bloch ALEPH parametrizations and a custom fit for ^{3}He.

Space-charge distortion effects

With an upgrade of the readout system and charge amplification stage in Run 3, the ALICE TPC can now operate at 50 kHz readout rate. Compared to Run 2, this is a 50 times higher time granularity and increases the data volume of the detector 50-fold. One full drift time of a released electron from the central barrel to the sensitive readout is on the order of O(90 μs) (compared to the 50 kHz $\rightarrow$ 200 ns readout intervals). This leads to $O(10^4)$ pile-up events in one TPC drift volume. Overlap-effects within this drift time can influence the time and position measurement of a drifting charge. As electrons are released from the detector gas,

positively charged (primary) ions are created and are accelerated in the opposite direction. Electromagnetic interactions with other drifting charges, such back-drifting ions cause positional distortions (typically called space-charge or space-point distortions). Positive (primary) ions released in the detector gas only account for a small fraction of the distortion effects. Major effects are caused by (secondary) ions released at the readout plane. With an amplification (gain) factor of approximately 2000 [12] and an ion backflow from the GEM stacks of approximately 1%, the contribution of positive ions from the GEM amplification surpasses the primary ion release by a factor 20. Due to the drift time (for the length of one half side of the TPC) of ions being $O(200\ \text{ms})$ compared to the drift time of electrons, the drifting ions can be imagined as disks moving through the gaseous volume.

The density of such charges and the resulting distortion effects increase with track occupancy. As the track density of primary tracks decreases linearly in radial direction, the space charge distortions are at least time- and radius-dependent. Additional effects of ϕ dependence and magnetic field inhomogeneities are typically accounted for within the space-charge correction framework. Both the drift velocity and distortion calibration are therefore essential to transform the time measurement of arriving charges at the readout plane to the z-position in the TPC. A data-driven approach, using extrapolated tracks from other detector systems that are not affected by such distortions (such as TOF [17] and TRD [18], outward in radial direction and ITS inward in radial direction) is used to determine cluster to track residuals and determine the space-charge distortion effects on a per-voxel level [13], [19].

TPC online reconstruction in Run 3

A major challenge during Run 3 is the processing of data in real time [20]. With 3.5 TB/s, the TPC is by far the largest data producer in the raw data stream ($> 99\%$) of the ALICE experiment. Since no hardware triggers are applied in ALICE throughout Run 3, the full readout must be processed and compressed. The online processing is split into two separate stages, the First-Level Processor (FLP) farm and the Event Processing Nodes (EPN).
Major tasks of the FLP farm concern raw data processing and compression, while the EPN farm performs reconstruction and calibration tasks. Each FLP is dedicated to a part of a single detector and only receives the localized information from this part, global information is not present at this stage. Each FLP builds a so-called subtimeframe (STF) containing the local information necessary for downstream, global event reconstruction. The compression achieves a reduction in transferred data size to approximately 900 GB/s at a nominal rate of 50 kHz, minimum bias, Pb–Pb interactions. The information from all FLPs included in a data-taking run are then gathered within one EPN for global event reconstruction. Load balancing is applied using a data distribution system to utilize the available servers / resources. Due to the high parallelism of several components of the reconstruction software, the event processing nodes

were upgraded with graphics processing units (GPU) for Run 3. Two server configurations are used in the computing farm at the time of writing

- 8x AMD MI50 hardware accelerators (GPU) + 4x AMD EPYC 7452 32-Core Processor + 512 GB memory
- 8x AMD MI100 hardware accelerators (GPU) + 4x AMD EPYC 7552 48-Core Processor + 1024 GB memory

While HEP experiments generally rely on double-precision floating point arithmetic (double = 64-bit floating point values), the ALICE software is optimized to operate using single-precision floating point (float = 32-bit floating point) operations. Lower precision (e.g. half precision, float16) is insufficient in accuracy for the track reconstruction process, while single precision results in a beneficiary trade-off between compute speed and accuracy. An overview of the computationally most demanding tasks on the EPNs is seen in figure (1.2.11).

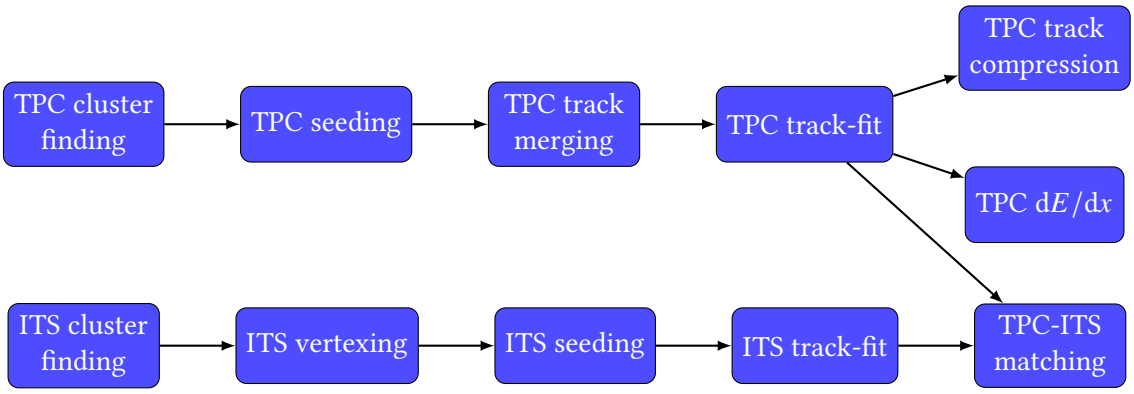


Fig. 1.1.8: Major data processing steps for ITS and TPC processing.

In Run 3, the triggerless readout under which the TPC is operated requires a matching with other detection systems, such as the ITS, as no trigger time information is present. Matching tracks over multiple collisions and assigning them to primary vertices is therefore one of the key challenges of the Run 3 setup for ALICE.

The final object containing all low-level, global information, assembled on the EPNs is called a timeframe (TF). It contains all corresponding timestamps and metadata information after the first-level compressions and is a standalone unit from which the reconstruction framework commences. This allows opportunistic compute-load distribution at online processing time and independent re-reconstruction of TFs at later stages. The TF length during Run 3 is set to be 32 LHC orbits or $\Delta t_{\mathrm{TF}} \approx 2.8$ ms. While longer timeframes require less frequent distribution and result in lower data losses, the chosen value is a trade-off between compute margins and compression factors experienced within Run 3 operations. After the reconstruction process, most of the raw data is discarded, with the most demanding contribution stemming from

TPC clusters and tracks. It is infeasible to store the entire raw detector readout, however for postmortem investigations, a fraction of approximately 0.1% of the raw data is stored. In all other cases, the cluster data is stored into a compressed timeframe (CTF) format.

1.1.5 TPC readout and cluster-finding

Drift charges amplified in the TPC GEM stacks are measured with a sensitive readout ASIC (application-specific integrated circuit) called SAMPA [21]. The SAMPA chip sends the signal to the CRUs (Common Readout Unit). The charges are converted via an ADC (analog-to-digital) converter to a digital signal which is used for computations. The units in which the computations on charges are done are hence called *ADC counts*, which are binned charges with intervals of 200 ns (50 kHz). The charge value of a single readout pad for one readout time interval is called a *digit*. Longitudinal diffusion D_L and track inclination $\tan(\lambda)$ lead to shape modifications of the arriving charge clusters. The expected cluster width in z direction in units of time is described by

$$\sigma^2_{\text{time}} = \frac{1}{v_{\text{d}}^2}\left(D^2_{\text{L}} z_{\text{d}} + \frac{\tan^2(\lambda) L^2_{\text{pad}}}{12}\right) + \sigma^2_{\text{PASA}} \tag{1.1.5}$$

with σ_{PASA} being the semi-Gaussian signal output of the PASA (pre-amplifier/shaper) on the SAMPA chip and z_{d} (v_{d}) the drift length (velocity) [12]. Charge diffusion leads to a more elongated, falling tail of a cluster. Even isolated clusters are therefore intrinsically asymmetric in time. An example of a single TPC row readout for a single event simulation at 50 kHz Pb–Pb interactions is shown in figure (1.1.9).

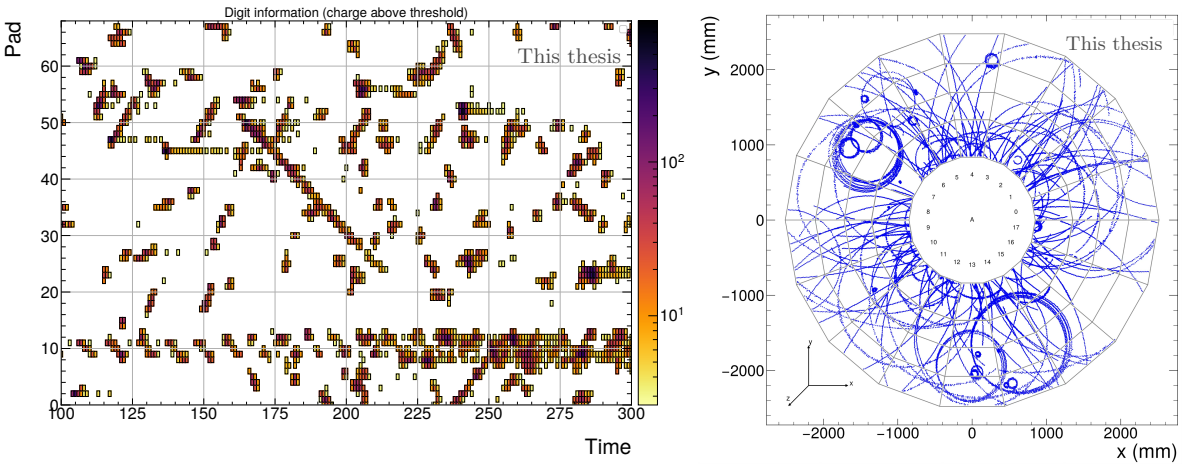


Fig. 1.1.9: Left: TPC row filled with charges after charge amplification and signal shaping. The color illustrates the digit charge value. Right: Resulting TPC clusters in the global xy coordinate system integrated over 100 time bins.

A cluster is characterized by four essential properties

- **Cluster center:** Arrival time and geometric (pad) position of the weighted charges arriving at the front-end readout
- **Cluster width:** An estimate of the width of a cluster in both pad and time direction
- **Total charge:** The integrated signal amplitude of the arriving charges
- **Maximum charge:** The charge of the local maximum around which a cluster is built

The current reconstruction algorithm determines the cluster components in two dimensions: pad and time. Clusters are reconstructed on a per-row basis. Since the reconstruction framework parallelizes the cluster finding step on a per-sector level, the sector information is implicitly provided. Each local charge maximum is considered as a cluster candidate. No intrinsic determination is made for the asymmetry of a cluster. Furthermore the best possible approximation for the cluster position is considered to be the cluster center of gravity. Similarly, the currently utilized estimator for the charge entering the $\mathrm{d}E/\mathrm{d}x$ computation is the total charge. Clusters are then reconstructed using a local window of charges in pad and time direction, from which the properties are inferred. A detailed description of the determination of cluster properties from these local charge values will be illustrated in the next chapter (II). The final cluster object is then used to perform track reconstruction.

1.1.6 Tracking and downstream processing

The essential element from which a physical meaning is inferred is a track. The task of track reconstruction is to connect clusters and create a track object. The approach for the ALICE tracking is divided into different phases. Firstly, track seeding connects 5 to $N_{\mathrm{rows}}/2$ nearby clusters, depending on availability in adjacent rows using a cellular automaton approach [22]. Seed extrapolation using the Kalman filter approach [23], [24], outwards and inwards in radial direction leads to additional cluster attachments. Track refitting in the inward and outward direction of the seed extrapolation and interpolation between the results refines the cluster attachments and marks a first step of cluster rejection [25]. Three iterations of the inward-outward-inward refit process are performed [26], [27]. Additional iterations are conducted for the track extrapolation and slice refitting. Since the cluster density decreases towards outer radii, the process is initiated with an inward pass over the available data. The final track object consists of 5 parameters

- y - local y position of the track
- z - local z position of the track
- $\sin(\phi)$ - track inclination in the local xy-plane, measured to x_{local}
- $\tan(\lambda)$ - track inclination in the global xz-plane
- q/p_T - charge to transverse momentum ratio

For physics analysis the inclination is frequently exchanged with the pseudorapidity $\eta = -\ln\tan(\frac{\pi}{4} - \frac{\lambda}{2})$. From the final track object and the attached clusters, physics properties are inferred [28]. Examples of such properties are the track momentum (p), the inclination angles (ϕ, λ) and the energy loss per unit distance $\mathrm{d}E/\mathrm{d}x$. Since the charge released by a particle is directly proportional to the loss of energy in the active detector material (TPC gas), the $\mathrm{d}E/\mathrm{d}x$ is calculated from the total charge of an attached cluster and the track length over the pad row. This quantity and hence the total charge per cluster is a crucial measurement for the physics performance of the TPC.

Space-charge distortions are the most dominant, adverse effect encountered when operating a TPC at such high rates. During the tracking process in online reconstruction, a precise knowledge of such distortions is however not available. A simplified version of scaling default distortion map objects, using the current instantaneous luminosity is used to enable a first-level correction. This is necessary for cluster-to-track associations in the TPC, track matching between detectors and data compression. The uncorrected cluster positions can experience distortion effects up to $O(20\ \mathrm{cm})$ in radial direction. Together with a calibration of the drift velocity, a final cluster position determination of the z-position within the detector volume is made.

With the triggerless readout, data losses at the (time-)edges of a timeframe cannot be avoided. Clusters too close to the boundary are considered for seeding and tracking, but discarded in the physics analysis due to incomplete data coverage. The total volume of lost data is calculated by

$$\text{loss}\ [\%] = \frac{\text{TPC electron drift time}}{\text{TF length}}. \tag{1.1.6}$$

For a timeframe of length 2.8 ms (reconstruction standard at the time of writing) and a drift time of 90 μs, the final data loss will therefore be $\approx 3.1\%$. All reconstructed clusters are then written into CTF files. Final calibrations performed on cluster level during the asynchronous reconstruction process will then lead to further improvements in tracking efficiencies to other detector systems [20]. Both track-model compression via entropy encoding [29] as well as cluster rejection of very low momentum particles ($p < 0.01\ \mathrm{GeV}/c$) (and therefore data reduction) are the main arguments why online tracking is necessary. The final calibration, reconstruction and physics quality measurements are however extracted in offline, asynchronous reconstructions of the data.

1.2 Neural networks and hardware acceleration

1.2.1 Hardware acceleration - CPU vs. GPU

With increasingly challenging environments encountered in Run 3 and beyond, analysis and reconstruction strategies have to adapt to the expected conditions. Hardware accelerators, such as GPUs (graphics processing unit) and FPGAs (field programmable gate arrays), have already made a great impact in first-level triggering and online reconstruction of the major LHC experiments. Their main advantage over purely CPU (central processing unit) based computations is the inherent ability for parallelization and concurrency of computations. A CPU processor typically consists of O(10) (consumer grade) to O(100) (server grade) individual cores, where a core is a hardware execution unit, which computes sets of instructions. A sequence of instructions is commonly called a thread and does not represent a physical object, but rather a sequence of logic operations. The ability to run several threads in parallel on a single CPU is called multi-threading, while the execution of multiple threads on a single core is called hyper-threading[1]. Both techniques are found in modern CPUs and allow for optimized hardware utilization. Additionally to the actual compute unit(s) of a core (ALU = Arithmetic Logic Unit, performs bitwise, logical and integer operations; FPU = Floating Point Unit, performs floating point arithmetic), on-chip memory allows fast data access and control units steer the incoming instruction sets and data accesses. The defining characteristic of a CPU in comparison to a GPU is the fast execution per core, but comparably low number of cores. Typical execution speeds of modern CPUs can reach up to $\sim$ 5 GHz, which refers to the number of CPU cycles. A cycle here defines the atomic unit of instruction processing: Instructions like addition and multiplication each require few cycles per evaluation (depending on parallelism of operations and availability of registers). The specific use case for a CPU is optimized for fast sequential execution of instructions on low levels of parallelism. An example of such a task is array sorting or binary searches.

In contrast, a GPU is designed for parallel executions. A computational (hardware) unit in a GPU is referred to as a *thread*. Up to 1024 threads are organized into a thread block, where the exact number of threads per block depends on the hardware chip. Each thread block is divided into *warps*, which represent groups of (typically) 32 neighboring threads, consecutive by their assigned index. The GPU executes all threads in a warp in lock-step fed by a single instruction decoder, such that all threads must always execute the same instruction or no instruction. Data transfer and scheduling overheads are thus minimized. This concept is commonly referred to as the SIMT paradigm (single instruction, multiple threads). The instruction that all threads in a process grid execute is called a *kernel*. Multiple thread blocks are grouped into

[1] This is Intel-specific naming. Other companies like AMD or IBM have other names for it (e.g. AMD: Simultaneous Multi-Threading (SMT))

a streaming multiprocessor (SM) and multiple streaming multiprocessors constitute the full GPU. A schematic is shown in figure (1.2.10).

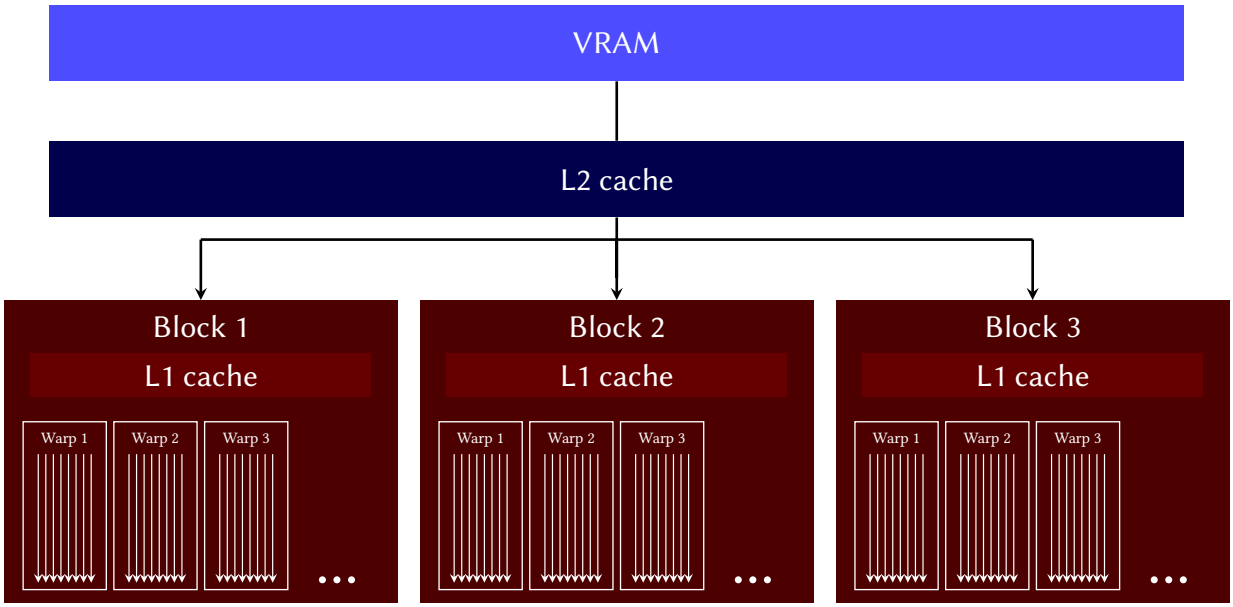


Fig. 1.2.10: Schematic of a GPU layout. Illustrated are the L1 and L2 cache, thread blocks, warps and individual threads.

Unless explicitly programmed, individual threads in a warp do not transfer information among each other, as this would result in serialization of instructions. For such a crosstalk, the respective data element needs to be written to the L1 / L2 cache memory. L1 cache is block-local and comparably small (O(100) KB) but optimized for fast read/write accesses (O(100) GB/s). It is used to store data for intermediate computations during the execution of a thread block. On the other hand, the L2 cache provides a larger memory array, is accessed on the SM level but typically provides slower read/write speeds than the L1 cache. Global operations that require communication of threads over multiple blocks can therefore be a limiting factor for GPU execution speeds. This limits the efficient applicability of GPUs to tasks with sequential aspects. For consecutive sets of instructions, threads sometimes need to be synchronized. All threads within a warp are synchronized by construction and operate in lockstep. Threads in different warps on the same block need to be synchronized explicitly (barrier synchronization). This can be necessary for sequential tasks, which require the result of a previous computation.

A heterogeneous system, is a system which uses different hardware architectures. The CPU is then typically referred to as the *host*. The dedicated hardware (in this thesis: GPUs) is referred to as the *device*. On systems with dedicated hardware, CPUs steer the on-device processes using specific device functions and calls. Furthermore, the operating system, memory operations and network data transfers require the use of CPUs, as GPUs are not designed to communi-

cate with these system components. Likewise, the host system can greatly benefit from the usage of a GPU, by offloading certain highly parallelizable processes to the dedicated hardware.

Due to the SIMT execution model, the design of GPU threads is greatly simplified in comparison to CPU threads. Individual thread-level scheduling and instruction distribution mechanisms are handled implicitly in hardware rather than explicitly managed per thread. GPU threads are therefore simplified execution units with lower sequential throughput rates compared to a CPU, but a significant increase in parallelism. Modern GPUs consist of O(10^4) to O(10^5) hardware threads (CUDA cores) and achieve computational speeds in the $O(10)$-$O(100)$ TFLOP/s range (Tera Floating Point Operations per second) for single-precision floating point values (single precision = float32, 32-bit representation). Dedicated hardware acceleration for half-precision floating point (float16) and tensor operations exists in modern GPUs and can further increase speed up for dedicated tasks. Examples of tasks which can heavily benefit from such cores include matrix-matrix and matrix-vector operations, commonly found in neural network computations.

1.2.2 Fully connected feed-forward networks - Introduction

Neural networks form the core of modern machine learning research. Their large scale applications in all areas of science and everyday life was made possible due to the onset of hardware acceleration, while the first theoretical concepts date back to the early 1940s [30], [31]. Core components of a neural network are the weights and activation functions, which form *neurons*, the atomic unit of computation of a neural network. For the simplest case of fully connected neural networks, a single neuron is represented by a non-linear activation function σ, a weight vector $\vec{w}$, a bias value b (scalar) and the vector of inputs $\vec{x}$:

$$\sigma(\vec{w} \cdot \vec{x} + b) = \sigma\left(\sum_{i=0}^{dim(\vec{w})} w_i x_i + b \right) \tag{1.2.1}$$

Multiple neurons are stacked into a layer. In most common constructions, all neurons in a layer use the same activation function, with a different set of weights per neuron. The bias value is absorbed into the weight vector by appending a "1" to the input and hence a layer at index ℓ with N inputs and M outputs can be written as

$$\vec{y}_\ell = \sigma\left([x_\ell^1, ..., x_\ell^N, 1] \cdot \begin{bmatrix} w_\ell^{11} & \dots & w_\ell^{M1} \\ \vdots & \ddots & \vdots \\ w_\ell^{1N} & \dots & w_\ell^{MN} \\ b_\ell^1 & \dots & b_\ell^M \end{bmatrix} \right) = \vec{x}_{(\ell+1)} \tag{1.2.2}$$

where the activation function σ is applied element-wise on the output vector. The weights and biases (w, b) represent the set of adjustable (learnable) parameters. A neural network is then formed by computing the final output y_L for a set of L layers using the above recursion relation, starting from x_0, the input list of values. Such networks are typically represented using circles for the nodes and arrows for each weight, connecting all nodes from layer ℓ to $\ell + 1$. This is shown in figure (1.2.11).

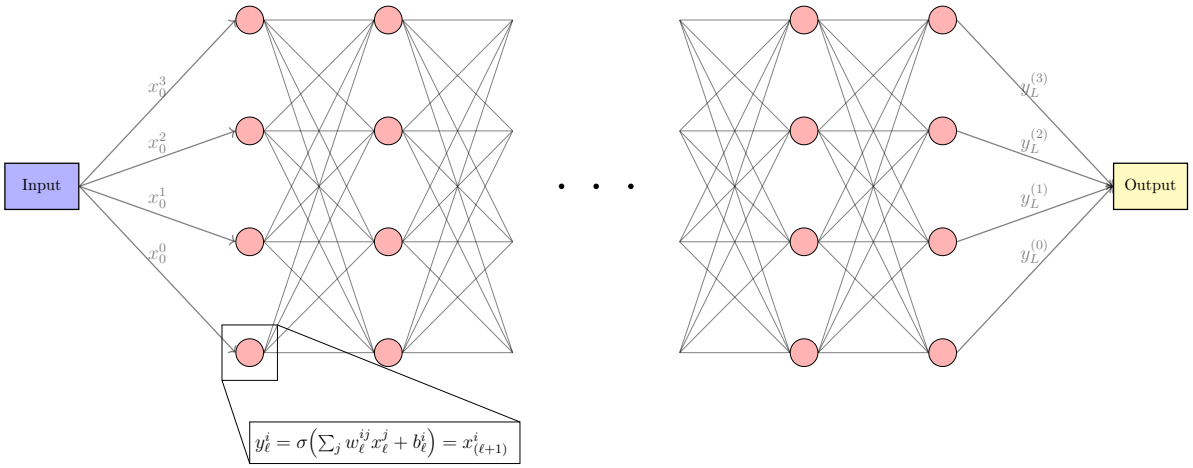


Fig. 1.2.11: Illustration of a fully connected network. Circles represent the nodes in which matrix-vector multiplication is performed, arrows represent the connections between layers (weights).

Classical neural networks fall under the category of supervised machine learning algorithms, where a set of training data is required on which the network is optimized. For neural networks this equates to the modification of the weight and bias values through backpropagation. Values, such as weights and biases, that are learned by backpropagation are called *parameters*, while all other characteristics of a neural network (e.g. activation functions, architecture, number of layers) that are not learnable from a gradient descent optimization, are called *hyperparameters*.

1.2.3 Backpropagation and gradient descent

The performance of a neural network on a set of training data is measured by the *loss function*. It calculates a single value, given a set of output values of a neural network computation and a set of target values, taken from the training data set. An example for such a function, that is typically used for regression tasks, is the mean-square-error loss (MSE)

$$\mathrm{MSE}(y, \hat{y}) = \sum_{i \in M_L} (y_L^i - \hat{y}_L^i)^2 \tag{1.2.3}$$

where y is the set of ground truth values, taken from the training data set and $\hat{y}$ the set of values predicted by the neural network for data point with index i and input x_i to the network. Ground truth refers to a set of data for which the desired output values of the network are known (e.g. simulated data). As seen in the previous subsection, the structure of a simple fully connected neural network is a rather simple concatenation of functions. The activation functions are chosen to be differentiable to the weight values in each layer, which makes the entire neural network differentiable (layer-wise). By choosing the loss function in a differentiable way to the weights of the last layer, the gradients for each weight in the network are explicitly calculated [32]. An algorithm that optimizes the weights in such a way, that the loss function is minimized is called *gradient descent*. Together with a learning rate, the gradients are used to find local minima in the high-dimensional loss surface. Since every weight of the neural network is a free parameter, the dimensionality of the optimization is $O(\#\text{weights})$. An update to the weights of the neural network is performed after the calculation is done on a subset of the data, a so-called *batch*. The correction applied to the weights depends on the calculated gradient and is scaled with the *learning rate*. Many different variations of gradient descent exist. For convex loss surfaces, the optimization is guaranteed to converge to the global minimum within distance ϵ in a finite number of optimization steps. Loss surfaces for real-world applications almost never obey the property of convexity, however it is an experimental observation that the number of local minima with high loss values decrease exponentially with increasing dimensionality [33]. A more critical problem in this regard is the convergence of optimization algorithms on local saddle-points with high loss values. This is solved using saddle-free Newton methods [33], [34] and stochastic gradient descent (SGD) algorithms [35], [36] which have become the baseline for optimization of large neural networks. SGD-based optimizers rely on computing gradients on randomly sampled subsets of the training dataset, largely avoiding convergence on saddle-points and have since surpassed many other algorithmic approaches in performance. They are commonly used to train even billion-parameter models. One common algorithm which will also be used in this thesis is the Adam optimizer [37].

1.2.4 Convolutional neural networks

The specific choice of architecture and optimization strategy is dependent on the problem at hand. While simple feed-forward neural networks have the advantage of flexibility of the input and output dimensions and generalization to many problems, their performance often falls short of dedicated architectures for specific tasks. Image recognition is one such challenge in which convolutional neural networks (CNNs) have outperformed many other architecture [38], [39]. Based on the principle of convolution, a layer in a CNN represents a filter operation on a local set of inputs. For a one-dimensional function $f(t)$, the convolution operation is defined as

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau) \cdot g(t - \tau) d\tau \tag{1.2.4}$$

where $g(t)$ represents a filter function. For CNNs, this process is implemented by multiplying a window of values of an input image (e.g. $n \times m$ with n, m typically 3 or 5) with a matrix G ($G \in \mathbb{R}^{m \times n}$) of weights, learned by gradient descent [38]. A convolutional layer is then the multiplication of G on all $n \times m$ windows of the input image. Optionally, striding and shifting can be applied to adjust the patches of data seen by the neural network. This process can be extended to three (and higher) dimensions, depending on the dimensionality of the input data. Since the dimensionality of G is typically small and all input windows are multiplied with the same weight matrix, CNNs are well parallelizable on GPUs. An example of the application of a 3D CNN is shown in figure (1.2.12).

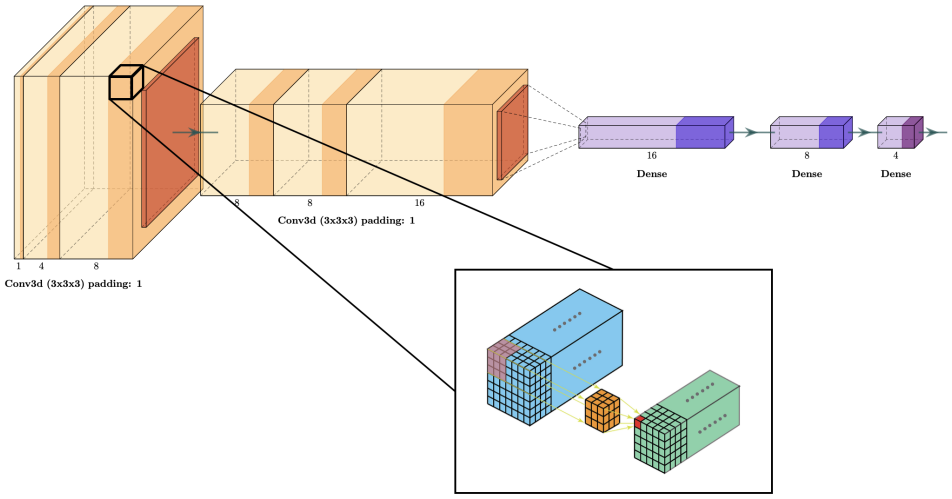


Fig. 1.2.12: Illustration of a CNN with three-dimensional convolution layers.

Both fully connected and convolutional neural networks will be investigated in this thesis. Dedicated studies are done to evaluate the performance of such network architectures in both their fit accuracy as well as their compute capabilities.

1.2.5 Open problems in the field of machine learning

Recent developments in the field of machine learning have produced many other architectures for the layers of a neural network. Many of them require more complex and compute-intensive transformations of the data (normalizing flows) or significantly more operations (attention mechanism, transformer architecture). Many open questions still exist in the field of machine learning. Examples of these questions are

1. **What is the best network architecture / choice of layers and hyperparameters for a given problem?**

2. **What is the most compute-efficient architecture as a trade-off to precision?**

3. **Are there macroscopic behaviors of neural networks that could be described by a fundamental theory from first principles (analogous to statistical mechanics)?**

The first two topics are usually addressed in an empiric approach with measurements and (in many cases) hand-tuning of the architecture. The optimization of the architecture is however rarely as critical as in tasks that apply neural networks in a compute-bound environment. This is one of the core guiding principles for this thesis. The third question is significantly more fundamental. While it exceeds the scope of this thesis, the observed scaling law behaviors in large language models hint at the existence of more global laws governing the field [40]. These laws have yet to be derived from first principles. Machine learning at the time of writing is therefore experiencing a similar evolution and curiosity as quantum mechanics in its early stages, over a century ago. It is with this curiosity that this thesis aims to deliver insights to the applicability of neural networks, while opening the possibility to a new era of machine learning on large scales at the ALICE experiment.

II Algorithmic approaches to cluster finding

2.1 Monte Carlo cluster finding

Essential to this study is the comparison and evaluation on Monte Carlo (MC) data. Both the creation of training data and the evaluation of the models will be performed on MC data. This approach relies on the assumption that track topologies with their charge releases and cluster shapes are accurately modeled. With MC labels at hand, ideal clusters are built (where MC labels are unique identifiers, connecting a measured charge to a simulated particle). Since this thesis focuses on the development of a new cluster finding algorithm, the design choices for building clusters and their usage in tracking closely follow the current implementation within the reconstruction framework. The following types of charges / clusters will be considered in this thesis

- **Digit** - One digit is the charge in one voxel and has a unique coordinate, specified by (sector, row, pad, time). The set of all digits represent the raw detector readout. Charges of multiple tracks can contribute to a single digit. In the case of simulated data, a digit can have contributions from multiple MC labels, weighted by their charge contribution.
- **Digit maximum** - Local charge maximum of the digits in a given row. Synonymously, the word "peak" is used.
- **Reconstructed clusters** - Clusters which are found by a cluster finding algorithm on a per-row basis and will be used in the downstream processes (e.g. track reconstruction).
- **Ideal clusters** - Clusters which are constructed on simulated data, accumulating charges of the same MC label. This algorithm is novel in this thesis and not part of the standard software.

Ideal clusters are essential for this work but are not available in the official ALICE software framework. The following section will therefore deal with the construction of such clusters and their usage and comparison to the reconstructed clusters.

The accumulation of charges to build ideal clusters is done on a per-row basis. This allows to compare them directly to the reconstructed clusters and disentangles the cluster center-of-gravity from the row direction. Several cases need to be taken into account to build ideal clusters at simulation time:

- **Clusters with overlap** - Charges of multiple tracks that overlap and cannot be trivially separated without MC labels
- **Looping tracks** - Low momentum tracks passing through a pad-row more than once by looping in the external magnetic field. Charges will create multiple clusters and all of them are given the same MC label (as they originate from the same MC particle).

Charges of a propagated MC particle are simulated at the readout plane and arrive at different voxels over the runtime of the simulation. Therefore, and due to the cases mentioned above, it is not straight forward to add up charges of the same MC label directly. In pad direction, the shape is mainly determined by charge diffusion, while in time direction both charge diffusion and the signal shaping of the readout influence the cluster shape. The charge signal of one cluster will therefore be limited to a Gaussian distribution in pad direction (one or few pads in width) and a semi-Gaussian shape in time direction, as described in chapter (I). The time window of one cluster can range from $O(1)$ to $O(100)$ time-bins. The final digit output is then constructed as an overlap between charges of one or multiple MC tracks and detector noise. Therefore, a dynamic, on-the-fly algorithm for calculating the cluster position and width is applied, as the memory usage would otherwise vastly exceed available compute resources. Welfords online algorithm is used to calculate the cluster properties [41]. This algorithm computes the sample mean and variance iteratively and with a single pass over the dataset, making it both compute-time and memory efficient. For the mean of the sample $\bar{x}_n = \frac{1}{n}\sum_{i=1}^{n} x_i$ and $s_n^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x}_n)^2$ the unbiased sample variance over the first n data points in the unordered set (i.e. $(x_1, \ldots, x_n) \subseteq X$) is calculated iteratively using

$$\bar{x}_n = \frac{(n-1)\bar{x}_{n-1} + x_n}{n} = \bar{x}_{n-1} + \frac{x_n - \bar{x}_{n-1}}{n} \tag{2.1.1}$$

$$\sigma_n^2 = \frac{(n-1)\sigma_{n-1}^2 + (x_n - \bar{x}_{n-1})(x_n - \bar{x}_n)}{n} = \sigma_{n-1}^2 + \frac{(x_n - \bar{x}_{n-1})(x_n - \bar{x}_n) - \sigma_{n-1}^2}{n} \tag{2.1.2}$$

$$s_n^2 = \frac{n-2}{n-1}s_{n-1}^2 + \frac{(x_n - \bar{x}_{n-1})^2}{n} = s_{n-1}^2 + \frac{(x_n - \bar{x}_{n-1})^2}{n} - \frac{s_{n-1}^2}{n-1}, \quad n > 1 \tag{2.1.3}$$

Numerical instabilities due to ever decreasing correction values to the mean with increasing sample size are resolved by using

$$s_n = s_{n-1} + (x_n - \bar{x}_{n-1})(x_n - \bar{x}_n) \tag{2.1.4}$$

$$\sigma_n^2 = \frac{s_n}{n} \tag{2.1.5}$$

$$s_n^2 = \frac{s_n}{n-1} \tag{2.1.6}$$

A weighted version of this algorithm exists, which can take individual sample weights into account [42]. The algorithm is written in pseudo-code as

Initialize: $W \leftarrow 0$, $\bar{p} \leftarrow 0$, $\bar{t} \leftarrow 0$, $s_{\text{pad}} \leftarrow 0$, $s_{\text{time}} \leftarrow 0$

foreach (w_n, x_n) *in incoming_charge_positions* **do**

$W' \leftarrow W + w_n$

$\delta_p \leftarrow x_n^{(\text{pad})} - \bar{p}$

$\delta_t \leftarrow x_n^{(\text{time})} - \bar{t}$

$\bar{p}' \leftarrow \bar{p} + \frac{w_n}{W'}\delta_p$

$\bar{t}' \leftarrow \bar{t} + \frac{w_n}{W'}\delta_t$

$s_{\text{pad}} \leftarrow s_{\text{pad}} + w_n\,\delta_p\,(x_n^{(\text{pad})} - \bar{p}')$

$s_{\text{time}} \leftarrow s_{\text{time}} + w_n\,\delta_t\,(x_n^{(\text{time})} - \bar{t}')$

$W \leftarrow W', \bar{p} \leftarrow \bar{p}', \bar{t} \leftarrow \bar{t}'$

The position of an arriving signal is therefore weighted with the charge value in the given pad-time bin. At simulation time, a map of MC labels with all above-mentioned variables is created per row and sector and the algorithm is evaluated using the stored and incoming values. The problem of overlapping charges within a cluster is naturally taken care by only considering charges of the same MC label. A new cluster is created in case the arriving charge has an MC label different from all MC labels in the map. In case of matching MC labels, only charges falling within a configurable pad-time window around the current center of gravity are considered to contribute to an existing cluster. If a new charge arrives, for which the MC label already exists in the map, but which falls outside the chosen window around the current center of gravity of the existing cluster, a new cluster with that MC label is added to the map. Figure (2.1.1) shows the accumulated charges per MC label, together with matched and unmatched digit maxima and ideal clusters.

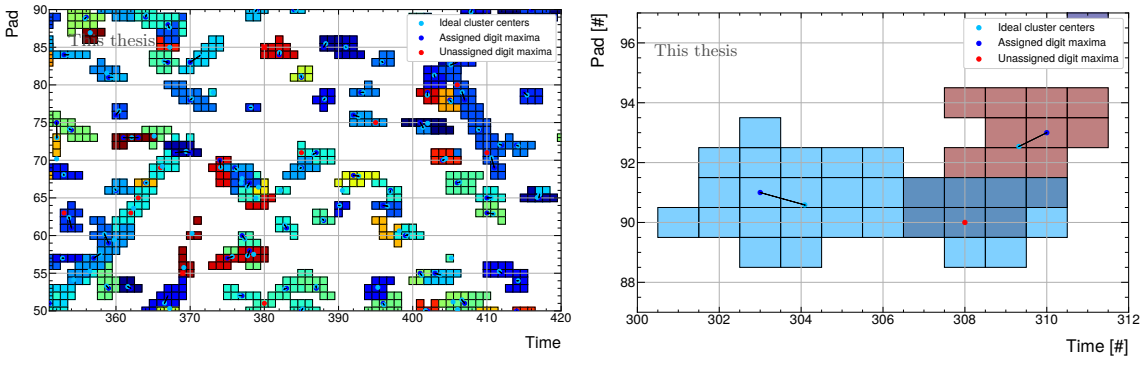


Fig. 2.1.1: Digit maxima and matched ideal cluster positions for a large pad-time window (left) and illustration of local cluster charge overlap of multiple MC labels (right). Each color represents a different track MC label.

Considering efficiencies and the generation of training data, clusters are not created for noise charges. However, their released charge will contribute to the digit charges. This algorithm

provides a robust way of building ideal MC clusters. The problem of cluster from looping tracks polluting the environment remains and is addressed in the following.

In the following, investigations are performed in two steps. The first is the creation of ideal clusters at simulation time, the second is the matching of such ideal clusters to reconstructed cluster or digit maxima in a separate post-processing step, independent of the simulation framework. Later on in the analysis, this will allow the computation of cluster matching efficiencies and fake-rates.

2.1.1 Looping tracks and occupancy tagging

Looping tracks (commonly abbreviated as loopers) are typically low-momentum electrons or hadrons (predominantly protons) crossing a single row more than once. As they pass through the detector material they approximately follow a helical path, looping multiple times until they are absorbed in the material. Such particles leave large charge remnants and can create a high number of clusters that are not considered useful for track reconstruction or physics analysis ($\approx 15-30\%$ in 50 kHz Pb–Pb interactions). By radial extent of the TPC ($r_{\text{max}} = 250$ cm) and setup of the experiment ($B_{\perp}^{\text{nom.}} = 0.5$ T), only tracks with a maximum transverse momentum can create looping tracks in the sensitive volume of the TPC. It is calculated to first approximation by equating the Lorentz force to the centripetal force of the particle.

$$\begin{aligned} \vec{F}_z = \frac{m \cdot v_{\perp}^2}{r} &= q \cdot \vec{v} \times \vec{B} = \vec{F}_{\text{B}} \\ p_{\text{T}} = rqB_{\perp} &\overset{B_{\perp}^{\text{nom.}}}{\Rightarrow} p_{\text{T}}^{\text{max,looper}} = 187.4 \text{ MeV/c} \end{aligned} \tag{2.1.7}$$

This calculation ignores material budget, energy losses along the track path and assumes looping tracks with $q = 1e$ emerging from the primary collision point, where the particle remains fully within the boundaries of the TPC (track radius ≤ 1.25 m). For more realistic purposes, the particle can reenter the TPC volume even after interactions with detectors further away in radial direction (TRD). Accounting for material interactions, the maximum transverse momentum of looping tracks with a charge value of $1e$ is $p_{\text{T}} \approx 300$ MeV/c.

As the looping particle passes through the detector multiple times, the track model is insufficient to describe pad-parallel parts of looping tracks. Furthermore, at the track merging stage, many such clusters cannot be attached to tracks due to their large charge spread and diffused center-of-gravity position. For this reason, tracking is interrupted at local inclination angles higher than $\alpha > 70^{\circ}$ with respect to the sector-local x-coordinate (see introduction, figure (1.1.4)). Clusters of looping particles which would require the track inclination to go beyond this value are thus not used in the track fit. In light of this consideration, resulting clusters

at higher local inclination angles inflate the data volume, increase computation time (as the Kalman tracking scales with $O(N)$ for N clusters) and pollute the environment, reducing reconstruction and physics quality. This poses a significant challenge for ALICE during Run 3, as tracking performance strongly decreases with increasing local occupancies, particularly in the densest environments of central Pb–Pb interactions at high interaction rates.
For loopers with a small curvature radius, charge diffusion increases the cluster width to such an extent, that charges cannot be cleanly separated into individual clusters anymore. The charge of a given looping track, where the track has passed a row multiple times, appears as a semi-connected area in one row, with multiple digit maxima appearing across this region. Such digit maxima appear from the crossing points of the incident particle, as well as statistical charge fluctuations. With an average cluster width of approximately 0.4 cm (σ_{pad}), such charge overlaps occur outside the tracked looper legs for loopers with a minimum radius of 4.8 cm. Together with the condition of a minimum of 10 connectable clusters for a track and a maximum inclination angle of 70°, the minimum looper radius must be 5.9 cm or approximately $p_{\text{T}} = 8.9\ \text{MeV}/c$. Since track reconstruction discards all tracks with $p_{\text{T}} < 10\ \text{MeV}/c$, all looping tracks with at minimum 10 clusters per leg will be found. However, even the clusters of discarded tracks are kept, inflating the storage space. The extent to which a looping particle can be tracked also depends on the particle species, as low momentum proton loopers will lose more energy per unit distance ($\text{d}E/\text{d}x \sim 1/\beta^2$) than electrons ($\text{d}E/\text{d}x$ is closer to minimum ionizing region).
The distinction between a looper charge-maximum and a regular track cluster overlapping in such a region is hardly possible from the two-dimensional information in any given row, when using a small receptive field of view. Clusters of looping tracks at high inclination angle and highly ionizing particles, like low momentum hadrons can create near identical charge remnants when considering the two-dimensional charge distributions in the pad-time plane.

It is necessary to exclude clusters of looping particles with larger than 70° inclination from the following calculations of attachment efficiencies (attachment of ideal clusters to digit maxima). While this condition cannot be enforced directly, as the inclination depends on the track fit and is not available for clusters without track attachment, an occupancy tagger was built to mitigate this problem. It operates based on the presence of MC clusters with the same MC label in a local pad-time vicinity. If more than a (configurable) number of ideal clusters with the same MC label are found in a (configurable) pad and time window, the area around the MC center-of-gravity position of all found clusters with that MC label is excluded from the calculation of efficiencies and fake-rates (including all digit charge maxima found in this region). An illustration is given in figure (2.1.2) where the high inclination part of a looping track is seen. Illustrated in pink is the excluded area from the occupancy tagger.

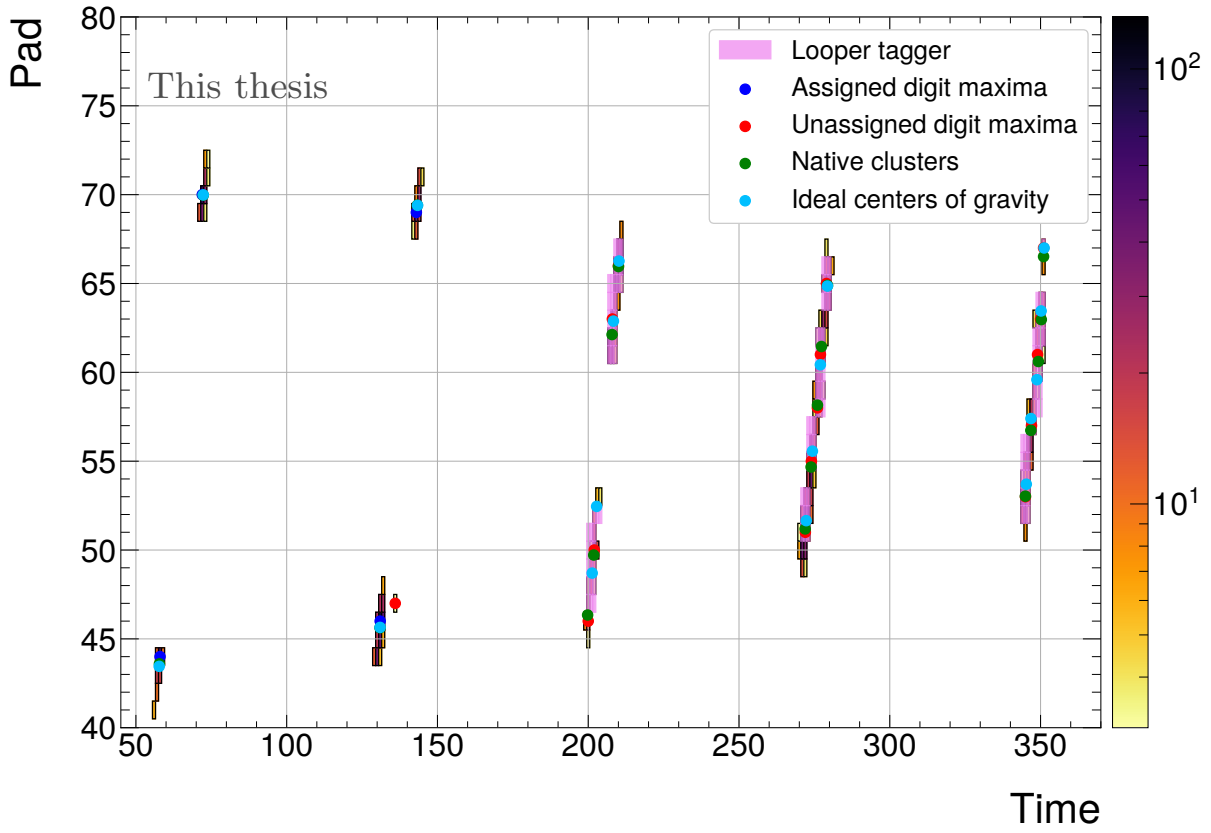


Fig. 2.1.2: Illustration of the looper tagging algorithm on a looper track passing a single row (pad and time in bins of the readout), the charge information is color-coded.

It is important to note that certain parts of a looper track, where the curvature radius is large, are locally indistinguishable from regular tracks, especially at low inclination angles. Wide looping tracks can still create several individual tracks, which are also found as *looping legs* by the tracking algorithm. Only at high inclination should such clusters be rejected. Several settings for occupancy tagging were tested. It is also necessary to distinguish between clusters belonging to looping tracks and large charge areas, released by highly ionizing particles (e.g. low momentum hadrons) which should not be rejected. An occupancy tagger with the following settings was used for the investigations in this thesis: 4 MC clusters with the same MC label in one row and total charge values greater than 70 units each, found in a sliding window of 20 (pad window) and 40 (time window) bins with a striding of 10 units in time. Since surrounding areas of each cluster need to be investigated, the algorithm is constructed in a sliding-window fashion which significantly reduces calculations. Testing for the choice of settings was conducted over multiple simulations and iterations, where the behavior on individual loopers and highly ionizing particles was investigated. Requiring more than 4 MC clusters for rejection in the local region can lead to false acceptance of tight looping tracks. Requiring less MC clusters or reducing the window size can however lead to significant reductions in acceptance of highly ionizing particles.

For each identified MC looping cluster, all digit maxima and MC clusters within the 1σ range

around the center of gravity in both pad and time direction are excluded. This significantly reduces incorrect exclusions of clusters and charge maxima of tracks with different MC labels, that cross the same row within the current sliding window.

2.1.2 Assignment strategy and ambiguity resolution

For the assessment of cluster finding quality and the generation of training data for the neural network, an assignment between the calculated clusters and the digit maxima must be performed (see figure (2.1.1)). The assignment is done by proximity between the ideal cluster and each digit maximum. It is noteworthy that a charge maximum in an ideal cluster does not necessarily result in a digit maximum, as the maximum in the ideal cluster is calculated for one MC label only. Multiple overlapping ideal clusters can therefore lead to the obstruction of individual ideal maxima.
Two strategies can be taken: the assignment of digit maxima to the charge maxima of the ideal cluster or the (rounded) ideal center-of-gravity position. In practice, no significant difference was found between both strategies in terms of assignment rates. For further studies, the second strategy (matching with the center of gravity) is chosen. To compare this with the alternative (neural-network-based) algorithmic approaches to cluster reconstruction presented in this thesis, the center of gravity will be the common trait of a cluster before and after the reconstruction process and application of tracking algorithms, as associated digit maxima are not permanently stored.

Algorithmically, the matching between digit maxima and centers of gravity is done by rounding the center-of-gravity position to the nearest integer value and matching it to a neighboring digit maximum. This process can result in matching ambiguities. An example would be an overlap of three ideal clusters that create three different centers of gravity, but less than three digit maxima, see figure (2.1.3).

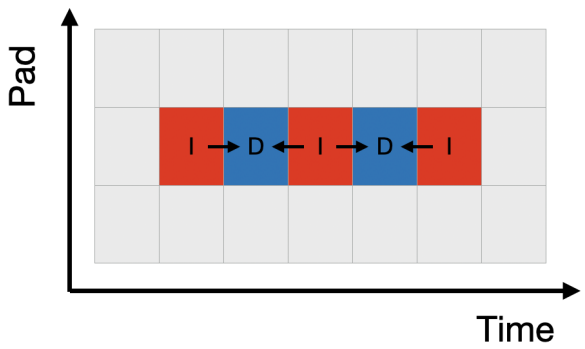


Fig. 2.1.3: Ambiguity resolution for multiple matching ideal clusters. (I, red) = position of the rounded center of gravity of an ideal cluster, (D, blue) = position of a digit maximum

In this case, the ambiguity is resolved by assigning the central ideal cluster to the digit maximum which is closest (Euclidean metric, not Manhattan metric). Ideal clusters, where the center-of-gravity positions overlap within the same pad-time bin are considered indistinguishable. The ideal cluster with the higher total charge is then kept for matching. A merging between such overlapping clusters is not performed, otherwise all evaluations will suffer from incorrect center-of-gravity positions and total charges. A [pad,time] window for the charge accumulation per cluster at simulation time of [2,4] ([3,8]) results in 1.5% (3.2%) overlapping clusters with the same rounded center-of-gravity value of all ideal clusters with 91% (98%) of those having different MC labels between conflicting clusters.

Three important quantities are derived from the assignments:

- **Efficiency**: Percentage of ideal (MC) clusters that were matched to a digit maximum, normalized to the total number of ideal clusters.
- **Clone rate**: Percentage of ideal clusters to which more than one digit maximum is attached, normalized to the number of ideal clusters.
- **Fake rate**: Percentage of digit maxima that do not have a matched ideal cluster, normalized to the total number of digit maxima.

The clone rate is calculated fractionally: For each ideal cluster (indexed with i), count the number of attached digit maxima (n_i). For each digit maximum with more than one attached ideal cluster, calculate $\frac{1}{d}\sum_{i=0}^{d} 1/n_i$ (where d is the number of attachments). These quantities give a direct measurement to assess the quality of the assignment. The assignment itself is performed in multiple iterations. At each iteration the distance from the current center (digit maximum) is increased, see figure (2.1.4, left).
All digit maxima and ideal clusters assigned in any given iteration are marked and not reassigned in later iterations, which avoids conflicts in highly populated areas. An exception to this rule is made for digit maxima in iteration steps 1 and 2. Even if an ideal cluster was assigned to a digit maximum, the digit maximum is not yet marked in iteration 1, in order to allow multiple ideal clusters in the immediate vicinity to be assigned to it, if no other digit maximum is present, see figure (2.1.4, right). In case a given ideal cluster is assignable to two digit maxima which both have additional attachments, the closest cluster (by floating point accuracy of the center of gravity) is chosen to avoid double counting.

2.1.3 Performance of the MC clustering algorithm

Given the previous design choices and degrees of freedom, the goal is to optimize the efficiency while minimizing the fake rate using the adjustable pad-time window of the charge accumulation algorithm (Welfords algorithm) at simulation time. For a given MC label, the window setting defines the maximum distance (in pad and time direction) around the current

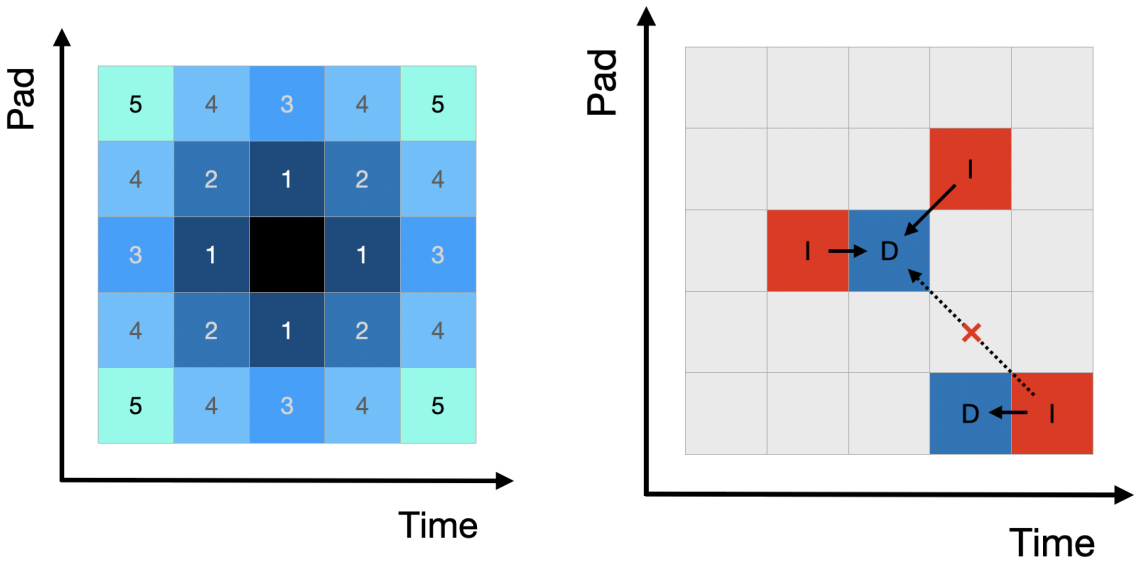


Fig. 2.1.4: Left: Illustration of the assignment. The number increases with distance to the center and represents the iteration step at which a given cell is considered for assignment. Right: Assignment of ideal clusters in iteration steps 1 and 2 to a digit maximum.

center of gravity at which a charge with the same MC label still contributes to an ideal cluster. The performance of the algorithm is also inspected visually: Obvious errors, such as clearly separable digit maxima with only one found ideal cluster in between them are avoided. The following observations (table (2.1.1)) were made

(pad, time) window	efficiency	fake rate
2, 2	76.2%	7.2%
2, 4	86.6%	14.6%
3, 5	92.0%	20.0%
3, 6	92.5%	20.8%
4, 6	91.2%	21.2%
5, 6	93.7%	24.0%
5, 7	94.3%	25.2%

Table 2.1.1: Efficiency and fake-rate for different settings of the ideal cluster finder algorithm. The evaluation was performed on 10 minimum bias events at 50 kHz Pb-Pb interaction rate.

This investigation was performed on 10 simulated, minimum bias, 50 kHz, Pb–Pb events in which space-charge distortion effects and detector noise were disabled, to disentangle the matching performance from detector effects. Clone rates are typically kept below 1% due to the matching capabilities of the algorithm and the design choice to make clusters with the same rounded center-of-gravity position indistinguishable.

The decrease in fake rates for smaller window sizes is explained by the occupancy tagging algorithm. Due to the strong increase in number of total clusters as the pad-time window of ideal clusters decreases in size, more and more regions (even of regular tracks) are tagged and rejected. This ultimately leads to false splitting of clusters at simulation time and a rejection of clusters, which are relevant for the final physics performance during the post-processing step. Therefore, the fake rate decreases with larger pad-time window sizes. In light of total charge accumulation, the smallest investigated sizes ([2,2], [2,4]) give inaccurate results for the total charge estimation of larger clusters. Therefore, a pad window size of 3 units and a time window size of 6 units was chosen for all further simulations.

2.2 The ALICE TPC online cluster finder algorithm

Clusters are reconstructed for downstream tracking within the online reconstruction of ALICE. This cluster finding algorithm is used throughout Run 3 data taking in the GPU-based reconstruction software. It is a heuristic algorithm designed for maximum compute efficiency on GPUs. The following section contains a detailed description of its working principles and the proposed, improved solution using neural networks.

2.2.1 Working principles of the heuristic cluster finder

The varying conditions under which the TPC operates require a robust and parallelizable clustering algorithm. The ALICE TPC cluster finder does not distinguish between different beam settings, occupancies, pad-sizes or detector conditions. It is a heuristic algorithm that is applied in all circumstances and data taking conditions.

The first step for this clustering algorithm consists of finding local charge maxima around which clusters are built. The heuristic cluster finder considers a peak according to figure (2.2.5). A peak is accepted if and only if all conditions are met. Furthermore, the peak is kept if a peak charge of greater than 3 is found, otherwise the peak is discarded.
The digit charge values after signal shaping are stored as single-precision floating point values. A maximum value of 1024 is allowed for the charge per pad-time unit in every row. The pattern for peak finding was chosen to reduce the number of reconstructed clusters that need to be built for regions achieving charge saturation (e.g. highly ionizing particles). If multiple neighbors have the same charge value, only the one with the highest pad and highest time value is accepted. The choice for this pattern is a trade-off between computing speed and physics performance. No further computation is invested to find a more centralized peak for a given cluster with saturated digit charges. Due to the asymmetric signal shape in time direction and slower falling edge of a cluster, the chosen peak is a better approximation of the

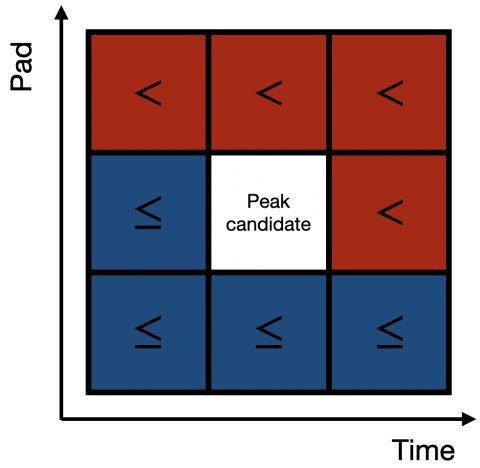


Fig. 2.2.5: Peak finding according to the ALICE heuristic cluster finding algorithm. The signs indicate that the neighboring charge in the cell needs to be $<$ or $\leq$ than the peak candidate in the center.

cluster center than the peak with the smallest time value. To reduce detector noise and further decrease combinatorial calculations single-pad-single-time peaks (i.e. peaks, where all nearest neighbor charges are 0) are rejected. Clusters are then built around all remaining peaks.

A deconvolution step is applied to account for overlapping charges. For each digit, the number of surrounding peaks (n) in the 5×5 neighborhood is counted. The digit-charge is afterwards divided by n and special charge flags are set for the surrounding peaks, which are used in the cluster error parametrization and cluster-to-track attachment downstream. To build a cluster from the neighboring charges, the 5×5 charge grid surrounding a given peak is used. All inner charges (peak and 8 neighbors) are summed. For the outer charges (16 charges surrounding the inner 3×3 grid), it is assessed if a charge is a local peak, in which case it is not considered for the computation of the current cluster. The cluster properties are then calculated from all charges and relative positions that contribute. Split-flags are attached to the cluster if the peak was flagged as "split" by the deconvolution algorithm. Finally, clusters with a total charge of less than 5 ADC counts are rejected to further decrease detector noise. Cluster objects are stored in a compressed format using the uint16 datatype for the charge values.

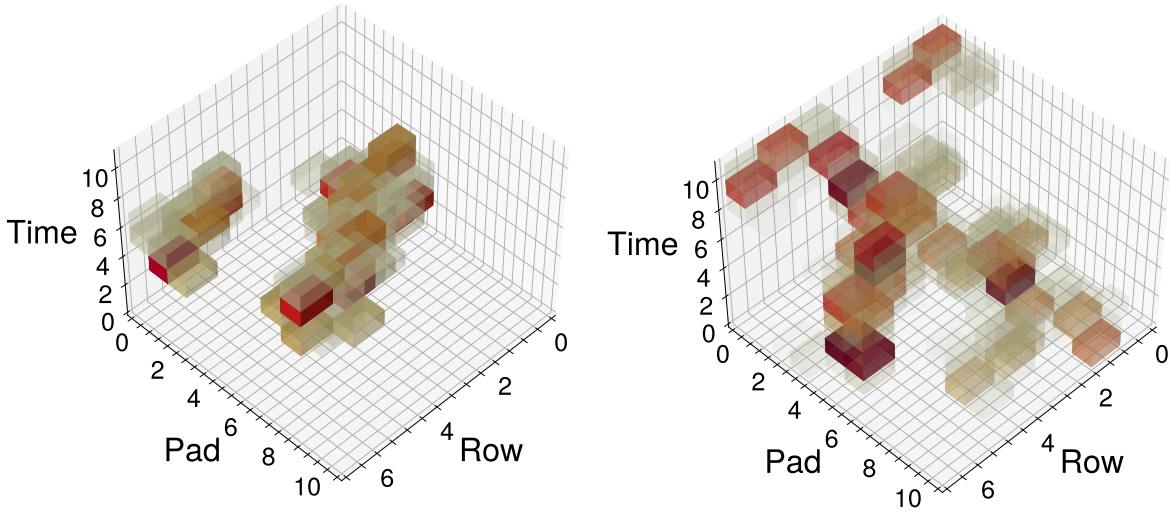
2.3 Neural network cluster finder

Due to the inherent rigidity of the heuristic cluster finder, no adaptations for different pad sizes, cluster sizes, interaction rates, occupancies or other detector effects are made. It is not a priori clear how these quantities could even be used in the context of rigid algorithmic approaches to this problem. However, these parameters are used or learned implicitly in the latent space of supervised machine learning algorithms such as neural networks.

2.3.1 Working principle and input data

A neural network is a mapping between a set of input values in $\mathbb{R}^I$ and a set of output values in $\mathbb{R}^O$. Both I and O are flexible and the parameters of the mapping are learned through a set of training data. In the case of cluster finding, the input will consist of (at least) local charge values surrounding a given peak. For simplicity and consistency with the current reconstruction processing, the current peak-finding algorithm is kept. An ordering of the input is expected and depends on the architecture of the network. In the simplest case of a fully connected NN, all values of interest are passed as a flat array to the input nodes. In case of CNNs, the input is reshaped as a two or three-dimensional image-like array over which the convolution kernels stride. All intermediate calculations are defined by the adapted (learned) weights and activations functions of the intermediate layers.

Two different types of tasks are investigated in this thesis: classification and regression. A classification network is trained to identify if a cluster should be created from a given peak position. After that a regression network calculates the cluster properties. As an input, the charges surrounding the peak position are used in a grid-like fashion. The peak of interest is always located in the center of the grid. The notation of the grid that will be used throughout this thesis are integers of the form (row width, pad width, time width), indicating the width of the grid in row, pad and time direction. As an example, the input grid for the current, heuristic cluster finding algorithm would be (1,5,5). For the neural network also three-dimensional input charge arrays will be considered. An example is presented in figure (2.3.6) where the charge of each digit is color-coded. Typical raw data contains more noise than the shown examples, but for the purpose of illustration a set of cleanly separable tracks was chosen.



(a) (b)

Fig. 2.3.6: Simulated charge inputs to the neural network (examples) with a size of (7,11,11) in both cases (larger size chosen for better visualization).

As pad sizes change over the rows of the TPC and sector boundaries are encountered by the algorithm, the sector, row and pad number are passed to the neural network in addition to the surrounding charge values, as they uniquely define the geometric position of a charge in the xy-plane. The final input size to the network will therefore be of size $\Delta R \times \Delta P \times \Delta T + 3$ (for a row width ΔR, pad width ΔP and time width ΔT of the input charge array). Normalization of input values is known to improve neural network performance, avoids value overflows at inference time and diverging gradients at training time. The sector value is therefore normalized to the maximum number of sectors (36), the row value to the maximum number of rows (152) and the pad value to the number of pads of the row containing the investigated digit maximum. For the charge array surrounding the peak, two studies were conducted in which the charges are normalized to the central digit charge under investigation, or to a constant value of 1024 (the maximum ADC charge value). The former method has the advantage that the input grid and resulting cluster properties are scale invariant to the maximum charge of a cluster. However, it also comes with the disadvantage that clusters of low total charge can have contributions of tracks releasing high charges within the input grid of the neural network. These secondary charges are fully disconnected from the current cluster but can distort the network result due to their potentially higher values. It is ensured that such cases are present in the training data, such that the network is trained on them, to mitigate this behavior. The second option is to normalize to a constant value (1024), however it was observed that particularly for small peak charges, the accuracy of the total charge estimate of a cluster by the network decreases drastically. Additionally, the problem of secondary, disconnected charge contributions remains (in a scaled version). Furthermore, for actual deployment in the online computing system only half-precision floating point neural networks (float16) come into question, for which the inaccuracy for low charge clusters normalized in this way is already fatal. As will be discussed in chapter (V) the floating point accuracy of a converted, half-precision neural network is found to be on the order of the fourth decimal place. For peak charges of e.g. 10 ADC counts and a normalization to 1024, the rounding error alone is within O(10%) of the total charge value, unacceptable for physics performance. Therefore, the first method, normalizing the input charge array to the central peak charge is chosen.

The outputs of the classification network are class labels which are intrinsically normalized by the last layer of the neural network using the sigmoid / softmax function.[1] The network will therefore return a floating point value (single or half precision depending on the application) between $[-\infty, \infty]$ where the acceptance threshold is inverse transformed ($\sigma^{-1}(\text{threshold})$). Avoiding the sigmoid computation within the network in this way is an additional design choice to reduce computational resource usage at inference time. Depending on the chosen

[1] To increase performance at inference time, the last layer has no activation function. The softmax computation is rather offloaded in the calculation of the loss function, see the PyTorch documentation: CrossEntropyLoss, BCEWithLogitsLoss.

threshold, the number of accepted clusters is varied. A visualization of the clusters with their class labels assigned by a trained classification neural network are shown in figure (2.3.7). This merely demonstrates how the neural network operates and what it is (in general) trained to reject. The outputs for the regression network will then be the cluster properties (center of gravity, width estimation and total charge) and depending on the investigated case also the momentum vector learned from the three-dimensional input charge array.

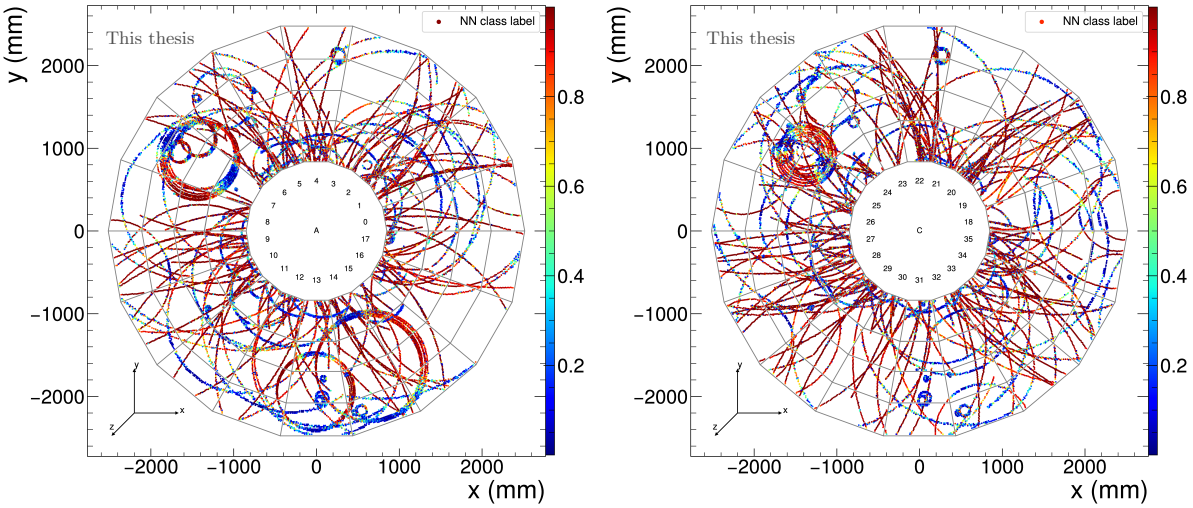


Fig. 2.3.7: Cluster labels assigned by the classification neural network after training (color-coded), as an illustration on few selected tracks (left: TPC A side, right: TPC C side). Low scores (blue) indicate clusters which will be successively rejected, while clusters with higher scores (red) remain, depending on the chosen threshold.

Flowchart of the processing chain

The application of the neural network and the implemented kernels, are illustrated in the following flow-chart (2.3.8). The names of the GPU kernels are descriptive of their task and are used in the same fashion in the O2 codebase. A detailed description of each kernel is provided in appendix (A.0.1). The evaluation is performed per sector and in case of multithreading (CPU) also in parallel per thread on a subset of the data. Names marked in brackets refer to the applied GPU functions. In case no brackets are present, the same function is used for the evaluation on CPU and GPU.

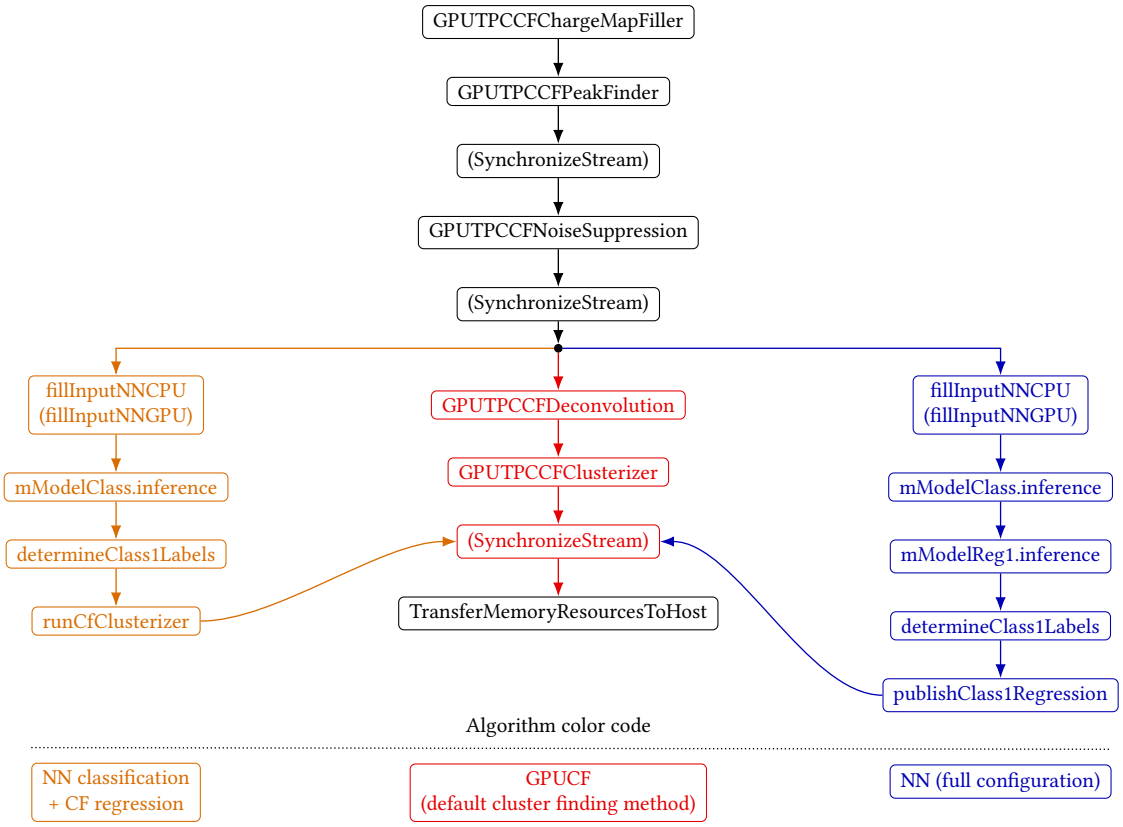


Fig. 2.3.8: Overview of the cluster-finding workflow showing the common GPU pre-processing stages (black), the GPUCF path (red), the NN classification with CF regression path (orange), and the full NN configuration (blue).

2.3.2 Training data and selection strategy

The training data for the neural network is created using the ideal clusters generated at simulation time and matched to the digit maxima in the post-processing step. For the task of classification, the number of ideal attachments for a given digit maximum are counted. If the cluster has an attachment, the class label of 1 is assigned, otherwise 0. The full training dataset for classification is then the set of digit maxima and their respective class labels. The network is trained on clusters with 0 and 1 labels as the ground truth. For the regression network, the set of all digit maxima with an attached cluster form the training data, where the cluster properties of the attached ideal clusters are used to train the final regression output of the network. For the position of the cluster (center-of-gravity), the network is trained to calculate the residual from the peak position in both pad and time direction. The width estimation is taken directly from the ideal cluster without modification. The total charge is learned as the ratio $q_{\mathrm{tot}}^{\mathrm{ideal}}/q_{\mathrm{max}}^{\mathrm{peak}}$. Additionally, the momentum vector of the reconstructed track is attached to the cluster in case an ideal cluster is matched. It is investigated in the next chapter (III) how the momentum vector can be learned by the network purely from the local digit information.

Crucial for the performance of neural networks is the training data selection. While the collected data might well cover the phase space, a sampling over it can ensure a more uniform distribution, leading to improved fitting results. The following variables are considered for the selection process:

- Class label: Boolean value. 1 if ideal cluster is assigned to any given digit maximum, 0 otherwise
- Difference in center-of-gravity and cluster width: A histogram sampling is performed to ensure a more uniform distribution over the training data
- $q_{\mathrm{tot}}/q_{\mathrm{max}}$ ratio
- Tsalis p_{T}-distribution of associated tracks
- Row and pad coordinate

It is an empirical observation that a flat density distribution over the input and output phase spaces leads to a significant improvement in the performance of a neural network. The input phase space of all possible charge positions and values can hardly be covered to perfection. Input grids are taken as a representative sample from the simulation, where sampling is only performed of the row and pad coordinate of each charge maximum, as well as the $q_{\mathrm{tot}}/q_{\mathrm{max}}$ ratio. It is assumed that the samples obtained from the simulation represent the charge distribution of clusters in real data well. Different sampling strategies are applied for the classification and regression network. A combination of the classification and regression networks into one network is not possible at training time, as the regression network can only be trained on clusters, which have an attached ideal position. In contrast, the classification network will see an equal sampling of cluster with class label 0 (has no ideal cluster attachment) and class label 1 (has ideal cluster attachment). Figure (2.3.9) shows the distributions over the row-pad coordinate space before (left) and after (right) the sampling process.

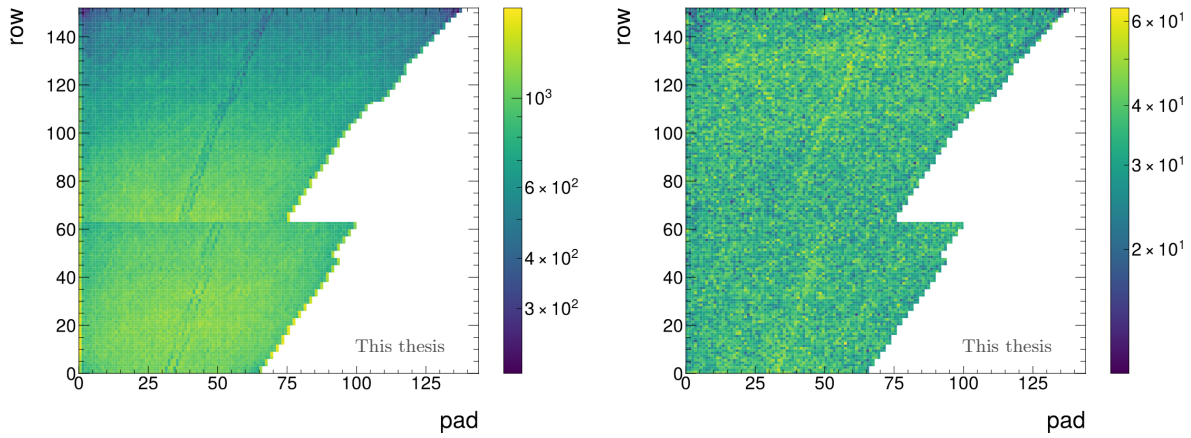


Fig. 2.3.9: Row-pad distribution of clusters before (left) and after (right) sampling.

A more uniform distribution over the row and pad coordinates is observed after the sampling process, ensuring a better phase space coverage. Pad-to-pad fluctuations of the density can

occur as a result of cluster availability. The training data elements is made configurable but chosen to be 1×10^6 for both the classification and regression network. An additional sampling is performed over the transverse momentum spectrum of the momentum vectors attached to the ideal clusters. It ensures a more uniform distribution of clusters with high and low inclination angles and their corresponding charge distributions. This implicitly downsamples over the input phase space of charges, regarding the inclination angles encountered, for three-dimensional inputs.

Several additional selections are made to improve the quality of regression data used to train the regression network. The following optimizations and cuts are largely empirically based, with some of them only investigated after the network training finished. Several iterations on these cuts were performed to improve fit performance without limiting the phase space or biasing the fit. While the training dataset should not be used to judge the final performance of the fit, it is a valid first check to identify if issues occurred during the training process. This argument is true as long as the training data distribution is a well-suited representation of the distribution of data on which the model will be ultimately applied to and no overfit has occurred at training time. The first optimization was noticed after training the neural network and investigating the performance on the training dataset. Figure (2.3.10) shows the distribution of the predicted q_{tot}/q_{max} estimate as a function of the MC truth.

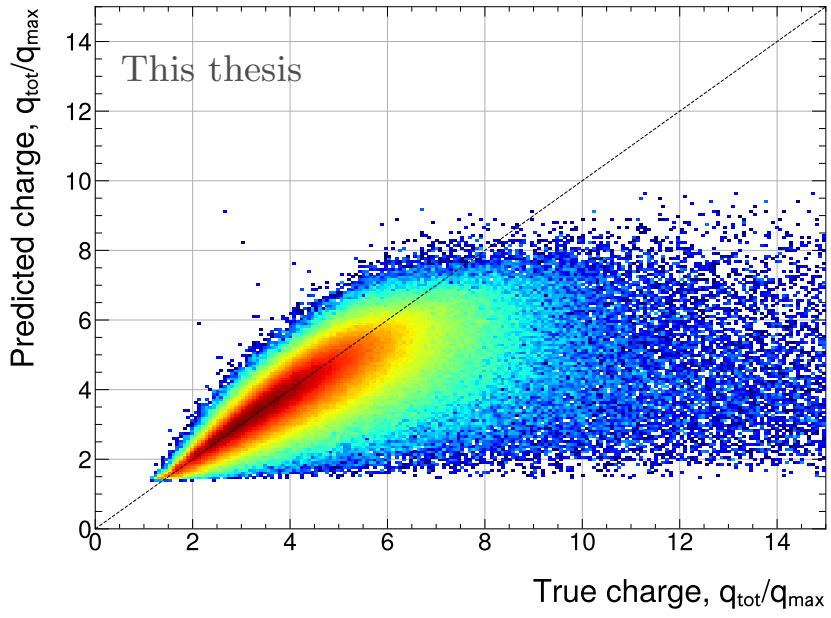


Fig. 2.3.10: q_{tot}/q_{max} estimate of the neural network compared to MC truth. The dashed line indicates a perfect fit to the ideal cluster property.

A clear distribution of outlier values is identified at high values of q_{tot}/q_{max} for the true charge ratio. Such high values (10 and above) can only be achieved by saturating clusters with tails

of the cluster distribution reaching outside the receptive field of view of the neural network. Additionally, clusters with such behavior have larger values of cluster widths, mainly in time direction. The combination of such charges into one cluster is not merely a weakness of the MC clustering algorithm, but rather a benefit, that even charge distributions of such large extent (with one MC label) are combined into one cluster. Nevertheless, such clusters can bias the fitting procedure to undesired side effects, with cluster centers too far away from the true digit maximum, resulting in random matches. To reduce such bad-behaved clusters in the training data distribution, a center-of-gravity, σ and q_{tot} calculation was performed in the standard way of summing the charges for the inner 3×3 neighboring cells around the peak. Cuts are then applied on the σ in pad and time direction and q_{tot}/q_{max} values, if certain values are exceeded. Figure (2.3.11) shows similar charge distributions with color coding for different cutoff values, where a quantity k (e.g. $k = q_{tot}/q_{max}$) is cut if $k^{MC}/k^{heur.,3x3} >$ threshold. The cutoff thresholds for the different scenarios are shown in the legend for the cuts on $[q_{tot}/q_{max}, \sigma_{pad}, \sigma_{time}]$.

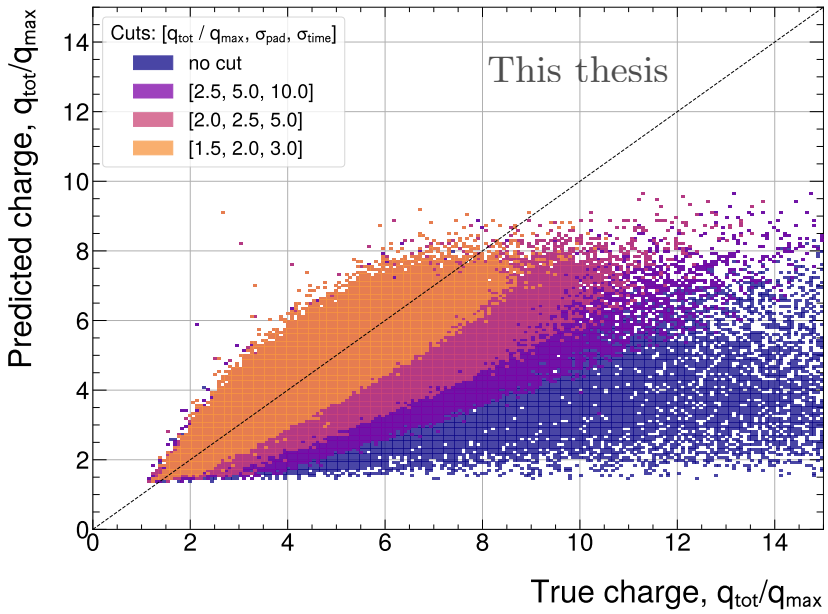


Fig. 2.3.11: Color coded training data distributions for different cutoff thresholds of the respective quantities listed in the legend. Illustrated is the projection of the predicted charge ratio as a function of the true charge ratio (q_{tot}/q_{max}).

For the network trained on the noisy distribution (no cut), the outliers in the training dataset are significantly reduced using cuts on the chosen variables. The cut on q_{tot}/q_{max} was found to have the most significant impact on the resulting quality. With the chosen cutoff of [2., 2.5, 5.] for $k = [q_{tot} / q_{max}, \sigma_{pad}, \sigma_{time}]$, the reduction in total training data volume was found to be 17% on the investigated dataset. A retraining of the network then leads to more stable behaviors even for large clusters.

2.3.3 Investigated model architectures

In view of computational complexity and memory operations, the input to both the classification and regression model will be structurally equivalent. An array of charges followed by the normalized values of sector, row and pad, which encode the position of the current peak, centered in the 3D charge array, is used in both the classification and regression case. The previous nomenclature will be adopted with (row,pad,time)-dimensions being marked as (r,p,t). The full coverage of all possibilities quickly exceeds compute and representation capacities, therefore a subset of interesting network structures within reasonable computational bounds is explored. For this project, fully connected and convolutional neural networks (CNN) are considered to be the most relevant prospects, with a realistic chance of application. While fully connected networks offer the best flexibility concerning the input phase space, with the possibility to add extra variables without structural changes to the model, convolutional networks have a long-standing history within the field, particularly concerning processing of image-like data. The architecture of CNNs is however more rigid, expecting a fixed $n \times k \times j$ input array (n, k, j integers). Additional inputs which do not obey such symmetry constraints cannot be processed by the convolutional layers. The CNN layers are therefore well suited to process the image type input data at hand, but require internal reshaping and fully connected layers once the geometric input of sector-, row- and pad-coordinate of the central digit maximum are added. For the full comparison with the current cluster finder implementation, both two-dimensional and three-dimensional input charge arrays will be investigated. In the two-dimensional case, the (r,p,t) nomenclature of the network will be reduced to (1,p,t), with only one row, containing the digit peak of interest, as the input array. Convolutional layers with three-dimensional architecture were considered in the first phase of this project as they offer wider flexibility on the kernel architecture of the convolutional layers compared to two-dimensional convolution kernels. However, since only one charge value is passed per (row,pad,time) coordinate in the input array, no advantage is gained in using the 3D convolutional layers (torch: Conv3D) over the 2D architecture, when using the channels of the 2D kernel as a third dimension (torch: Conv2D). For the main part of the investigation, only 2D convolutional layers will be used. This is not only a decision by first principles, but also a consideration concerning computational time. While 2D convolutional layers are optimized and common in many frameworks, 3D convolutional layers are less commonly used making their evaluations exceedingly expensive, as fewer efforts of optimizing this kernel are made within the machine learning community. Parallelization of convolution operations is in many cases a non-trivial task with choices to be made at runtime about the scheduling for layers with multiple channels. An example is the scheduling of NCHW (N = batchsize, C = number of channels, H = height, W = width) where parallelization and warp scheduling is first performed over the input channels compared to NHWC where scheduling is performed over the height and width dimensions of the kernel input. When many channels are used, NHWC is the

preferred format for GPU parallelization, optimized within many of the deep learning libraries. With a flat memory layout, fully connected networks do not require any such optimizations at runtime, but can significantly benefit from a smart choice in the designed architecture. This will be addressed in much greater detail towards the end of this thesis when investigating computational capacities and their impact on hardware acceleration (see chapter V).

III Analysis and performance on Monte-Carlo data

3.1 Reconstruction performance metrics

This project develops a new approach for cluster finding, which is the lowest level of TPC reconstruction algorithms (closest to raw data). Therefore it needs to be shown that reconstruction performance can either be maintained or improved. The following "standard candles" measure the quality of the reconstruction process and are considered necessary to quantify the validity of the algorithm

- Tracking and cluster-to-track attachment efficiencies and fake-rates
- χ^2/NDF distribution of the track fit after the reconstruction process
- Cluster rejection (i.e. reduction in data size)
- Matching efficiencies, most notably ITS-TPC matching

Many of the above mentioned quantities require exact knowledge of the processes at play and will therefore be investigated on Monte Carlo data. At analysis level, physics properties of the resulting tracks are of highest relevance. For this the following quantities are considered most relevant:

- $\mathrm{d}E/\mathrm{d}x$ performance (separation power, intrinsic resolution, PID performance)
- Invariant mass resolution of topologically identifiable V0 decays (Λ, K^0_S)
- Tracking performance (χ^2, shared cluster contributions)

Several of the listed items have large dependencies on calibrations, alignments and adjustments performed in the data processing chain. The analysis will aim to disentangle these effects where possible. While this chapter focuses on the investigation on simulated data, the succeeding chapter will investigate these properties using real data, including studies on environments of lower occupancy (e.g. pp, 500 kHz).

3.2 Cluster and track reconstruction on simulated data

This section is dedicated to the analysis on Monte Carlo (MC) data and can thus make use of MC labels at different steps of the reconstruction chain. MC labels are represented as a set of

three integers which connect a property of the detector readout or reconstruction process (e.g. charges, clusters, tracks) to an original process of a simulated particle. The three integers consist of track ID, event ID, source ID. The track ID has a direct mapping to the particle generated in the simulation, the event ID to the event from which the particle originated, and the source ID specifies the type of interaction from which a particle emerged. Based on these identifiers, each simulated digit charge, cluster and track can be uniquely traced back to the simulated process. Each digit can carry an arbitrary number of MC labels and weights. Digits with sub-threshold charges are removed. MC labels of reconstructed clusters are then created at reconstruction time from the digits contributing to a cluster, where each MC label is weighted by its charge contribution. A cluster can carry an arbitrary number of MC labels, including noise labels. This method offers MC label propagation through the full reconstruction chain and will be used to assess the reconstruction quality at the track level. While a reconstructed cluster can have more than one attached MC label, a track can not. The track is assigned the MC label which occurs most often among the clusters assigned to the track. If less than 90% of its clusters carry this MC label, the track is marked as fake (but carries the MC label of the majority vote).

Vital measures for the reconstruction quality are efficiencies and fake rates for both cluster attachment and reconstructed track to the propagated MC particle position. Several criteria are applied to limit the set of all simulated particles which are considered:

- $|\eta| < 0.9$
- $0.01 < p_{\mathrm{T}} < 20$ GeV/c
- If the particle is primary: $p_{\mathrm{T}} > 0.1$ GeV/c

The tracking efficiency is then computed as the ratio of tracks with a valid MC label (not noise or fake) to the total number of simulated particles, that satisfy the above criteria. Fake tracks do not contribute to the efficiency computation. The fake rate is computed as the ratio of fake tracks to the total number of reconstructed tracks (in contrast to the efficiency computation, the set of reconstructed tracks are used). If more than one track with the same MC label is found, all tracks except for the track with the highest number of attached clusters (i.e. the "longest" track) will be considered clones. The longest track with this MC label will enter the efficiency computation. Ghost tracks (i.e. tracks that are reconstructed due to incorrect cluster connections) can count to either the efficiency, clone or fake rate depending on their MC label content (e.g. 90% majority vote) and the presence of other tracks with the same MC label. In nearly all cases such tracks will count to the fake rate. A prime example for clone tracks are individual tracks found from a looping particle, as illustrated in the previous chapter (II). They are commonly referred to as "looper legs". No distinction between tracks on the level of individual track parameters is made and therefore all clone tracks are counted as real tracks.

The application of the neural networks for cluster finding then follows a simple process. At first, the input charge array is built in memory and passed to the classification network. It returns a floating point number between 0 and 1 for each cluster candidate (digit maximum), which determines the class label. If a cluster candidate exceeds a given threshold setting it will be used in downstream reconstruction processes, otherwise it is discarded. The tuning of this threshold setting will be a major part of this analysis. The second and more complex step of cluster regression is performed using different types of networks, namely fully connected networks and convolutional networks. The results are then passed to the reconstruction chain and compared to the current cluster finding algorithm. The currently employed algorithm with standard settings will be named as "GPU CF" (GPU Cluster Finder) in subsequent figures. The following cases are distinguished for the application of the neural networks:

- **FC**: Fully connected network,
  CNN: Convolutional neural network.
  If it is not explicitly specified, a fully connected network is used.
- **CF reg.**: Application of the heuristic regression algorithm to determine the cluster properties (center-of-gravity, width and total charge).
- **NN reg.**: Application of a neural network for cluster regression.
  NN class.: Neural network used for classifying clusters.
- **GPU CF**: Default cluster finder algorithm, no NN applied.

For the following MC studies central Pb–Pb ($\sqrt{s_{\mathrm{NN}}}$ = 5.36 TeV) interactions at high interaction rates are considered, as they represent the most challenging environment encountered by the ALICE TPC throughout Run 3 and 4 of LHC. The decisive factor for the reconstruction process are detector occupancies. With increasing occupancy, adverse effects of space-charge distortions result in degraded tracking and calibration quality. Contiguous central events are not realistic and result in significantly higher occupancies, cluster and space-charge distortions than are encountered in real data. Nevertheless, they offer a comparison between the proposed neural network algorithm and the current cluster finding method in dense environments. The following investigations will therefore consider different cases:

Pb–Pb simulations

- Collision system: Pb–Pb, $\sqrt{s_{\mathrm{NN}}}$ = 5.36 TeV, 350 events
- Anchored[1], Period: LHC24ar, Interaction rate: 35.4-38.0 kHz
- Detector noise on empty pads: enabled

[1] Anchoring refers to the usage of calibration and reconstruction objects and the tuning of the simulation to the data taking conditions of the real data run (anchor).

1. Case 1: Space-charge distortions: disabled, minimum bias[2], Injected high-p_{T} pions (30 π^+ + 30 π^- per event), low-momentum neutron capture processes: disabled
2. Case 2: Space-charge distortions: enabled, minimum bias, low-momentum neutron capture processes: enabled
3. Case 3: Space-charge distortions: enabled, centrality enforced collisions (0-5% most central)[3], Injected high-p_{T} pions (30 π^+ + 30 π^- per event)[4], low-momentum neutron capture processes: disabled

These cases are then compared in order to investigate the behavior of the neural network in the densest tracking environments (Pb–Pb, centrality enforced) and in data-taking conditions representing a realistic scenario. The centrality enforced simulation guarantees that the cluster finder works under extreme conditions, which ensures that rare, local high density regions in normal, simpler minimum bias collisions are well covered. The minimum bias simulation is created as an official ALICE simulation where looping tracks from such processes are enabled, generating an environment where the track density is significantly reduced compared to the centrality enforced, Pb–Pb cases.

It should be noted that the settings for the centrality enforced Pb–Pb simulations were used similarly for the creation of training data for the neural networks. Like this, large occupancy ranges are covered. It is ensured that the training dataset and the dataset on which the networks are evaluated in the following differ from one another, by specifying different seeding values for each simulation.

3.3 Pb–Pb simulation without space-charge distortions

This section will deal with the cluster properties on minimum bias Pb–Pb simulations without space charge distortions. These investigations are shown to illustrate and explain the investigated properties. Further studies then follow in similar fashion on space-charge distorted data in the second part of this chapter, connecting the improvements on cluster reconstruction to track level efficiencies.

3.3.1 Cluster residuals

Comparisons of cluster reconstruction quality is made against different variables such as the row number or local detector occupancy. The resulting distributions will be the residuals

[2]Minimum bias: Event generation without modification of the event (e.g. no centrality enforcement, signal / background injection, etc.)
[3]Centrality is enforced by triggering the simulation on events that satisfy the centrality criterion. The selection is performed on the impact parameter of the colliding nuclei
[4]High-p_{T} pions are injected to increase track statistics at high momentum (injected for $p > 3$ GeV/c)

(center-of-gravity and cluster width estimations) or ratios (total charge) between the reconstructed and matched ideal cluster properties. The residual distribution of such quantities does not follow a Gaussian nor otherwise easily quantifiable, well-known statistical distributions. However, determining the width of such distributions is necessary to quantify how changes in the reconstruction quality. As a distribution-independent measure for the width estimation of the residual distribution, percentiles will be used. An example is illustrated in figure (3.3.1).

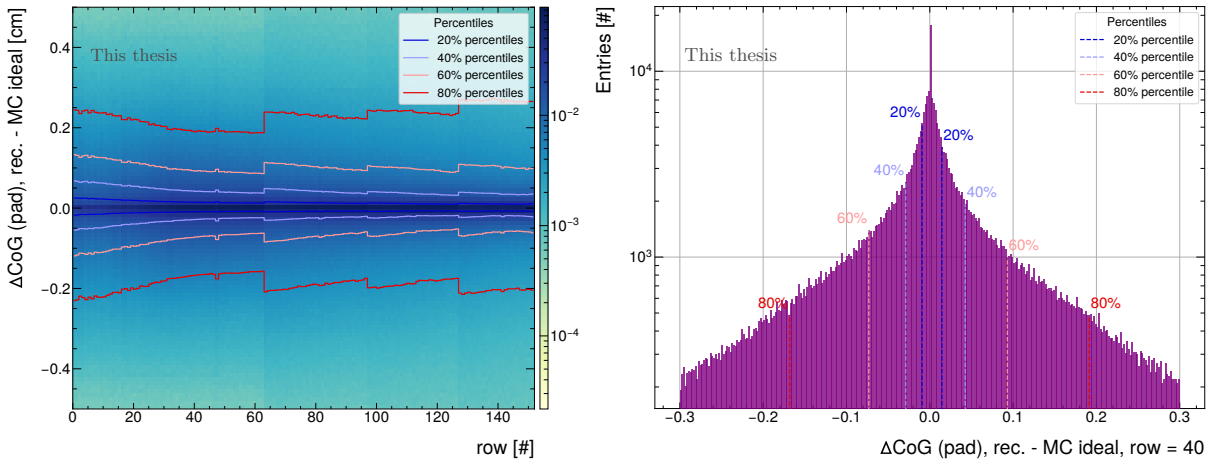


Fig. 3.3.1: Center-of-gravity residuals in pad direction (left) as a function of the cluster row number and slice at row 40 for illustration of percentile values (right).

The presence of a single, central peak of the distribution in all cases allows the calculation of percentiles for each half-side independently. The distribution is split along the bin with the maximum number of entries (typically close to $y = 0$). This accounts for potential asymmetries of the distribution and the finite receptive field of view of the input (i.e. the spatial charge distribution) of each clustering algorithm, specified by the row, pad and time window from which each cluster is built. To compare such distributions, the width (percentile) profiles of a given quantity, reconstructed with different algorithmic approaches, will be illustrated together in one plot. Each dashed line in figure (3.3.1, right) is created for each bin of the chosen x-axis (e.g. row number) in the two-dimensional plot (3.3.1, left) indicating the width of the distribution across a given variable (e.g. the row number). Figure (3.3.2) illustrates the combination of such width profiles using a neural network setting and the default cluster reconstruction method as an example. Percentile distributions for the (20%, 40%, 60% and 80%) are shown and will also be used in the subsequent figures.

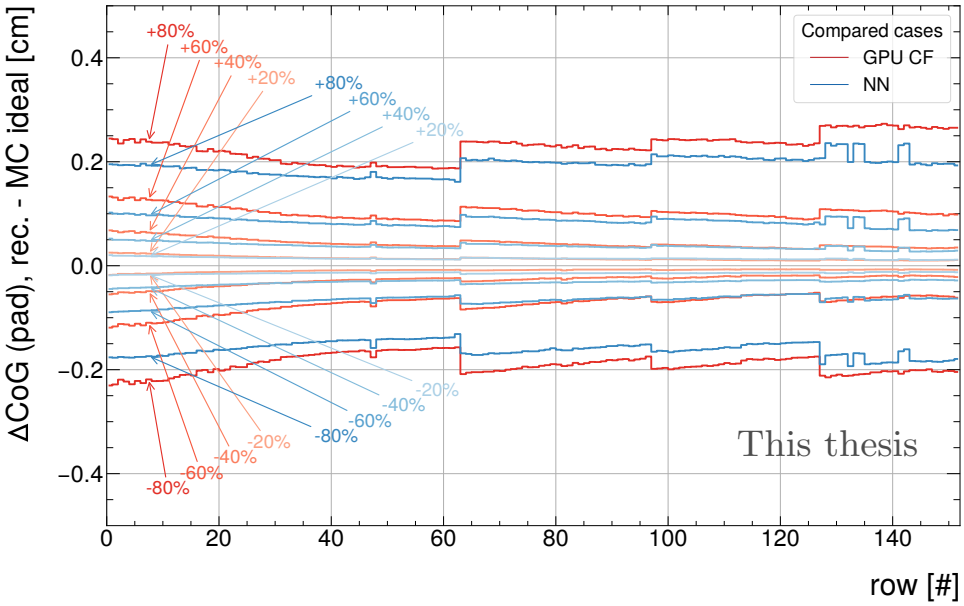


Fig. 3.3.2: Illustration of a width profile of the center-of-gravity (CoG) pad residuals with two different reconstruction methods, the default cluster finder and a chosen neural network cluster finder setting to illustrate the percentile lines.

The four major readout chambers of the TPC contain different pad sizes, visible as steps in the distribution. Different pad sizes lead to different cluster resolutions in radial direction, marked by the irregularities at rows 63, 97, 127. Within the set of rows for one readout chamber, a reduction in center-of-gravity residuals is observed with increasing row number. This is explained by track densities and therefore detector occupancy reducing approximately as $1/r$ (r being the radial distance away from the primary vertex). Reduced occupancies lead to fewer digit charge overlaps and therefore intrinsically better charge separation. Despite the increased overlap at low row numbers, the cluster resolution is still best in the innermost readout chamber (rows 0 - 62) as the pad-size is the smallest overall.

To separate the improvements found by the application of a classification and the regression network, two different algorithmic variants are implemented. The first one applies a classification network to the dataset, rejecting a subset of peaks (and therefore clusters) and afterwards applies the heuristic algorithm for the determination of cluster properties (typically marked by "NN class. (...) + CF reg."). This will leave the cluster properties unchanged, with the only effect being the removal of non-useful clusters, identified by the neural network. The second setting will make use of a cluster classification and a regression neural network (typically marked by "NN class. (...) + NN reg. (...)"). Usually these distributions will be compared to the default cluster reconstruction method ("GPU CF"). In the presented cases, only the width profiles of the percentiles are shown for this investigation. Minor variations of the peak position per slice on the x-axis of the distribution can result in small fluctuations, making it hard to visualize improvements of different algorithmic settings. The full profiles, including the minor

fluctuations of the peak are provided in appendix (B) for the space charge distorted case.

The first case, which only utilizes a classification network, is illustrated in figure (3.3.3). A threshold setting (0.03) was chosen and kept fixed for all classification networks, rejecting approximately 7-9% of clusters (depending on the network input, illustrated in the legend). Fully connected networks of 32 neurons per layer and 4 hidden layers are used for the task of classification and regression (in case a fully connected network is used). As will be shown in the computing performance (V) chapter, this is a preferable architecture for online deployment and is therefore the baseline of comparison for the following investigations.

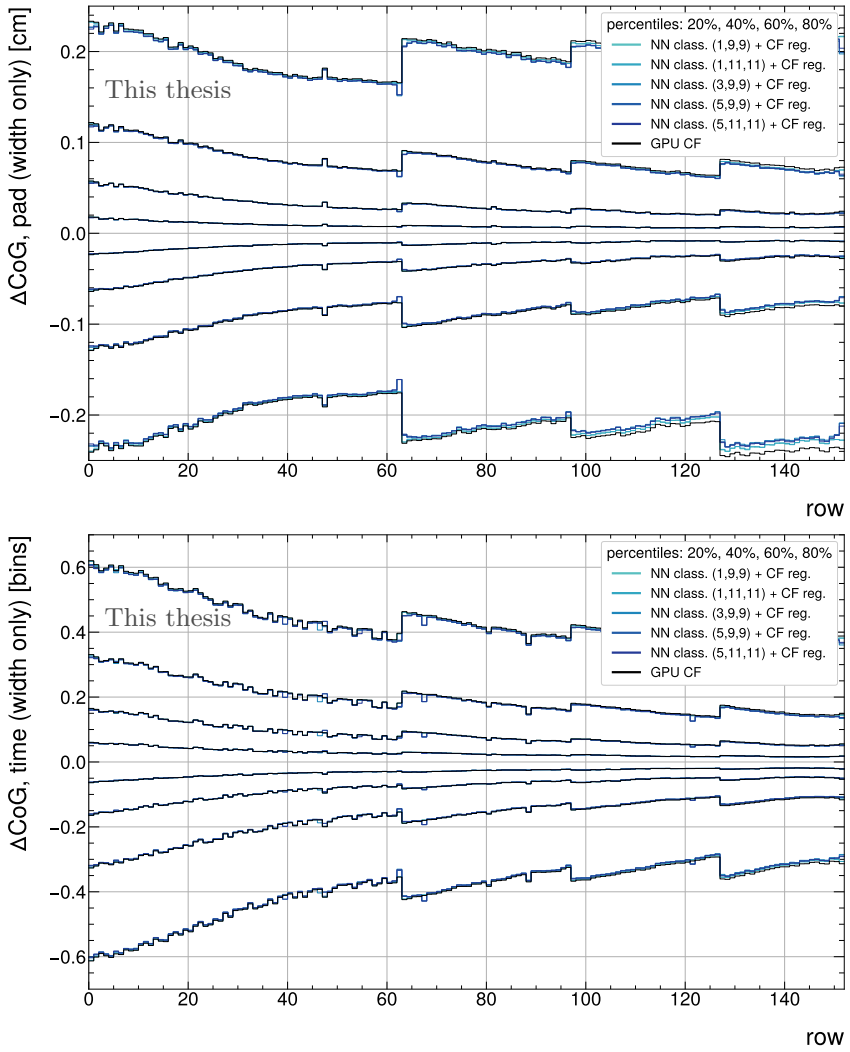


Fig. 3.3.3: Comparison of the center-of-gravity residuals in pad (top) and time (bottom) direction versus row number for the default cluster finder reconstruction algorithm (GPU CF, black line) and cluster classification networks combined with the heuristic regression method (blue shaded lines).

Therefore, cluster rejection alone does not provide a significant impact on the residuals. Only minor statistical fluctuations are observed. The property of cluster rejection will become sig-

nificantly more important in the case of tracking efficiencies and overall data size and motivate the studies performed in the next section (3.4) on space-charge distorted data. The residual distributions change significantly when the regression neural network is utilized. Figure (3.3.4) shows the center-of-gravity residuals using classification and regression networks (fully connected in both cases) in comparison to the default cluster reconstruction method.

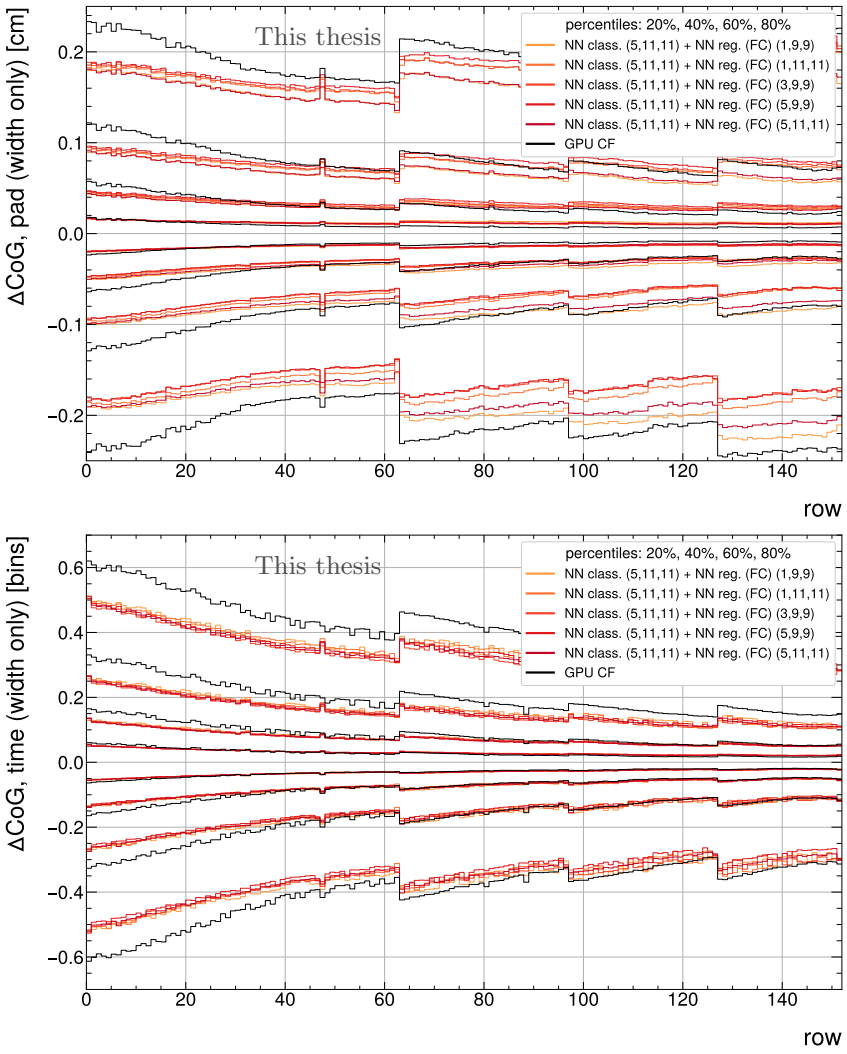


Fig. 3.3.4: Comparison of the center-of-gravity residuals in pad (top) and time (bottom) direction versus row number for the default cluster finder reconstruction algorithm (GPU CF, black line) and cluster classification and regression networks for fully connected networks used in both cases (orange-red shaded lines).

As the cluster classification does not significantly impact the residual distribution, the input size of the charge array to the classification network was kept identical (as (5,11,11)) in all cases and the input size to the regression network was varied. Towards the 80% percentile and for the center-of-gravity estimation in pad direction, variations in the network fit are apparent for different input sizes. The regression network performs overall on par or better than the current reconstruction algorithm in the inner readout chamber (up to row 63), dominated by

high detector occupancies. Towards the outer readout chambers, the network performs better at higher percentiles of the residual distribution. This region is dominated by clusters with high charge overlaps or noise contributions, in which case the network improves the center-of-gravity residuals. Towards the inner percentiles, the network does not improve the cluster residuals. This reflects the fact that in well separated areas with no overlap by other charges, clusters of small sizes are well described by the current algorithm. Slight deviations from the ideal cluster position do not impact the final track fit significantly, as these clusters are easily found and attached by the tracking algorithm. Clusters at the highest percentiles play a much more crucial role in this regard. In time direction (figure (3.3.4), right) the improvement by the regression network is noticeable across all rows and all percentiles. Particularly towards row 0 (innermost TPC radius), where detector occupancies are highest, the improvement at the highest investigated percentile (80%) is found to be approximately 20%. This reflects the fact that the time position is significantly harder to estimate using the center-of-gravity approach, as clusters are inherently non-Gaussian in time direction and can obtain a wider charge spread (see SAMPA time response function, eq. (1.1.5)). In this case, the network benefits from the flexibility of learning this property from training data, without relying on a rigid algorithmic approach and the larger input window of charges, compared to the default cluster reconstruction method. In both cases, no improvement is found between using a network with two or three-dimensional input or input charge arrays of different sizes.
A minimum size of the input array in pad and time direction is chosen to be (9,9), as the default reconstruction method utilizes a (5,5) charge accumulation grid and an algorithmic charge deconvolution window up to (9,9). The network labeled (1,9,9) therefore has the most-similar usage of information compared to the default algorithm. Providing fully equivalent information is not possible, since the default cluster finder uses different windows for charge deconvolution and for center-of-gravity computation, which the NN does in one step. It should be emphasized that the deconvolution is done fully by the neural network, no algorithmic deconvolution is applied in this case. From these studies it is concluded that the cluster classification has no impact on the cluster reconstruction quality, while the application of the regression network significantly improves the cluster residuals.

In addition to the fully connected networks, CNNs with the same input charge arrays were investigated. Figure (3.3.5) shows the cluster position residuals using CNNs for the task of regression.

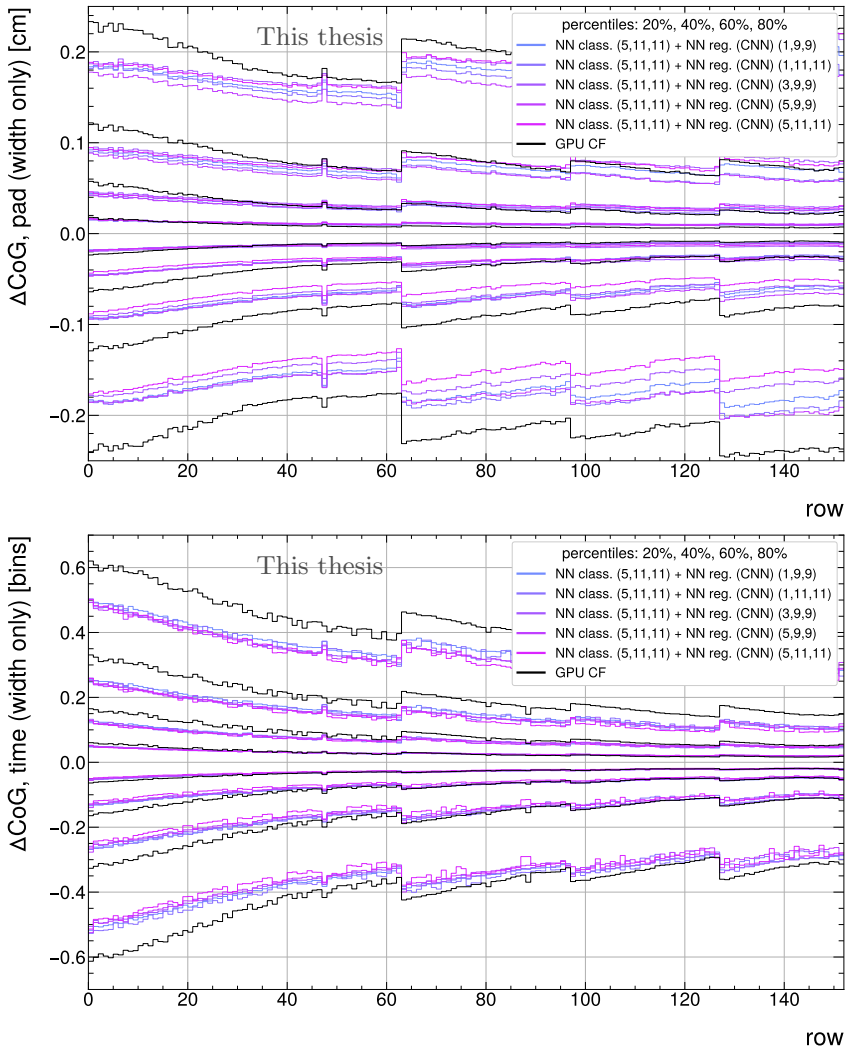


Fig. 3.3.5: Comparison of the center-of-gravity (CoG) residuals in pad (top) and time (bottom) direction as a function of the row number for the default cluster finder reconstruction algorithm (GPU CF, black line) and cluster classification and regression networks for convolutional connected networks used in both cases (violet shaded lines).

Similar improvements as with the fully connected regression neural networks are found in comparison to the default reconstruction algorithm. This indicates that the chosen architecture of the network does not have a major impact on the performance, concerning cluster residuals. The size and choices of architecture will be motivated in more detail in chapter (V) concerning the available compute capacity for online and offline deployment (V). In the following, all network settings will be combined in each figure representing the full spectrum of investigated architectures for this task. The color-scheme of the previous studies is kept, while the legend will be removed to avoid overlaps with the shown graphs (for the full legend, see appendix (B.0.1)). The remaining properties of cluster width (σ) in pad and time direction and the total charge estimation are the main focus for this part of the analysis.
Figure (3.3.6) shows the residuals on the cluster width distribution with all investigated neural network settings.

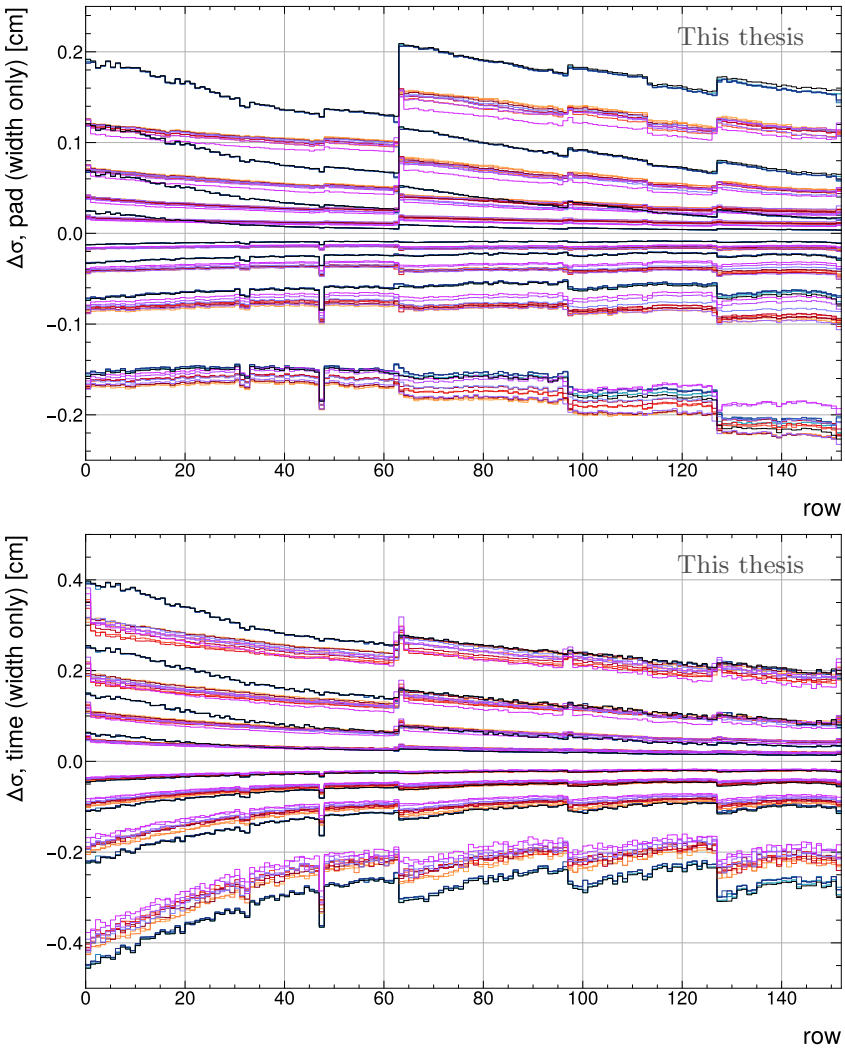


Fig. 3.3.6: Cluster width residuals in pad (top) and time (bottom) direction versus row number (for the full legend, see appendix (B.0.1)).

Similar to the center-of-gravity residuals, the neural network achieves a better performance at low row numbers. A closer inspection also reveals that in this case the improvement stems from the application of the regression network, with next to no change in the distribution when only the classification network is applied (blue curves). At the points of sector boundary and pad-size changes, clear spikes in the distribution are noticeable (e.g. row 47, 62), stemming from the three-dimensional input passed to the neural network: As the pad size changes the window of charges to the neural network still requires the symmetry of a regular grid, such as (3,9,9) although the charges in row n-1 might not equally represent the cluster shape compared to the charges in row n (because of the pad-size change). Such effects can be compensated by using larger neural networks, which however exceed the computing budget available in online deployment. This study is then once more repeated for the total charge estimate. Figure (3.3.7) shows the relative residuals of the total charge estimate by the neural network to the ideal clusters.

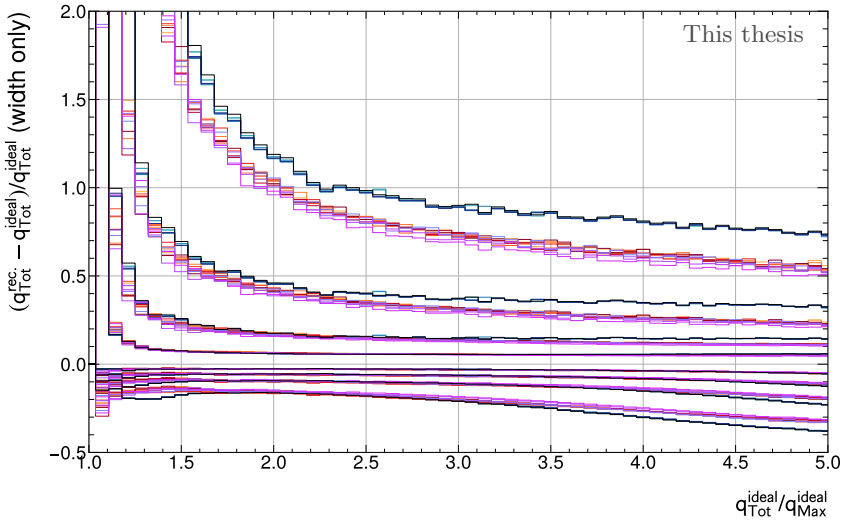


Fig. 3.3.7: Total charge estimate of the cluster performed by all investigated reconstruction settings as a function of $q_{\mathrm{Tot}}^{\mathrm{ideal}}/q_{\mathrm{Max}}^{\mathrm{ideal}}$ (for the full legend, see appendix (B.0.1)).

In contrast to the previous figures, the total charge estimate is not shown as a simple residual, as the residuals to the ideal charge scale with the maximum charge and spatial extent of the cluster. Rather it is shown as a function of the $q_{\mathrm{Tot}}^{\mathrm{ideal}}/q_{\mathrm{Max}}^{\mathrm{ideal}}$ ratio. This ratio is interpretable in the following way: Lower $q_{\mathrm{Tot}}^{\mathrm{ideal}}/q_{\mathrm{Max}}^{\mathrm{ideal}}$ ratios imply a more localized cluster (more charge contained in the peak), while higher values indicate a larger charge spread of the contributing charges. The network is therefore expected to perform increasingly better at increasing ratios of $q_{\mathrm{Tot}}^{\mathrm{ideal}}/q_{\mathrm{Max}}^{\mathrm{ideal}}$ due to the larger input window. From a ratio of 1.2 onwards, the network performs better than the default algorithm over all percentiles. At the very lowest values of $q_{\mathrm{Tot}}^{\mathrm{ideal}}/q_{\mathrm{Max}}^{\mathrm{ideal}}$, no significant advantage is gained with the network. However, such clusters are topologically close to single-pad-single-time bin clusters (= single digit cluster). Single digit clusters are rejected in the online reconstruction framework and are therefore not impacting the physics performance in case of an incorrect total charge estimation. At the highest charge ratio of 5.0 shown in figure (3.3.7), the regression neural network improves the total charge estimation by approximately 30% in the 80% percentiles relative to the default cluster finder algorithm and approximately 16% in absolute terms. This demonstrates the enhanced capabilities of the network to remove contaminating charges, compared to the algorithmic cluster finder. For $(q_{\mathrm{Tot}}^{\mathrm{rec.}} - q_{\mathrm{Tot}}^{\mathrm{ideal}})/q_{\mathrm{Max}}^{\mathrm{ideal}} > 0$ (i.e. the cluster charge is overestimated $\rightarrow$ result of charge overlap), the network performs a better charge deconvolution while at $(q_{\mathrm{Tot}}^{\mathrm{rec.}} - q_{\mathrm{Tot}}^{\mathrm{ideal}})/q_{\mathrm{Max}}^{\mathrm{ideal}} < 0$ both charge deconvolution and the larger charge window of the network take effect in the improvement. In all cases the application of the regression network is preferable to the heuristic method.

3.3.2 Impact of cluster resolution on tracking performance

The resulting improvements in the center-of-gravity residuals have a noticeable effect on the reconstruction quality of tracks. This is easily verified with the χ^2/NDF distribution per track after the track reconstruction stage. Figure (3.3.8) illustrates the desired improvement in the per-track χ^2/NDF distribution.

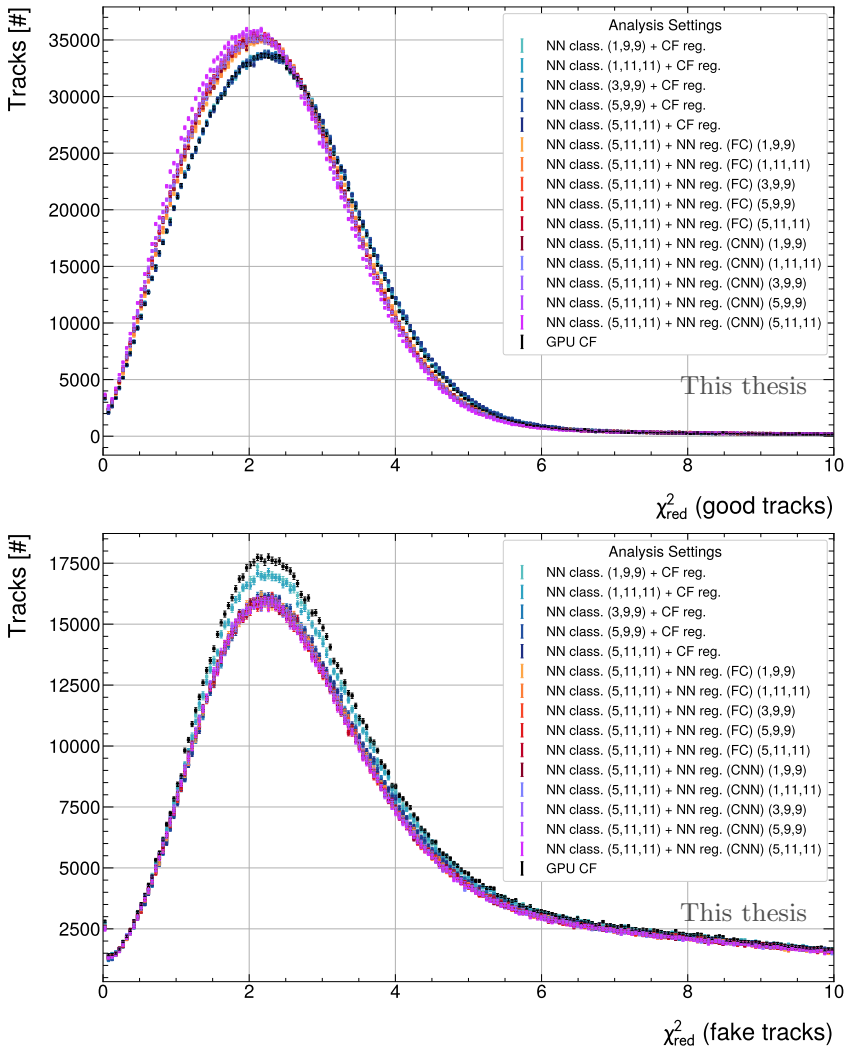


Fig. 3.3.8: χ^2/NDF distribution for the resulting tracks with both algorithms, for tracks with valid MC label (top) and tracks with fake MC label (bottom).

The χ^2/NDF distribution of good tracks (3.3.8, top) improves significantly when the regression neural network is used. Fake tracks (3.3.8, bottom) are reduced noticeably in all cases where a three-dimensional input to the classification network is chosen. This highlights the two main benefits of each applied network: While the classification network mainly improves fake cluster rejection and results in a cleaner tracking environment, the regression network improves the cluster properties and the track fit.

The number of degrees of freedom are determined by the number of free fit parameters and dimensions of the fit. For the helical track model, 5 free parameters exist. With two degrees of freedom per cluster (in pad and time direction), the number of degrees of freedom (NDF) is therefore $2 \cdot \mathrm{NCl} - 5$. A key value influencing the χ^2/NDF value are the cluster errors. Cluster errors are parametrized by a function and are scaled by a flag attached to the cluster: *isSplit*. This flag is set in the deconvolution step of the heuristic cluster algorithm and defines, if digit peaks are found in the 5×5 neighborhood of the investigated peak. While these flags are set in an identical for the application of the neural network, they are removed from error scaling for the χ^2/NDF calculation performed for this investigation. In this way, the charge deconvolution step is independent from scaling the cluster errors without artificially increasing the χ^2 for the neural network when split clusters are disproportionately highly removed by the classification step. The χ^2/NDF distribution for the good tracks (tracks with an assigned MC label) is shifted towards lower values, with a decrease in the peak position by 0.11 to 0.20 units (depending on the algorithmic setting), whenever the regression network is used. For fake tracks, cluster rejection already takes effect while in combination with the regression network, very short tracks (NCl$<$ 15), leading to large contributions to the fake distribution, are removed disproportionately often from the spectrum and lead to an overall decrease of fake tracks in the χ^2/NDF. In contrast to good tracks, no shift in the distribution is observed.

3.4 Pb–Pb simulation with space-charge distortions

A similar study as to the one outlined in the previous subsection is conducted on MC data, where space-charge distortions are enabled. This distortion is applied to the digits at simulation time using an inverted distortion correction map. At reconstruction level, the correction maps are then applied to the resulting clusters (not digits, by design of the ALICE reconstruction software). This simulation is specifically designed to emulate the detector response in the most adverse scenarios of high occupancy and track-density environments in mind. Two cases (case 2 and case 3, section (3.2)) are investigated side by side in this subsection: The centrality enforced Pb–Pb simulation with injected high momentum pions on one hand and the minimum bias Pb–Pb simulation on the other hand. The minimum bias simulation represents a nominal tracking environment and tests the algorithm under realistic conditions. The comparison of cluster properties will consume the first part of this study and will only be conducted on the centrality enforced simulation. Tracking efficiencies and fake-rates will then be compared on both simulations.
For the minimum bias sample, low momentum neutron capture processes are simulated using a novel generative adversarial network (GAN) to overcome the issue of compute time, while creating a realistic physics environment.

Figure (3.4.9) shows the residuals of the center-of-gravity estimate of the fully connected regression neural network cases and the default reconstruction in pad (top) and time (bottom) direction. To show the improved capabilities of charge deconvolution in dense environments, the plots illustrate the residuals as a function of local occupancy. For this study, occupancy is defined as the number of digits with non-zero charge in a window of 40 time bins and all pads within the readout chamber of the current charge maximum of interest, normalized to the total number of pad-time bins (therefore it is a value between 0 and 1, corresponding to a fraction of occupied digits). It quantifies how local occupancy degrades the cluster regression performance due to increasing charge overlaps.

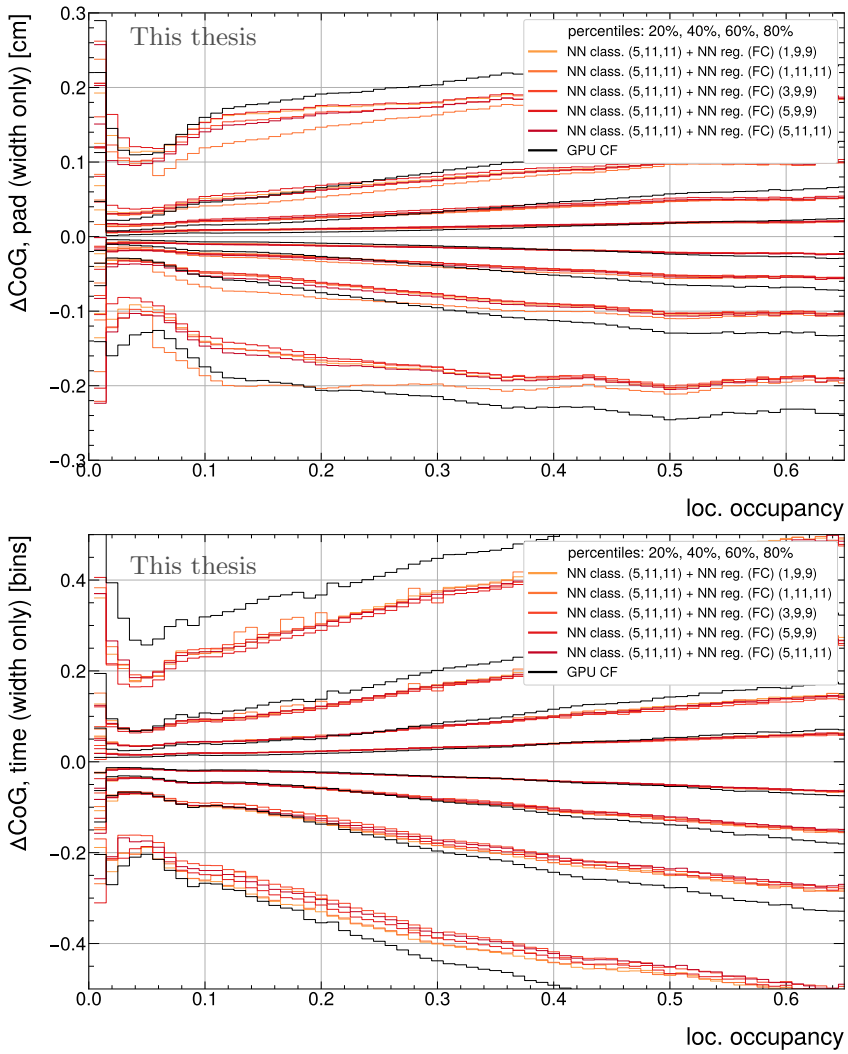


Fig. 3.4.9: Center-of-gravity residuals to ideal clusters using the default cluster finder method (GPU CF, black line) and the neural network cluster finder in pad (top) and time (bottom) direction for space-charge distorted data.

With increasing local occupancy, the neural network regression achieves a noticeable improvement in the center-of-gravity residuals compared to the heuristic cluster finder regression. This effect is a direct consequence of the application of the regression network, as the

applied distortion correction maps are locally smooth, changing cluster positions on the O(cm) scale, while the improvements take effect on the O(mm) scale. Therefore, the maps are considered sufficient for a direct comparison without recreating them for every case individually. All networks perform on par, with no noticeable difference in quality improvements between the networks with two or three-dimensional input windows. At the highest investigated occupancies of 65%, a performance improvement of $\approx$ 20% is observed for the 80% percentiles, which match the improvements found for non-space-charge distorted case from the previous section, at the innermost pad-rows. This emphasizes again, that space-charge distortions do not exhibit a strong performance decrease on local cluster reconstruction abilities.

To complete the cluster properties, the width and q_{Tot} estimate are investigated. Figure (3.4.10) illustrates the residuals of the cluster width in pad (top) and time (bottom) direction.

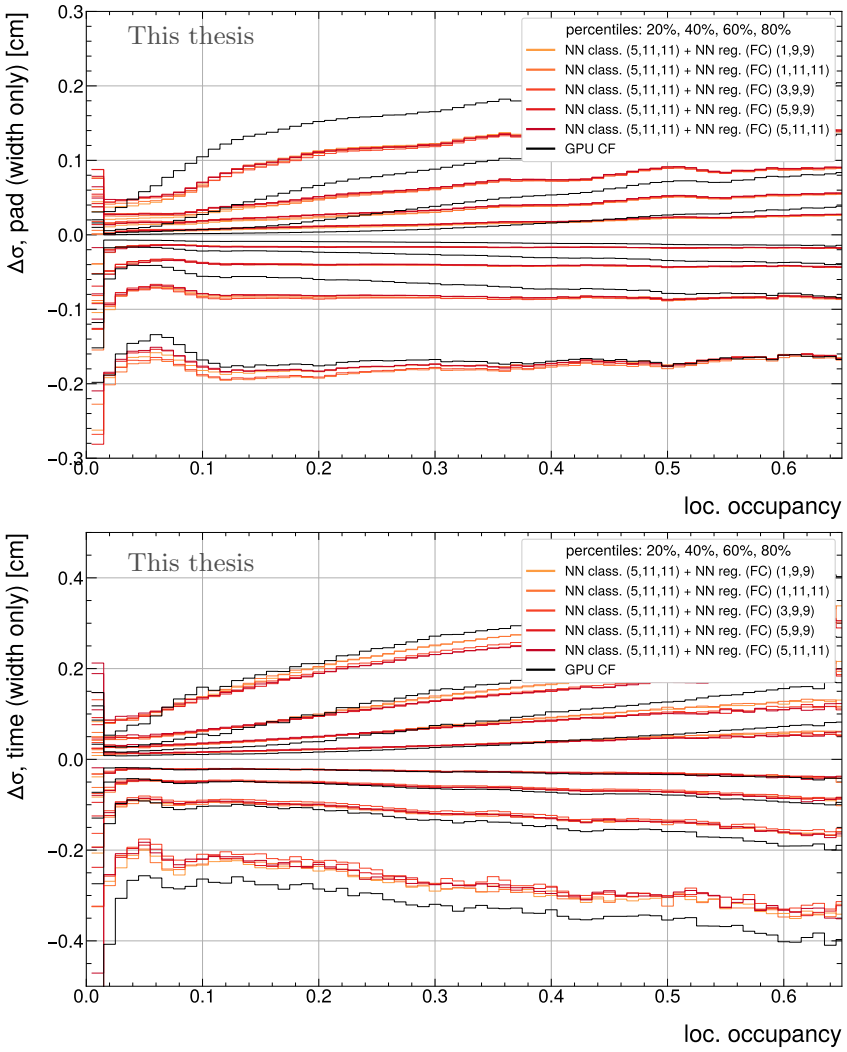


Fig. 3.4.10: Cluster width residuals to ideal clusters using the default reconstruction method (GPU CF, black line) and the neural network cluster finder in time direction for space-charge distorted data.

Similar to the center-of-gravity residuals, the neural network cannot bring a significant advantage over the heuristic cluster finder regression for the cluster width residuals at lowest levels of occupancy. This matches to the non-space-charge distorted case investigated in the previous section. In fact, at low occupancies and inner percentiles (20%, 40%), the network performs worse than the default cluster finder. This is an effect of the mean-square error optimization used for the network training: Lower values, close to 0 will create smaller additions to the total loss value, used in the backpropagation process. These will then lead to less significant updates of the network weights compared to clusters which have a large difference to the expected value. Therefore, for all investigations, it is expected that the network has a shallower peak structure around zero (wider 20% percentile distributions), while it will also create smaller tails of the distribution, noticeable at high percentiles. In pad direction and with increasing occupancy, the network and default cluster finder perform equally well for clusters where the reconstructed width is smaller than the width computed from the accumulation at simulation time (negative $\Delta\sigma$ values), while the network significantly improves the width estimation of clusters where the reconstructed width is larger than the ideal width (positive $\Delta\sigma$ values). This demonstrates the ability of charge deconvolution, as it shows that the network can reproduce width estimations more accurately for clusters appearing larger than they are, due to contaminating charges. In time direction, an improvement is observed for all cluster widths. The interpretation for small clusters remains, while for large clusters the network can more accurately reconstruct the width, due to a better utilization of the input charge array. Additionally, this demonstrates that the networks latent space representation is superior to the algorithmic charge deconvolution and cluster finding step, even at the same input size as the default reconstruction method. Overall, the width estimation is satisfactory and not considered in further investigations, as it is not used in downstream processing.

As a final investigation on the basic cluster properties, the total charge estimate is compared. As highlighted in the previous section, the relative residuals are compared. The total charge estimate will have a direct impact on the quality of the $\mathrm{d}E/\mathrm{d}x$ resolution per track and is considered crucial for the physics performance of this algorithm. Figure (3.4.11) illustrates this as a function of occupancy.

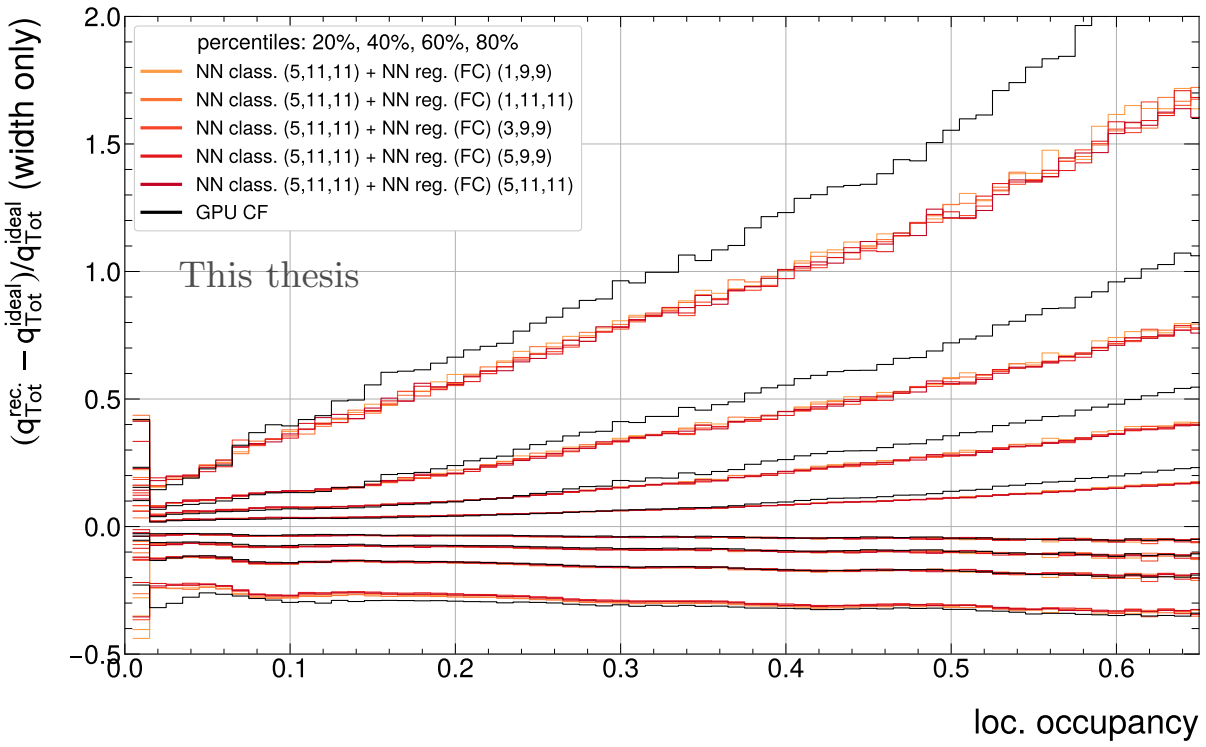


Fig. 3.4.11: Total charge, relative ratios to ideal clusters using the default reconstruction method (GPU CF, black line) and the neural network cluster finder for space-charge distorted data.

Here, the network improves the estimate noticeably, by up to 25% at highest occupancies (0.6 = 60% digit occupancy) and maintains the quality at lowest occupancies.

In conclusion these results demonstrate that all cluster properties improve using the neural network regression approach. Improvements are observed most notably for the higher percentiles, resulting in overall narrower residual distributions. Especially at highest occupancies, a 10% improvement was demonstrated for the center-of-gravity residuals. Furthermore, a 25% improvement was demonstrated on the total charge estimate at highest occupancies. In comparison with the cluster finder regression algorithm, it can further be said, that the improvements stem from the usage of the regression networks and not from cluster rejections performed by the classification network. While at lowest occupancies, both cluster finding methods agree in their results, the improvements increase with occupancy, demonstrating the charge deconvolution capabilities of the network.

3.4.1 Cluster matching

While the previous section highlighted the advantages of using the regression neural network on cluster properties, this section aims to disentangle the effects of the classification and regression improvements further. The focus will now shift towards the improvements brought

by removing non-useful clusters. For this purpose, the effect of different threshold settings is studied on the matching between reconstructed and ideal clusters. The centrality enforced Pb–Pb simulation is used for this purpose. The heuristic regression algorithm for the cluster properties is used in combination with the classification networks to disentangle between effects stemming from the improved cluster position estimation by the regression neural network. Matching the resulting clusters with the ideal cluster positions gives a direct measure for the efficiency and fake rate for each case. This is illustrated in figure (3.4.12). The efficiency is measured as the total number of reconstructed and matched clusters as a ratio to the total number of ideal clusters, while the fake rate is measured as the number of unmatched reconstructed clusters as a ratio to the total number of reconstructed clusters. No removal of high inclination clusters is performed for this first study.

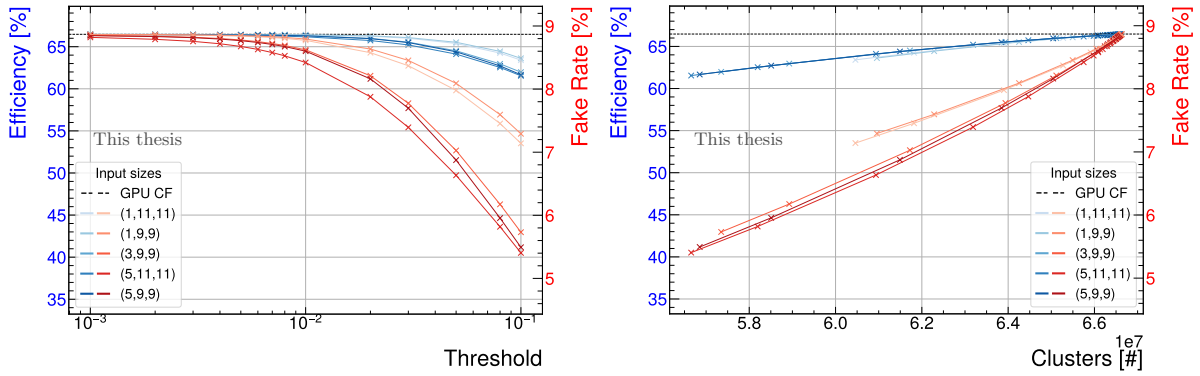


Fig. 3.4.12: Efficiency (blue) and fake rate curves (red) for different input sizes as a function of classification threshold (left) and number of reconstructed clusters (right).

Figure (3.4.12) has two separate y-axes, one for the efficiency and one for the fake rate. The axes are scaled to match for the efficiency and fake-rate obtained with the default cluster finder (dashed line). Blue curves are to be read on the blue axis (left), red curves correspond to the red axis (right) and represent the efficiency and fake rate at a given point on the x-axis. The points are connected linearly for better readability. The two plots differ by the representation on the x-axis: The left plot illustrates the behavior against the chosen threshold setting, while the right plot illustrates the total number of reconstructed clusters. The relation between the number of reconstructed clusters and the chosen neural network threshold is shown in the appendix (B.3.17). With increasing threshold the number of reconstructed clusters decreases, however, this correlation is not linear. The points used in the left-hand plot correspond directly to the points in the right hand-plot (e.g. for the networks with three-dimensional input arrays, the points at the highest rejection threshold, 0.1 in the left plot, correspond directly to the points with the lowest number of reconstructed clusters, close to 5.7×10^7 in the right plot). Different input sizes are illustrated with different color intensities. The most opaque colors indicate the networks with two-dimensional input.

Both the neural network and default cluster finder utilize the same peak finding algorithm. Therefore, if no cluster classification is applied, both algorithms have the same efficiency and fake rate. In the NN case, the efficiency and fake rate will therefore asymptote towards the default cluster finder (indicated by the dashed line) as the threshold approaches 0. With more aggressive threshold settings, the network rejects clusters, achieving a reduced fake rate. The efficiency is reduced with increased cluster rejection, as clusters of looping tracks are present in the sample of ideal clusters (and explicitly not excluded) but are suppressed in the reconstruction process with increasing threshold. As a function of number of reconstructed clusters, all networks with three-dimensional input perform noticeably better than the two-dimensional networks, achieving the same matching efficiencies at lower fake rates. The improvement in the fake rate is approximately 10%. Additionally, the fake rates are reduced compared to the default cluster finder without cluster classification. This motivates the choice to use a three-dimensional input for the neural network, while keeping it as small as possible, to reduce computational complexity. The study therefore concludes that the (3,9,9) input window is the best choice, with no significant performance losses compared to the other chosen three-dimensional input sizes.

Previous studies have shown that a different peak-finding algorithm can improve the efficiencies above the default cluster finder. The effect was however found to be only ≈1% improvements in efficiency (absolute). Such a study, done with 3D convolutional layers can be found in the appendix (B.4.18). Both the 3D convolutional layers and the alternative peak finding algorithm are however more computationally expensive, as significantly more peaks have to be investigated. Furthermore, no beneficial results on track reconstruction level were found. For this reason, the current peak finding algorithm was not modified.

To better observe the behavior of clusters useful for reconstruction, figure (3.4.13) illustrates a different metric for the efficiency: the number of ideal clusters with matched reconstructed clusters, excluding looper tagged regions, normalized to the total number of ideal clusters. In this way, it is demonstrated how the efficiency improves even when clusters of looping tracks are avoided.

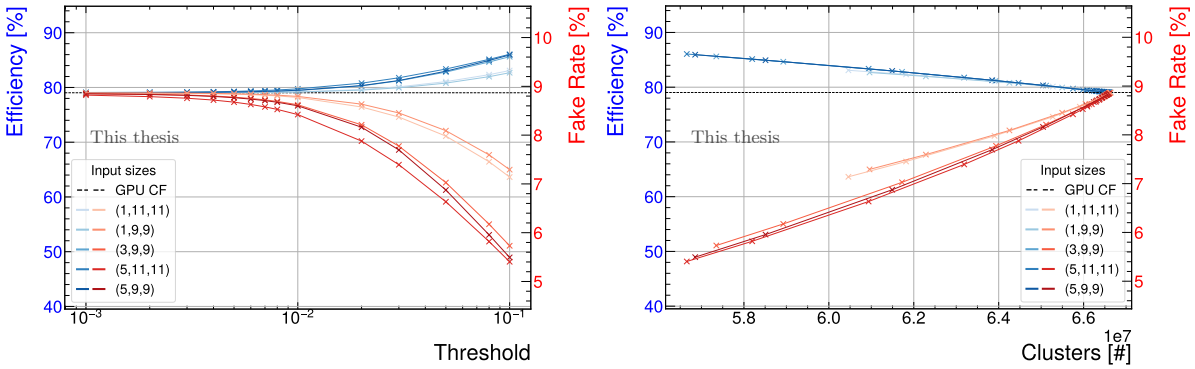


Fig. 3.4.13: Efficiency (blue) and fake rate curves (red) for different input sizes as a function of classification threshold (left) and number of reconstructed clusters (right). Efficiency is calculated as the number of non-looper track clusters normalized to the total ideal clusters, excluding all occupancy tagged regions (see figure (2.1.2)).

This further illustrates that the network accurately rejects clusters within high inclination areas of tracks. Measuring the efficiency in this way can artificially inflate the improvements with increasing number of looping tracks. However, it illustrates that the relative number of matched and useful clusters is increasing, showing that clusters of looping tracks are successfully removed from the dataset. A decision on the threshold setting can then be made in the following study, on track reconstruction level, as the cluster-to-track attachment is measured similarly. Instead of the ideal MC clusters, the cluster matching to tracks is investigated based on the MC label attached to a cluster at reconstruction time (from the sum of digits and a majority vote) and the MC label of a track (by majority vote of all cluster MC labels, contributing to the track fit). The MC label of a track and the attached cluster are compared and counted towards the efficiency if the labels match and are otherwise declared as fake matches. Figure (3.4.14) shows the relation between the correctly attached clusters for non-fake tracks as a function of the total number of reconstructed clusters.

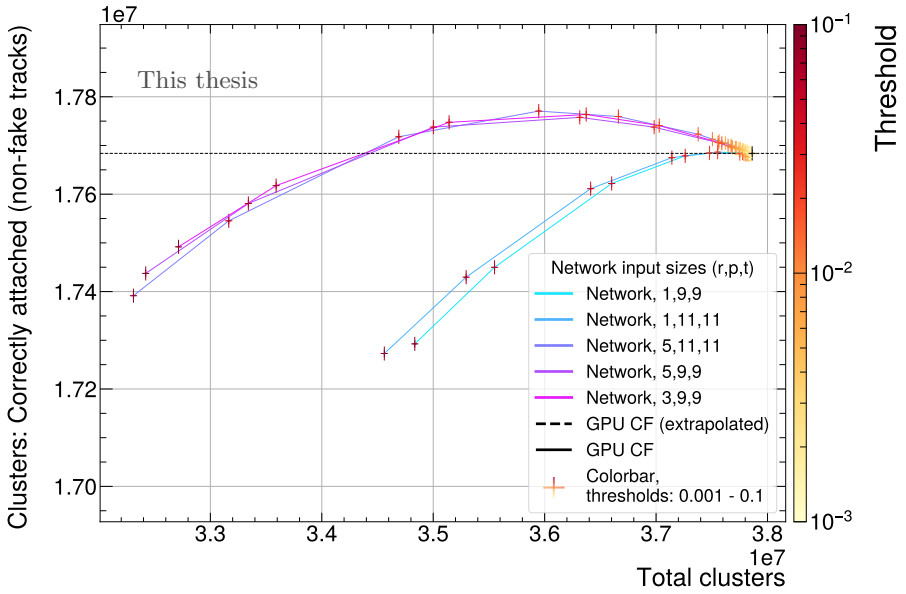


Fig. 3.4.14: Correctly attached clusters as a function of total reconstructed clusters for clusters from non-fake tracks. The color axis corresponds to the chosen threshold setting of the individual points, connected by colored lines. The GPU CF point is extrapolated, to illustrate at which threshold range the network crosses the number of correctly attached clusters.

With increasing classification threshold the total number of attached clusters decreases. The characteristic shape of the graph indicates that correctly attached clusters are removed disproportionately less than total clusters. This is a direct consequence of fake cluster removal and looping track rejections. It can further be concluded that at the same number of correctly attached clusters as the default reconstruction method, the networks with three-dimensional input can still noticeably reduce the number of total clusters compared to the networks with two-dimensional input, matching the conclusion from the previous studies. This effect further demonstrates that it is not advisable to consider networks with two-dimensional inputs for the classification task, while no apparent distinction can be seen for either one of the networks with three-dimensional. This motivates the choice of (3,9,9) as the smallest investigated size to be the best trade-off between quality and compute time. For non-fake tracks, using the three-dimensional input networks, the total number of clusters is reduced by approximately 10% (threshold 0.05), while maintaining the same amount of correctly attached clusters. Between 0.05 and 0.08, the number of correctly attached clusters for non-fake tracks passes under the extrapolated threshold of the current reconstruction algorithm. While this demonstrates the strong capability of the network to reduce the total number of clusters, it also shows how fake-cluster attachment can degrade tracks. In the intermediate regime with thresholds smaller than 0.05, the network finds more correctly attached clusters showing that fake tracks are converted into good tracks due to reduced contamination. This conversion takes effect on up to 0.5% of total clusters, while at the same number of correctly attached

clusters, the network achieves a 10% total cluster reduction. Both effects have an impact on the clusters entering the tracking phase and the $\mathrm{d}E/\mathrm{d}x$ calculation. Moreover, this graph can be used to determine a feasible range of classification thresholds for a given network configuration under specified quality criteria for deployment purposes. In conclusion, the network finds more good tracks while reducing the total number of clusters. Concerning the efficiency and fake rates of non-fake tracks, a threshold between 0.03 - 0.08 is recommended, depending on the requirements of cluster rejection.

3.4.2 Impact of model size

This part of the analysis will consider the effect of different model sizes on the cluster finder performance. It is a well established idea within the machine learning community that larger models with more training data will (on average) result in better final results than smaller models ("The bitter lesson", Richard Sutton[5]). This observation was quantified in the context of billion-parameter large language models (LLM) in the form of scaling laws [40]. The models used for the problem of cluster finding are orders of magnitude smaller as they are designed for throughput-critical applications. While training time of the models are of no concern for this work (due to small models and ample resources), inference time considerations are a critical part of this analysis and will be addressed in chapter (V) of this thesis. Contrary to industry developments or the recent trend of exponentially increasing model size (in terms of number of parameters) with time [43], it is one of the goals of this thesis to show what can already be achieved with small models, smart design choices and careful selection of the training data, even for large scale applications and non-trivial, physics-based problems.

In order to investigate the impact of the model size on the final physics performance, the impact on cluster resolution is tested with different fully connected models ranging from 2 to 5 intermediate layers and 16 to 128 neurons per layer (increasing exponentially as 2^x, where x is the number of neurons). The result on the center-of-gravity residuals in pad direction are shown in (3.4.15).

[5] http://www.incompleteideas.net/IncIdeas/BitterLesson.html

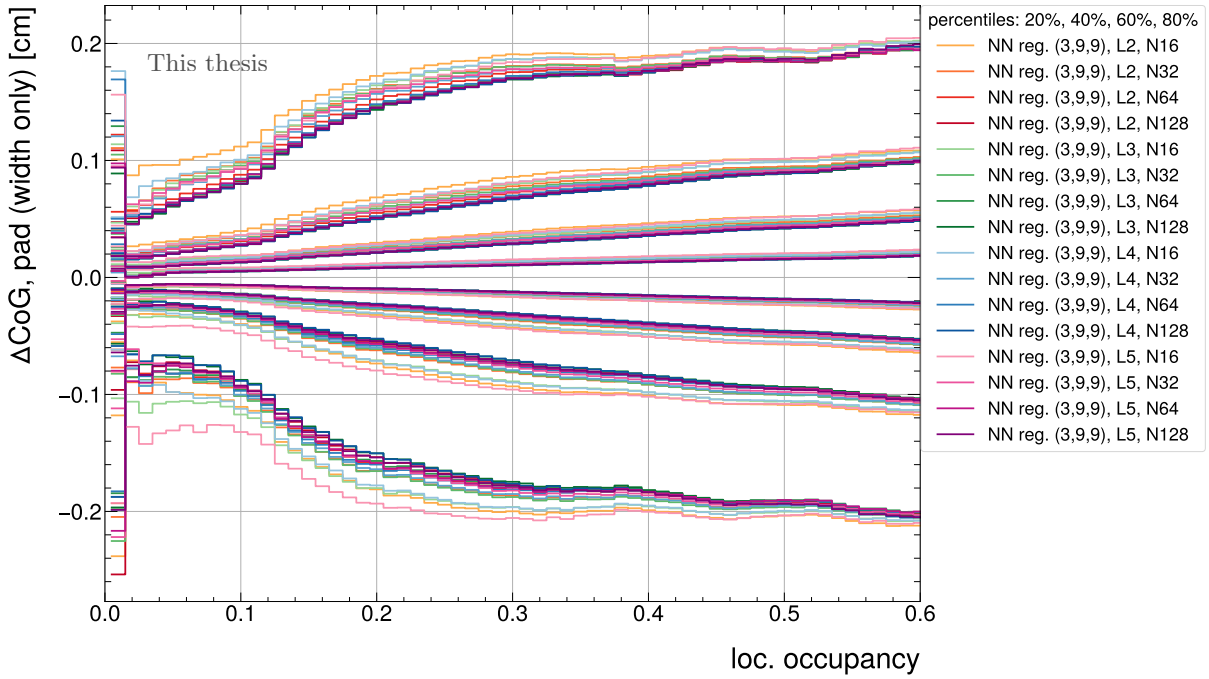


Fig. 3.4.15: Effect of model sizes on the center-of-gravity residuals. L: number of intermediate layers, N: number of neurons per layer.

The same classification model and threshold was used to reject clusters for all cases ((5,11,11) input size, threshold = 0.03). While the smallest models of 16 neurons per layer show significant reductions in quality of the cluster resolution, all other models tend to result in similar performance. At a local occupancy of 0.5, the percentile distributions widen by approx. 10%. At the innermost percentiles of 20%, the regression quality deteriorates even more, up to 24% for the smallest model (L2, N16) compared to the largest one (L5, N128). Models with more neurons per layer perform overall best, shown by the accumulation of darker colored lines towards the center of each percentile distribution, particularly noticeable at high occupancies. Furthermore, the residual distributions further improve noticeably with increasing number of layers. The trade-off concerning inference time will be made at a later stage, while for now all models with N$\geq$ 32 are considered acceptable.

3.4.3 Results on track reconstruction level

For a comprehensive overview of the final model performance it is imperative to study the effects on the track level. For this analysis, the two separate simulations (minimum bias Pb–Pb simulation and centrality enforced Pb–Pb simulation) will be shown next to each other. The intent of showing the two simulations (case 2 and case 3, section (3.2)) is to distinguish the rejection capabilities in environments of moderate to high track densities, covering the nominal to most adverse scenarios. Figure (3.4.16) shows the tracking efficiency for primary particles as a function of the MC p_{T}. Only certain representative networks are chosen from

each category (2D input, 3D input, FC, CNN) for illustration. This first analysis focuses on the differences between different input sizes and network architectures. Figure (3.4.17) shows the tracking efficiency of primary particles reconstructed with the selected network settings as a function of their transverse momentum. For all classification models, a threshold of 0.03 was chosen, amounting to ≈ 8% reduction in total number of clusters.

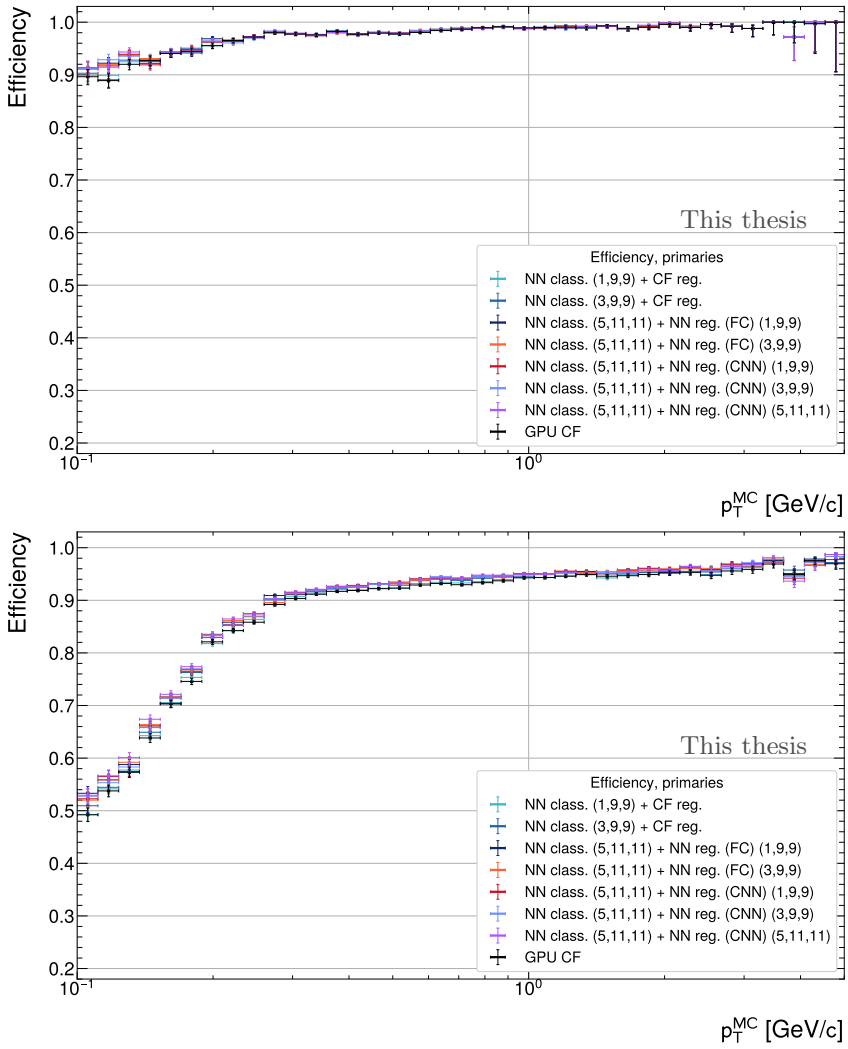


Fig. 3.4.16: Reconstruction efficiency for the minimum bias simulation (top) and the (unrealistic but dense) centrality enforced (bottom) Pb–Pb simulation as a function of p_T for primary particles.

For the centrality enforced simulation, the neural networks improve the efficiencies p_T intervals. On the minimum bias sample, this effect is only mildly noticeable at very low transverse momentum (< 0.2 GeV/c). It can also be seen how tracking degrades with increasing centrality / detector occupancy, as the efficiency only reaches 50% in the lowest p_T bin for the centrality enforced simulation, compared to 90% in the minimum bias sample. It should be emphasized that the tracking framework was not developed for such environments, where cluster density exceeds the nominal scenario of minimum bias 50 kHz, Pb–Pb interactions. To investigate the

effect of the different networks in more detail, the efficiency is shown as a ratio to the default cluster finder in figure (3.4.17) for primary particles.

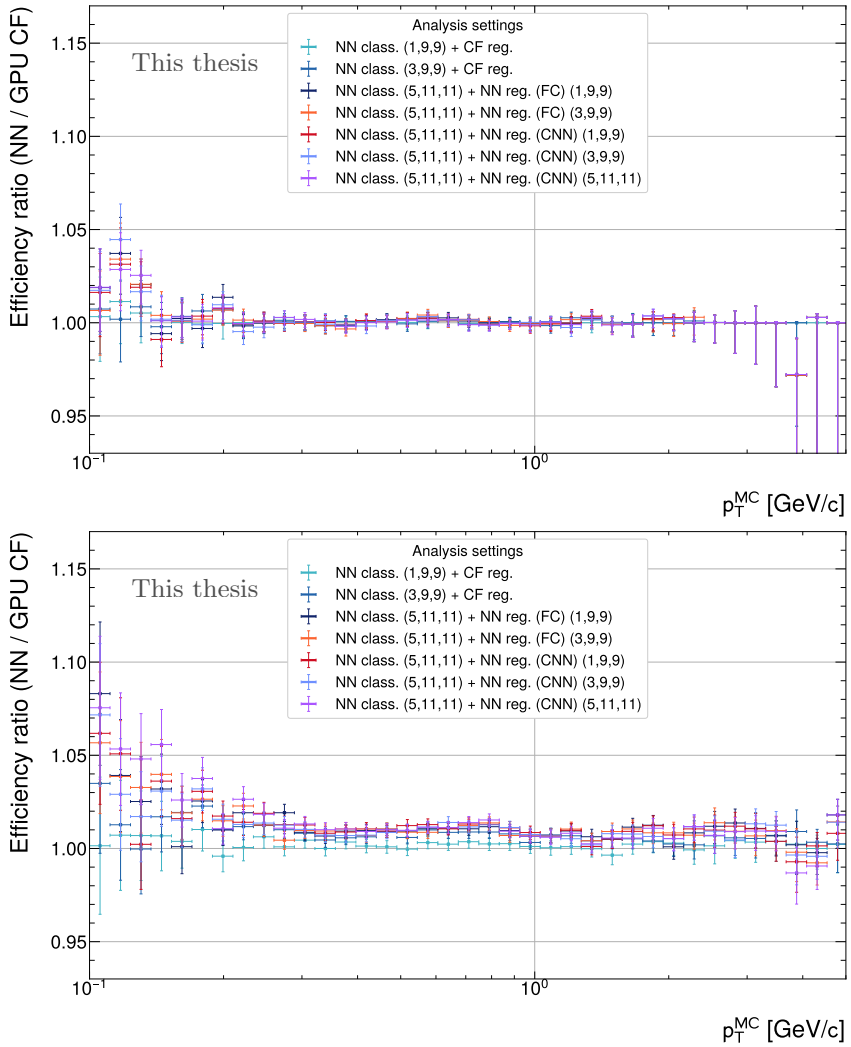


Fig. 3.4.17: Reconstruction efficiency ratio to the default cluster finder reconstruction for the minimum bias (top) and the (unrealistic but dense) centrality enforced simulation (bottom) as a function of p_T for primary particles.

A systematic trend towards higher efficiencies is noticeable on the centrality enforced sample, while all efficiencies agree within the uncertainty for the minimum bias sample. For the centrality enforced simulation, the neural network can improve efficiencies by approximately 1.5% across the full p_T range when a classification network with three-dimensional input is used. A more striking difference is found for secondary tracks, since they will extend to lower transverse momenta and will include looping tracks. All tracks, that are not produced in the primary collision and have a distance larger than 100μm to the primary vertex when propagated to the beam pipe, are counted as secondaries. Figure (3.4.18) shows the efficiency for such secondary tracks as a function of p_T for both simulations, where the kinematic range is now extended to $p_T \geq 0.01$ GeV/c.

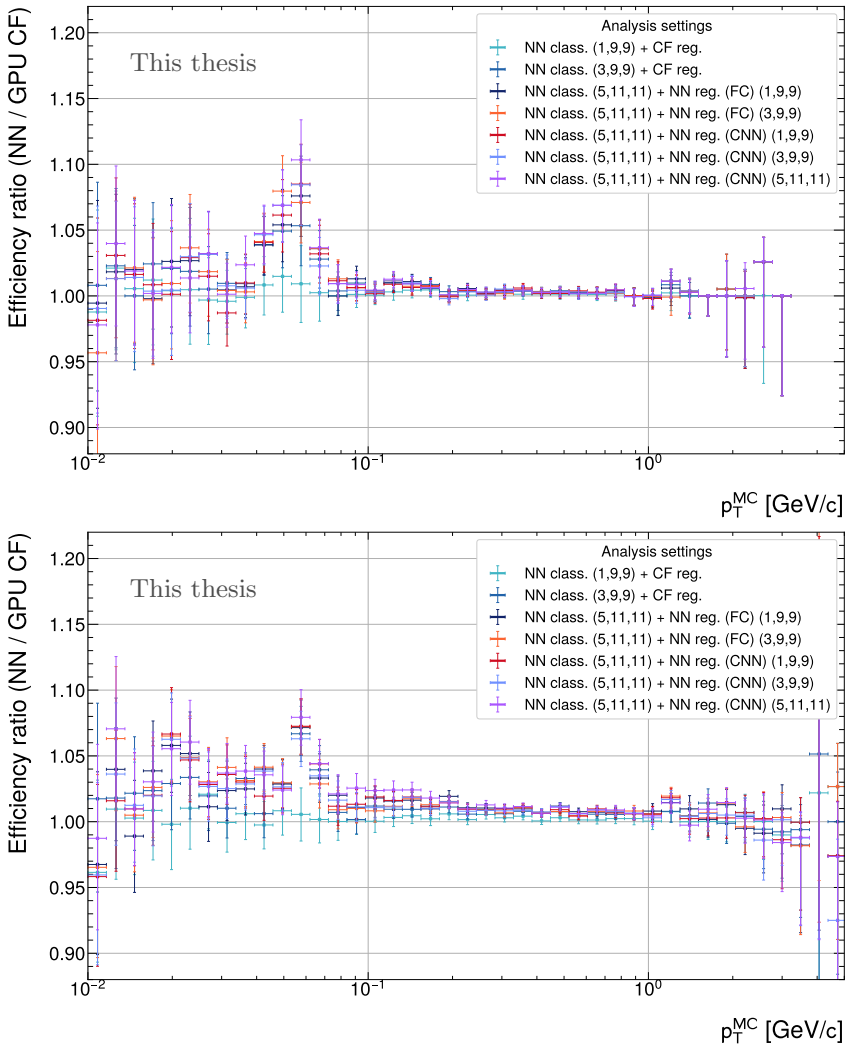


Fig. 3.4.18: Reconstruction efficiency ratio to the default cluster finder reconstruction for the minimum bias (top) and the (unrealistic but dense) centrality enforced simulation (bottom) as a function of p_T for secondary particles.

With compatible efficiencies at high transverse momenta for all networks, a systematic improvement is observed for $p_T < 0.08$ GeV/c. This region is mainly dominated by low momentum electron looping tracks, for which many clusters are rejected by the network. All cases using the (5,11,11) network for classification show an improvement here. The network maintains or improves the efficiency of secondary particles across the full kinematic range. From this investigation it is concluded that all networks using three-dimensional input for classification perform on par with each other. The analysis will therefore be continued on one network, which is considered most relevant for deployment in real-time processing. The choice and size of network is justified in a chapter dedicated to the computing performance (see chapter (V)), by the cluster residual studies from the previous sections and by the ROC (Receiver Operator Characteristic) [44] study shown in figure (3.4.19).

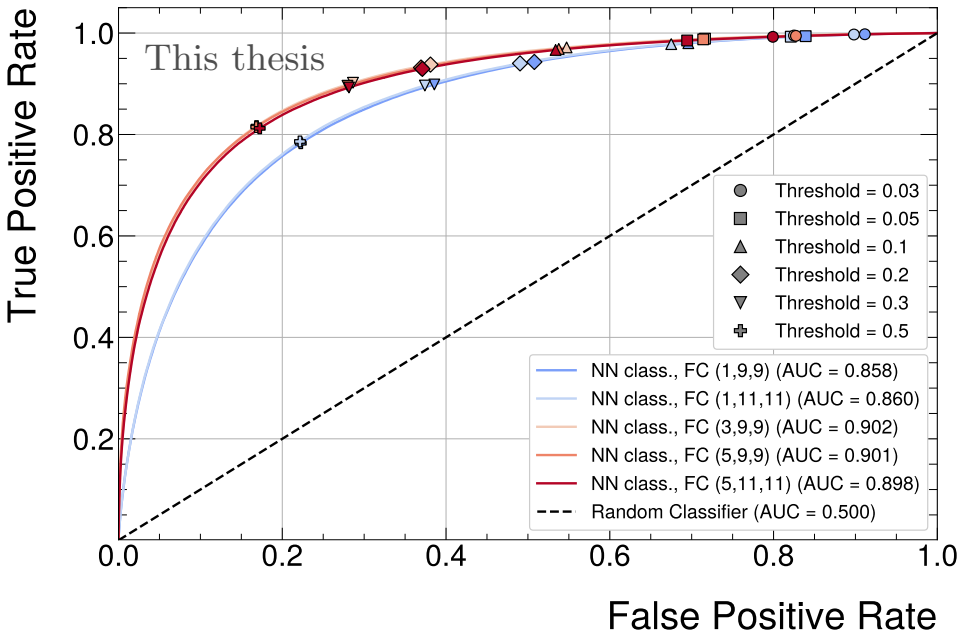


Fig. 3.4.19: ROC curve and AUC values for classification networks with different input sizes on 1×10^6 elements of the centrality enforced Pb-Pb simulation.

The ROC characteristic shows the true positive rate as a function of the false positive rate where the threshold of the classifier is varied, while the Area Under Curve (AUC) gives a single value, quantifying the quality of a classifier (1.0 = perfect classification, 0.5 = random classification).
By the AUC metric, all networks with three-dimensional input perform better than the networks with two-dimensional input. However, no significant improvement between the network with three input rows and five input rows can be observed, therefore additional rows do not contribute valuable information for the task of classification. As a trade-off between physics and compute performance, the network with (3,9,9) as input is preferred.

The network with input size (3,9,9) is chosen for both classification and regression purposes (fully connected, 4 layers, 32 neurons per layer), representing the intended deployment scenario for online productions. In the following analysis, the main focus will be put on the choice of different classification thresholds and their effect on the track reconstruction. Cutoff thresholds between 0.003 and 0.1 are chosen to cover cases of low (0.003, $< 1\%$) to high (0.1, 15%) cluster reductions.

At first, it is important to understand which tracks are mainly rejected. Figure (3.4.20) therefore shows the number of tracks as a function of number of crossed rows with attached clusters. Particularly, long tracks should maintain their cluster attachments.

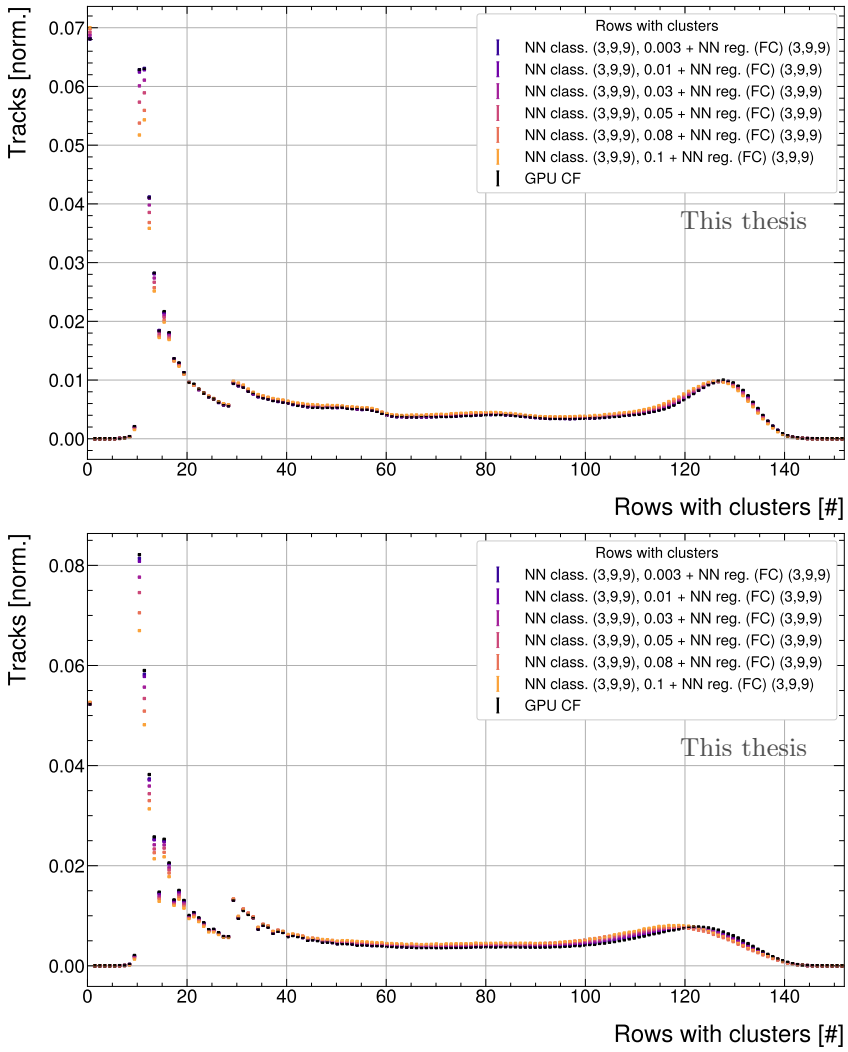


Fig. 3.4.20: Normalized number of tracks as a function of crossed rows with attached clusters for the minimum bias (top) and centrality enforced simulation (bottom). Each histogram is normalized to the number of tracks found for each case.

With increasing threshold, the highest rejection occurs for tracks with less than 20 crossed rows in both cases. Such tracks are irrelevant for physics analysis and will be discarded by standard track selections. Long tracks experience minor cluster reductions, leading to a shift of the peak (close to 128 crossed rows) at high number of crossed rows. The shift is up to 3 clusters (rows) per track for the centrality enforced simulation and 1 cluster (row) per track on the minimum bias simulation, even at the highest rejection threshold setting. Higher cluster overlaps in the centrality enforced simulation lead to higher "randomized" cluster rejections, including clusters of long tracks. Additionally, clusters with high overlap can decrease the quality of the track fit and the $\mathrm{d}E/\mathrm{d}x$ estimate, in which case their rejection is even beneficial for the physics performance. Furthermore, clusters with very low classification thresholds will typically not be included in the training dataset of the regression network. The regression network will therefore have never been trained to create accurate predictions for such clusters,

producing potentially incorrect results. This further emphasizes that it is necessary to reject a certain amount of clusters before the regression network is useful. Between the lowest (0.003) and highest (0.1) illustrated threshold, the total number of clusters (NCL) decreases by 15.6% with a 6.6% loss of tracks on the minimum bias simulation. The reduction in NCL per track and the reduction in total datasize are a trade-off that needs to be carefully considered when this algorithm is deployed. Many of such tracks are rejected because their NCL is low (short tracks). Physics analysis which utilize the specific energy loss for particle identification only considers tracks with NCL > 60 due to the otherwise unstable $\mathrm{d}E/\mathrm{d}x$ estimate. Very short tracks can therefore be rejected without loss in physics performance. A maximum threshold setting of 0.05-0.08 is recommended to keep clusters of long tracks. At such thresholds, the number of tracks with NCL > 60 is reduced by less than 1%.

The track reductions are investigated kinematically. Figure (3.4.21) shows the distribution of tracks on the minimum bias simulation in total numbers (top) and as a ratio to the default cluster finder (bottom). Both simulations exhibit a very similar behavior, hence only the minimum bias simulation is shown here.

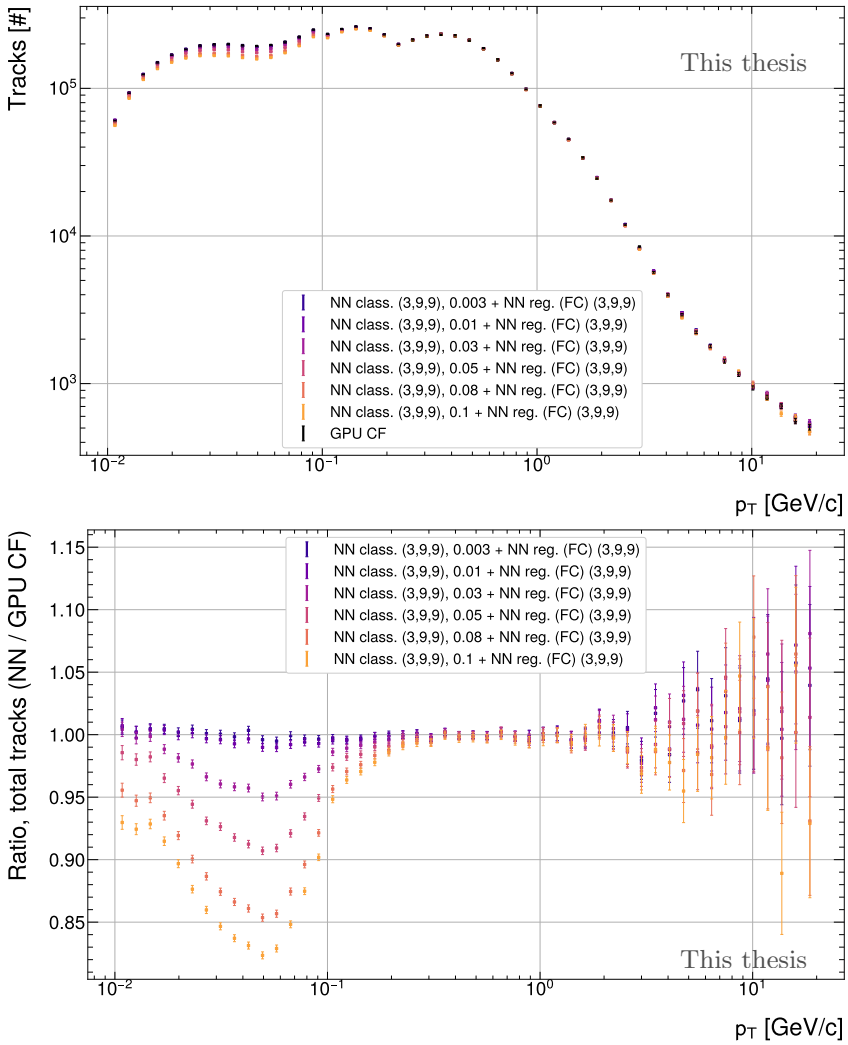
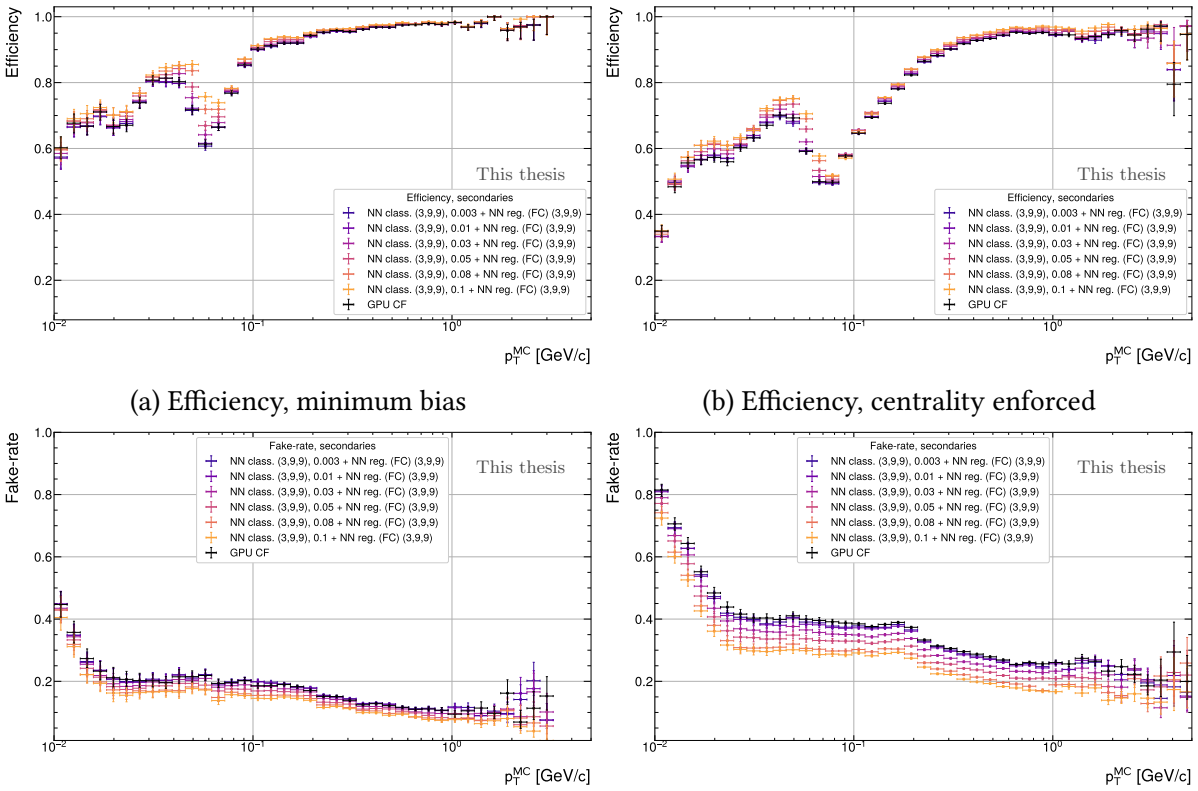


Fig. 3.4.21: Normalized number of tracks as a function of crossed rows with attached clusters for the minimum bias simulation in absolute yields (top) and as a ratio to the default cluster finder (bottom).

Local cluster rejections lead to a reduction in number of clusters per track, mainly noticeable for highly inclined parts. This leads to a reduction in total tracks up to the maximum looper momentum of $p_{\mathrm{T}} \leq 300$ MeV/c as shown in the ratio plot of figure (3.4.21). Major track reductions are noticeable between $10 \leq p_{\mathrm{T}} \leq 300$ MeV/c, fully within the range of such looping tracks. To further illustrate that such reductions are stemming from the reduction of fake and looper tracks and to show that good tracks are kept, the efficiency and fake rates of secondary particles are investigated. Figure (3.4.22) shows the tracking efficiency and fake rate of secondary particles for both simulations.



(a) Efficiency, minimum bias

(b) Efficiency, centrality enforced

(c) Fake-rate, minimum bias

(d) Fake-rate, centrality enforced

Fig. 3.4.22: Tracking efficiency (top) and fake rate (bottom) as a function of p_T for secondary particles for the minimum bias (left) and centrality enforced (right) simulation.

With increasing threshold, the efficiency of finding secondary particles improves while the total number of tracks decreases. Track reductions rather occur due to fake cluster / track rejections. This clearly illustrates that the reductions occur due to fake track removals. Towards higher p_T, the efficiencies saturate with only minor efficiency increases for the cases with high cluster rejection. Overall, the rejection delivers consistent improvements across the kinematic range. Similar plots for primary particles as a function of the pseudo-rapidity can be found in the appendix (C).

The resulting track quality is quantified using the $\chi^2_{red} = \chi^2/\text{NDF} = \chi^2/(2*\text{NCl} - 5)$ per track. To distinguish between the quality of good and fake tracks, the χ^2_{red} is shown in figure (3.4.23) for the minimum bias simulation as a separate plot between good tracks (top) and fake tracks (bottom).

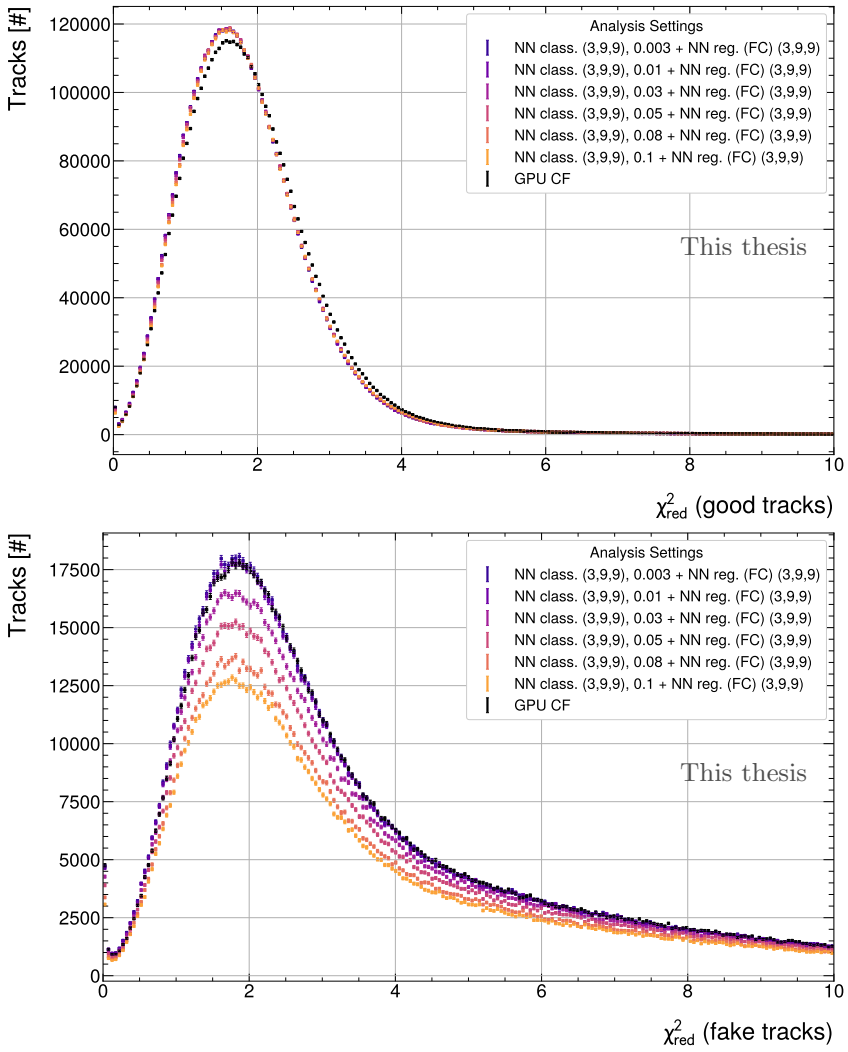


Fig. 3.4.23: χ^2/NDF distribution on the reconstructed tracks for good (top) and fake (bottom) tracks on the minimum bias simulation.

The dominant contribution to the full χ^2_{red} distribution, stemming from good tracks, shows a clear improvement, validating the regression capabilities on the cluster center of gravity until track reconstruction level. The fake track distribution shows a clear reduction in absolute number of tracks, while the peak of the χ^2_{red} distribution stays mainly unaffected.

As an impact on the final physics quality of tracks, the residual p_{T} and η resolutions are investigated. These properties will be investigated on the centrality enforced simulation to illustrate the effect of higher occupancies. Illustrated in figure (3.4.24) are the relative p_{T} and λ residuals and RMS resolution compared to MC truth after track reconstruction. The p_{T} resolution is shown as a function of p_{T} while the λ resolution is shown as a function of the pseudo-rapidity η. Figures (3.4.25) then show the ratio between the Root-Mean-Square error (RMS) of both algorithms as a ratio to the default reconstruction algorithm.

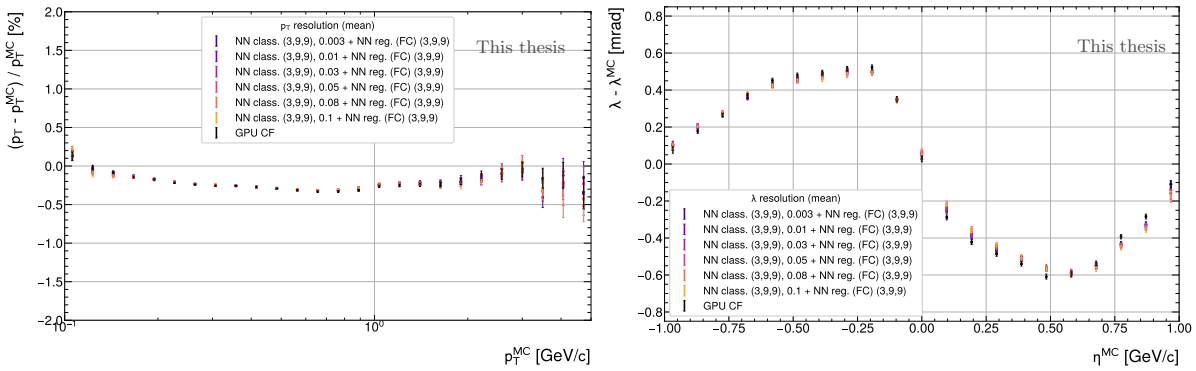


Fig. 3.4.24: p_{T} (left) and λ (right) mean resolution.

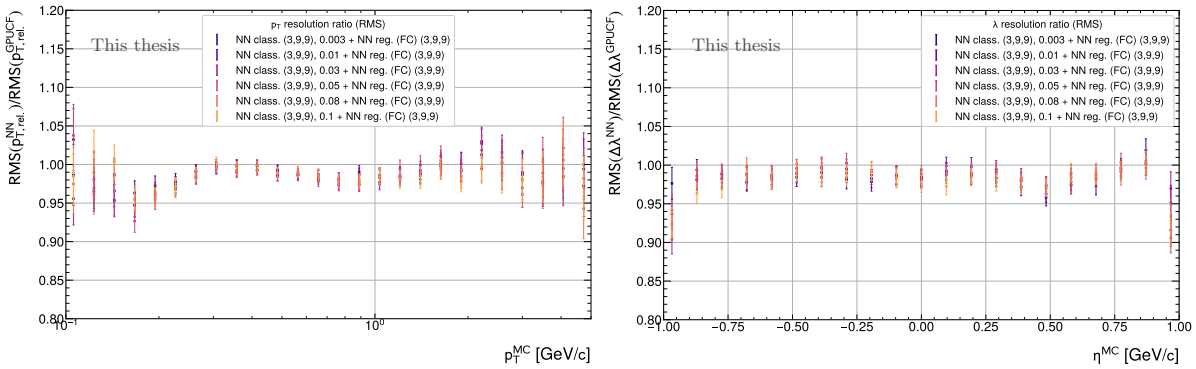


Fig. 3.4.25: p_{T} (left) and λ (right) RMS resolution of resulting tracks as a ratio to the default cluster finder.

All cluster finder implementations agree within the statistical uncertainty with the default reconstruction method. However, a systematic reduction in the RMS width is noticeable for any threshold setting. While the mean p_{T} resolution does not improve, the λ residuals improve towards the central region by up to 10% (excluding the $\eta = 0$ bin).

To conclude the investigation on Monte Carlo level, matching efficiencies between different detectors are investigated. Most notably, primary tracks passing the TPC with a sufficiently high number of cluster, chosen here as NCl ≥ 60 (as it is the typical physics analysis cut), will traverse and in most cases be reconstructed by the inner tracking system (ITS). Previously matched tracks should persist even in high rejection regimes. Investigated were primary tracks as a ratio to the total number of TPC matched tracks, as shown in figure (3.4.26). As it offers a comparison to a realistic scenario, the minimum bias simulation is used here.

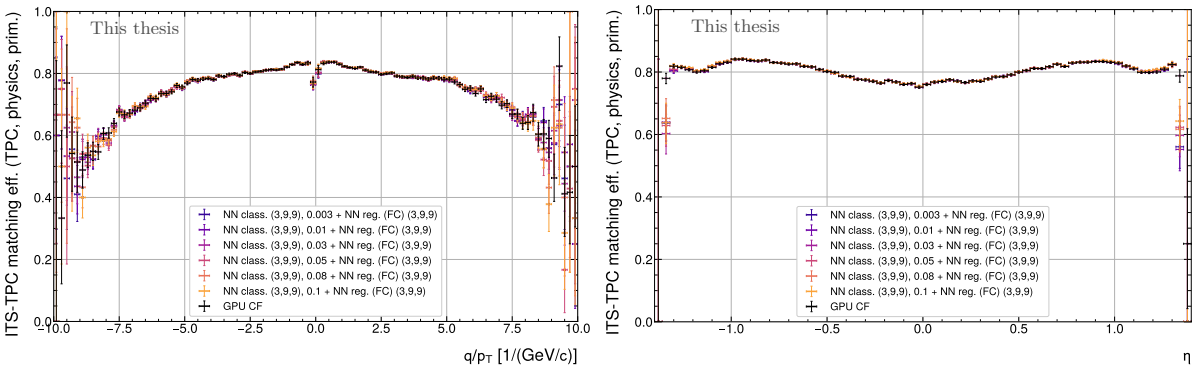


Fig. 3.4.26: ITS-TPC matching efficiency, normalized to the number of reconstructed TPC tracks for primary tracks with NCl $\geq$ 60 as a function of q/p_{T} (left) and η (right) for the minimum bias simulation.

For all rejection thresholds the matching efficiency is maintained, except for very high $|\eta|$ tracks, which are however not used in physics analysis (typical cut: $|\eta| < 1$). The inefficiencies are considered to stem from the used detector calibrations, which were not tuned for the neural network cluster finder and are subject of further studies.

As the last part for the investigation on track reconstruction level using Monte Carlo data, the DCA_r resolution of extrapolated TPC tracks is investigated. It describes the distance of closest approach of TPC tracks (normalized to the number of reconstructed TPC tracks) to the primary vertex associated with the collision, shown in figure (3.4.27).

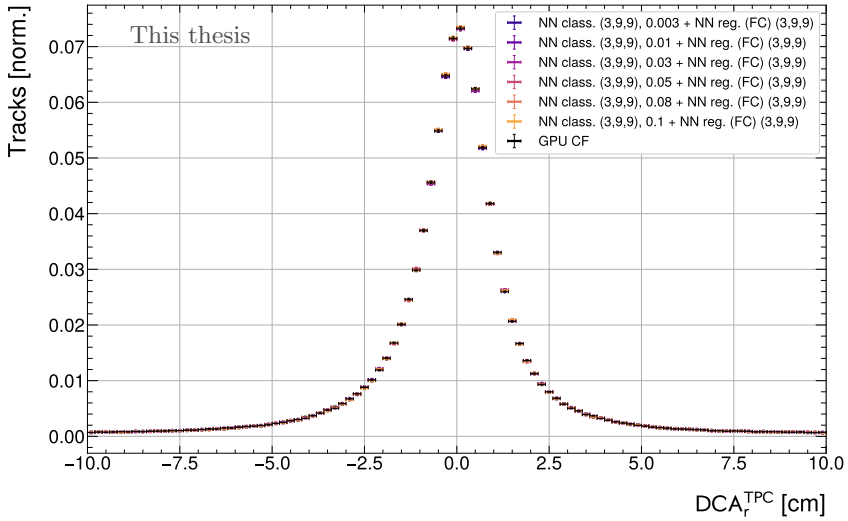


Fig. 3.4.27: Normalized $\mathrm{DCA}_{\mathrm{r}}^{\mathrm{TPC}}$ resolution for the minimum bias simulation.

The distributions match within the statistical uncertainty, showing that the track and primary vertex resolution is maintained. Minor track reductions are found towards the central peak of the distribution stemming from inefficiencies and removed tracks at very high $|\eta|$.

All aforementioned "standard candles" of the reconstruction process are addressed and show maintained or improved performance. It was demonstrated that tracking efficiencies are improved by up to 5%, while fake rates are reduced by up 3% (absolute numbers) on the minimum bias sample for secondary particles, corresponding to 7% and 10% in relative numbers. Major reduction of fake tracks were illustrated within the $0.01 < p_{\mathrm{T}} < 0.3$ GeV/c range, while maintaining the number of good tracks with improved χ^2/NDF distributions by up to 0.13 units for the good tracks. The reduction in number of tracks was found to stem from short tracks which do not influence physics analysis. Physics quantities, such as the angular (η, λ) and p_{T} resolution are maintained. This makes all algorithmic approaches suitable for physics analysis from the Monte Carlo point of view, with general performance increases and data savings using the neural network approach.

3.5 Extensions of this work

Due to the flexibility of the neural network approach, many extensions of this work can be considered. In the following, several interesting outlooks are highlighted, which require further studies to be applicable in the actual reconstruction chain.

3.5.1 Training on real data

In order to tune the neural network for cluster finding, it would be desirable to train it on real data. In the absence of Monte Carlo labels, one option considered was to train the network on residuals between digit maxima and extrapolated track positions. However, this approach risks training the network to correct for space-charge distortions, which cannot be reliably corrected on the local charge input. Implementing such a training scheme would require extrapolating tracks to the crossing point of a given pad row, transforming the z to time coordinate. This requires applying inverse distortion correction maps to bring the track position into the vicinity of the original charge maximum, constituting a substantial technical challenge. More importantly, any residual, uncorrected distortions would bias the cluster positions learned by the network. This bias is expected to compound over successive iterations: once a newly trained cluster finder network is deployed and used to retune distortion maps, the updated maps would inherit the bias introduced by the previous network. For these reasons, the more principled approach adopted throughout this thesis (training on MC data) was deemed to offer a higher probability of success, while reducing the risk of accumulating systematic biases and human-induced errors.

3.5.2 Momentum vector estimation

As a concluding subject to this investigation, a novel feature of the NN cluster finder algorithm is investigated. This feature concerns the estimation of a momentum vector on digit-level information. This feature is particularly useful for track seeding. As described in a previous chapter (I), TPC tracks are reconstructed by connecting an initial set of seed clusters across the TPC sector. This connection between clusters can benefit from the approximated information of a direction vector. This vector can be provided by the network, utilizing the three-dimensional charge input. The network is therefore trained to predict the inclination components p_Y/p_X and p_Z/p_X. These components are extracted for each cluster from the matched tracks after the reconstruction process. They directly relate to the local ϕ inclination angle by $\phi = \arctan(p_X/p_Y)$ and θ inclination angle by $\theta = \arctan(p_Z/p_X)$. Conversions between the global and local coordinate systems are then taken care in the reconstruction framework. Figure (3.5.28) compares the residual ϕ inclination predicted by the network with the inclination extracted after the track reconstruction ($\Delta\phi = \phi_{NN} - \phi_{rec}$) for three different neural networks.

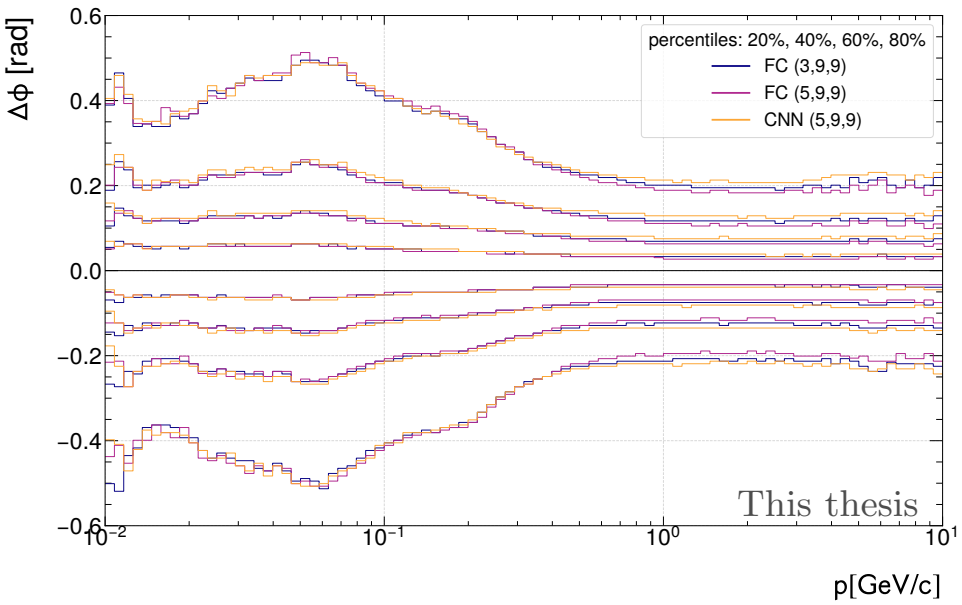


Fig. 3.5.28: Percentiles of residual ϕ distribution between the network prediction and the reconstructed track inclination ($\Delta\phi$) as a function of track momentum for three different network types.

With increasing track momentum the distribution significantly decreases in width, indicating a lower uncertainty of the network estimation in comparison to the true track inclination. This effect is explained, as the proportion of transverse momentum (in general) increases with higher track momentum, leading to a significantly lower curvature radius ($r \sim p_T$). This in turn leads to ϕ values closer to 0 radians, resulting in smaller absolute values that the network has to predict. At higher local inclinations, even small variations of the digit charges in pad direction in adjacent rows of the three-dimensional input can significantly influence the

inclination angle determination. This effect increases in magnitude for particles of low transverse momentum $p_T < 0.5$ GeV/c. The network overall achieves a resolution to 0.2 radians for high momentum tracks within the 80% percentiles, which increases to 0.4 radians at 0.1 GeV/c. All networks perform on par with each other. The performance benefits from an increased input charge window ((5,9,9) compared to (3,9,9)) are considered marginal. Given the increase in compute time required to provide the charge window to the network, the (3,9,9) input is considered sufficient. The fully connected networks perform overall slightly better than the convolutional network, particularly noticeable at high momentum for the 60 and 80% percentiles. The same study is repeated for the θ angle inclination and shown in figure (3.5.29).

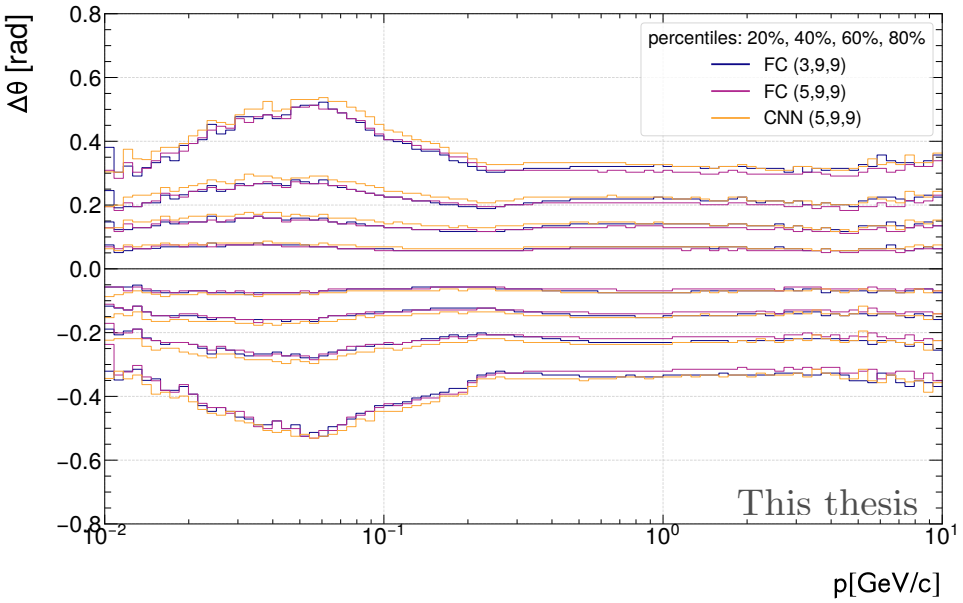


Fig. 3.5.29: Percentiles of residual θ distribution between the network prediction and the reconstructed track inclination ($\Delta\theta$) as a function of track momentum for three different network types.

A similar conclusion concerning the network input charge array ((3,9,9) vs. (5,9,9)) and the type of network (FC, CNN) is drawn as for the ϕ inclination. Contrary to the case of the ϕ inclination, the improvement between the CNN and the FC network with the same input charge window is most noticeable for low momentum tracks (< 0.5 GeV/c). Overall it is concluded that the network with an input charge window of (3,9,9) is sufficient for the task of determining the local inclination, with only minor improvements found for the (5,9,9) case. Additionally, it is concluded that fully connected networks are preferable over convolutional architectures for this task.

The final influence on tracking needs to be validated using the reconstruction framework, however the implementation of this algorithm in the track seeding step requires significant changes in the cluster storage and tracking algorithm and exceeds the scope of this thesis. It

has the potential to be a beneficial factor that comes with (almost) no computational cost at online reconstruction time and should be taken into consideration for the operation in Run 4.

3.5.3 Application to reconstructed clusters

The classification of clusters is not limited to the application on digit level. A feasibility study for the application on reconstructed cluster level is conducted. The focus here is the application for offline cluster rejection and further cleanup of the tracking environment and reduction in datasize. The resulting center-of-gravity positions are rounded and read in as a three-dimensional grid to a classification neural network, specifically trained to receive such inputs. A comparison between the digit level input and a cluster level input is shown in figure (3.5.30).

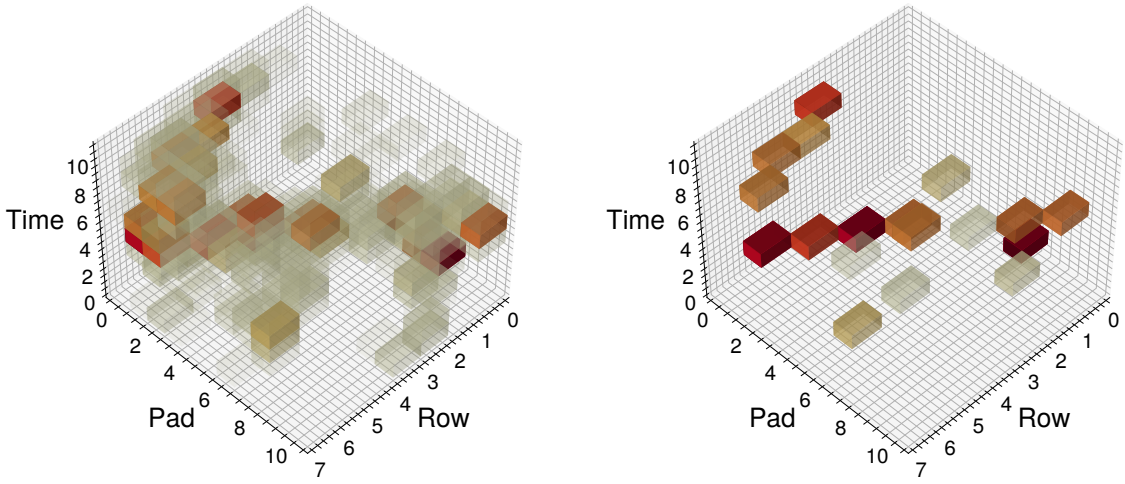


Fig. 3.5.30: Illustration of the input using digits (left) and clusters (right) with a (7,11,11) window.

This highlights that significantly cleaner input data is passed to the network in case cluster level information is used. The floating point precision of the cluster positions is not passed to the network. While a window size of (7,11,11) is chosen for illustration purposes, the window size for the neural network (e.g. (3,9,9)) is not increased, to reduce computational throughput. Additional studies would need to be conducted to investigate the optimal input grid size. While (3,9,9) results visually in satisfactory results, smaller input grid sizes could be considered for this application as the environment is significantly less noisy. Figure (3.5.31) shows the class label assigned to clusters after the application of a fully connected network with a (3,9,9) charge input.

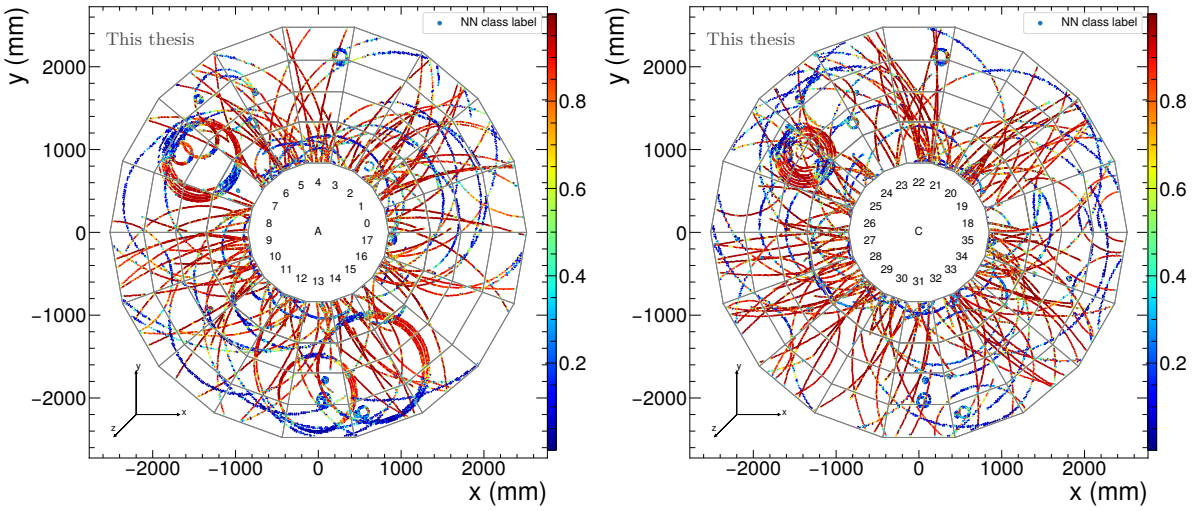


Fig. 3.5.31: Application of the classification neural network to the cluster level input on the TPC A and C side and illustration of resulting class label assigned to each cluster using a fully connected network with a (3,9,9) charge input.

While this application shows promising results for the reduction of high inclination clusters, it exceeds the scope of application in the online system (e.g. due to a secondary step of input filling on native cluster level). Further studies are required to validate this algorithmic approach and tune the rejection threshold. However, promising results for the reduction of datasize, in similar quantities as for the online application are expected and could be achieved on all (offline) stored data.

3.5.4 Cluster split-flag setting

At cluster finding time, split flags are set in order to increase the cluster error for later cluster-to-track associations. Cluster errors approximate the uncertainty used in the Kalman filter update step of the covariance matrix and for cluster-to-track attachments. The cluster error is determined on-the-fly during track reconstruction and cannot make use of the digit level information. This is a trade-off between physics and compute performance as the cluster error would increase storage of raw data by approximately 20%. Therefore, only the flag setting is a feasible alternative that can be set by the neural network. To disentangle the effects of setting the flags from the classification and regression networks, a third network is trained. Three different training methods were investigated to mark a digit maximum as split

1. If more than one peak is in the surrounding 5×5 neighborhood
2. Using a set of reconstructed clusters, if at least one cluster is in the surrounding 5×5 neighborhood that is marked split
3. Using a set of reconstructed clusters, if at least two cluster is in the surrounding 5×5 neighborhood that is marked split

The first strategy closely aligns with the current algorithm of setting a split flag. However, the neural network approach offers the flexibility of choosing a threshold (between 0 and 1) for the flag setting. Strategy two and three did not yield any significant improvements, therefore the focus will be put on strategy one. The performance is studied on the centrality enforced Pb–Pb simulation. Figure (3.5.32) shows the efficiency and fake rate ratio relative to the default, heuristic flag setting algorithm of primary particles as a function of the MC p_{T}.

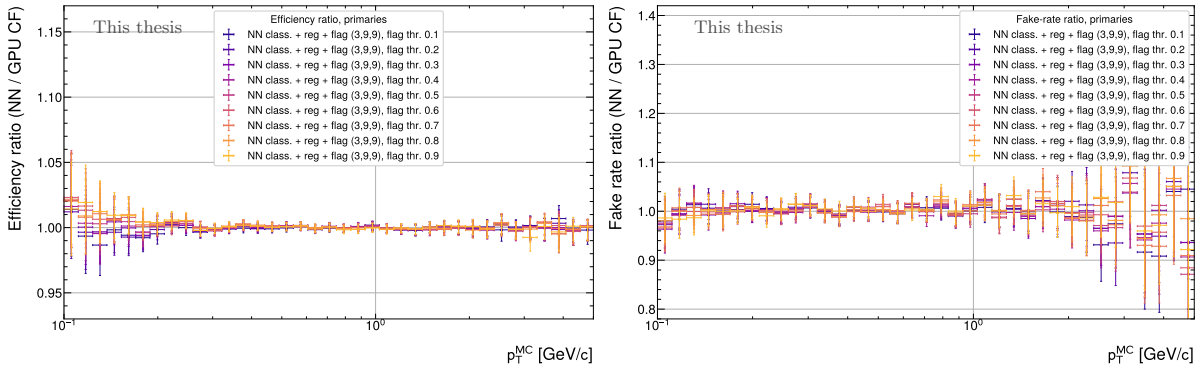


Fig. 3.5.32: Efficiency (left) and fake rate (right) ratio of primary particles as a function of the MC transverse momentum for different threshold settings of the split flag neural network.

For primary particles and for all chosen thresholds, the efficiencies and fake-rates of primary particles are compatible. No improvement or performance regression can be observed for any threshold setting. The same ratios are also illustrated for secondary particles in fig. (3.5.33).

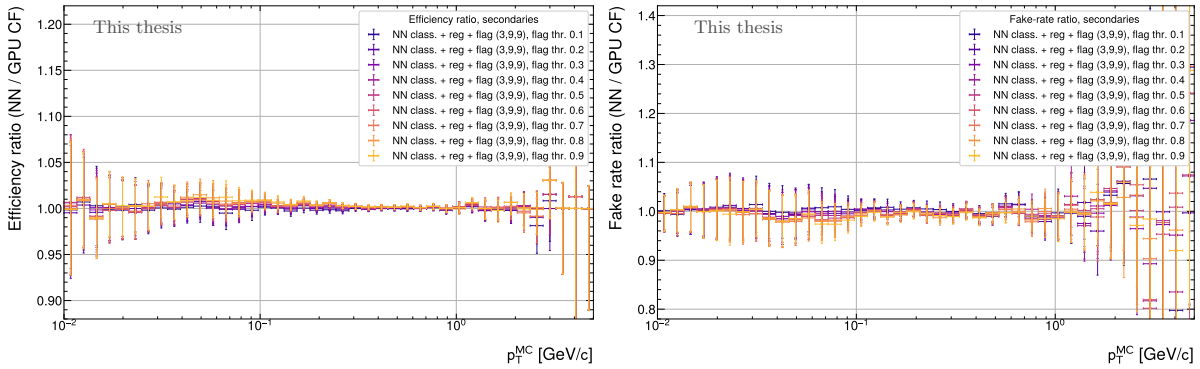


Fig. 3.5.33: Efficiency (left) and fake-rate (right) ratio of secondary particles as a function of the MC transverse momentum for different threshold settings of the split flag neural network.

A systematic, but only minor improvement is noticeable for $0.04 \leq p_{\mathrm{T}} \leq 1$ GeV/c for a threshold setting of ≥ 0.5. The effect of different cluster flag settings is overall found to be at maximum 2%. All efficiencies and fake-rates are compatible with the current flag setting algorithm within the uncertainty. A more noticeable change can be found on the χ^2/NDF for good tracks and fake tracks shown in fig. (3.5.34).

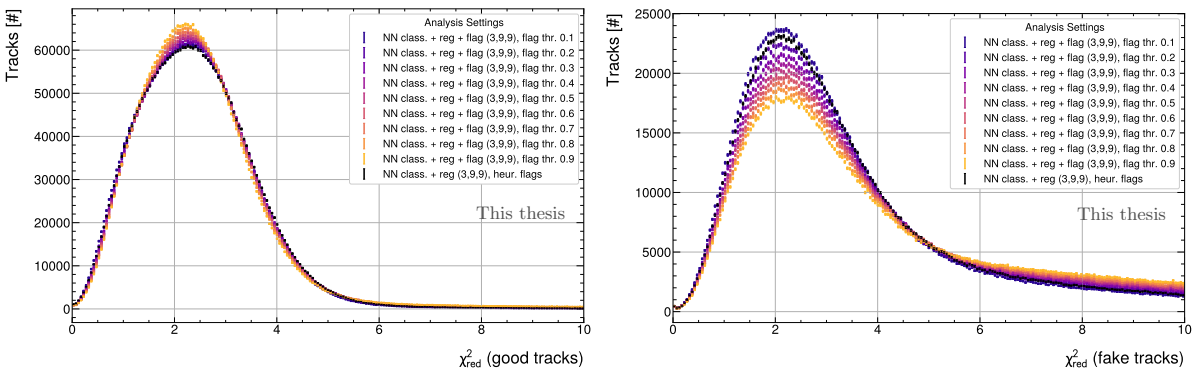


Fig. 3.5.34: χ^2/NDF for good tracks (left) and fake tracks (right) for different threshold settings of the split flag neural network.

While it is noticeable that the number of good tracks with low χ^2/NDF increase, the number of good tracks with high χ^2/NDF also increase. In the intermediate range of $2.3 \leq \chi^2/\text{NDF} \leq 5.2$ the number of tracks is reduced. The increase in the number of tracks for low values of χ^2/NDF stems from long tracks, where a stronger threshold setting results in less clusters receiving the split flag, leading to smaller cluster errors and ultimately less cluster attachment of clusters far away from the track tube. The increase in large values of χ^2/NDF is a result of many tracks being split into two tracks. This leads to shorter tracks with a significant increase at NCL < 60 for thresholds greater than 0.7. This is illustrated in figure (3.5.35) on the distribution of number of tracks as a function of number of crossed rows and the clone rate of primary particles as a function of MC p_T.

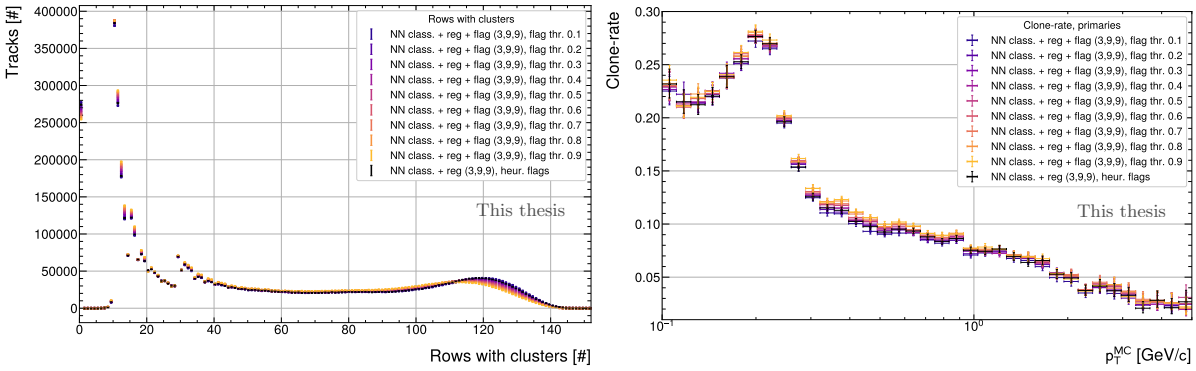


Fig. 3.5.35: Number of tracks as a function of crossed TPC rows (left) and clone-rate of primary particles (right) for different threshold settings of the split flag neural network and the heuristic flag setting method.

Overall it is concluded that no significant improvement can be brought by setting the cluster flags without further information content or very advanced approaches, exceeding the scope of these studies. Aside the demonstrated effects, other calibrations (such as the space-charge distortion calibrations) are not tuned on this new flag setting, which could have an impact on the reconstruction performance.

3.6 Prelude on the analysis on real data

The next chapter (IV) will concern the analysis on real data. As an introduction to this topic, initial settings of the reconstruction are tested between the Monte Carlo simulated and real data. One setting in the reconstruction process required significant modification: the q_{tot} noise threshold setting. This setting mainly influences which clusters are removed as noise clusters after the cluster reconstruction step, given their total charge. Due to the higher input charge grid of the network, the total charge of noise clusters (particularly with overlap) is expected to be higher than the default value of $q_{\mathrm{tot}}^{\mathrm{noise}} \leq 5$. While it is possible to tune this threshold on simulated data, a more robust way (independent of simulation effects) is the comparison on real data. The threshold is tuned, such that a similar number of clusters and tracks, compared to the current reconstruction algorithm, is found. No cluster rejection by the classification network is performed, only the regression network is applied for this task. The initial evaluation on real data shows that a significantly higher noise threshold needs to be selected. Data from 10, 32-orbit timeframes is used to tune the q_{tot}-threshold. Both pp (1 Mhz interaction rate) and Pb–Pb data (38 kHz interaction rate) are investigated. The result is presented in figure (3.6.36).

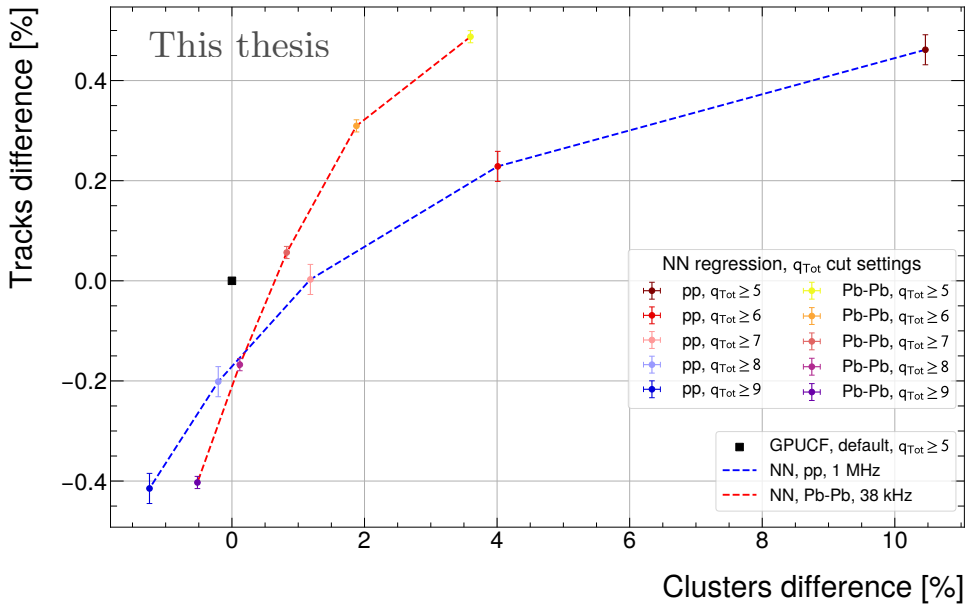


Fig. 3.6.36: Relative difference of number of clusters and number of tracks to the default cluster finding algorithm for different q_{Tot} noise thresholds.

By proximity of the produced clusters and tracks to the standard reconstruction both 7 and 8 are considered valid thresholds for the noise rejection on the pp dataset. Due to closer proximity in the cluster difference, a threshold of 8 is chosen. This amounts to nearly identical number of clusters and a track reduction by $\approx 0.19\%$ which is considered acceptable for the tuning of this parameter, given that downstream calibration objects are tuned on the default cluster finding algorithm. The setting of $q_{\mathrm{Tot}} \geq 7$ or 8 for the total charge threshold also re-

mains valid for the Pb–Pb data. The increased noise threshold is thus a direct consequence of detector noise, largely independent of the increased local occupancy. For all further studies on real data, a setting of 8 will be used.

IV Particle identification and analysis on real data

Previous works have successfully deployed neural networks for the calibration of the PID response by correcting the mean and estimating the band width of the $\mathrm{d}E/\mathrm{d}x$-distributions [45]. The introduction and method used in this thesis strongly align with this work [45], [46]. In light of the newly developed algorithmic approach to the cluster reconstruction process, this chapter illustrates the effect on particle identification - a central component of offline physics analysis and a key element of collaboration-wide data production. In this sense, it completes the reconstruction chain from raw detector signals to calibrated observables.
Unless stated otherwise, the investigated dataset will be the dataset corresponding to the stored raw data from the LHC24ar period (Pb–Pb, 30-38 kHz, apass2). This period also serves as the reference for the simulated data from the previous chapter III. The total volume of processed raw data amounts to approximately 0.01% (8.8×10^5 processed events, 7.2×10^5 events after standard event selection cuts) of the total statistic collected in this period.

4.1 Practical introduction to particle identification

Particle identification aims at assigning a mass hypothesis to a track given the measurements of one or multiple detector systems. Each detector contributes complementary information to the overall characterization of a particle. Detectors close to the interaction vertex require high spatial resolution, while minimizing multiple scattering and energy losses within the material. Since track occupancies increase as $1/r$ towards to the primary vertex, high granularity and resolution are crucial for track and vertex reconstruction. Reducing the material budget, measured in units of the radiation length X_0 (the length in which a particle loses e^{-1} of its energy) at inner radii, is therefore a central challenge within high energy physics experiments. Silicon wafers achieve much greater precision without distortion effects, whereas gaseous detectors can achieve lower material budgets. The continuous ionization of gas molecules further enables a measurement of the particles' energy loss per unit distance, the $\mathrm{d}E/\mathrm{d}x$, as illustrated in the introduction of this thesis (chapter I). A unique feature of the ALICE TPC is its gaseous volume of 88 m^3, making it the largest TPC every operated in the field of high energy physics. It provides up to 152 (159) individual $\mathrm{d}E/\mathrm{d}x$ measurements per track in Run 3 (2) of LHC.

To first order, the accumulation of the final track $\mathrm{d}E/\mathrm{d}x$ then follows the characteristic shape, the Bethe-Bloch curve (see eq. 1.1.1). As the functional shape of the Bethe-Bloch formula only depends on the $\beta\gamma$ value of the track, the mass hypothesis can be inferred from the momentum p (being an experimental observable) and the associated $\mathrm{d}E/\mathrm{d}x$ band. This constitutes the basis for particle identification using the TPC.
However, the precision of this identification depends on the detector performance in various regions of the kinematic phase space. While the Bethe-Bloch curve describes the mean value of each $\mathrm{d}E/\mathrm{d}x$ band to $O(1\%)$ level precision, physics analysis typically requires a calibration to an accuracy of $O(1‰)$. Such accuracy can only be achieved using secondary corrections, extending beyond a one-dimensional function. Moreover, the bandwidth of the $\mathrm{d}E/\mathrm{d}x$ signal for each mass hypothesis varies across the kinematic range and geometry of tracks. A quantification of both values (mean correction factor and bandwidth) allows to assign a mass hypothesis using $N\sigma$ cuts on the corrected spectra as

$$N = \frac{\mathrm{d}E/\mathrm{d}x_{\mathrm{TPC}} - \mathrm{d}E/\mathrm{d}x_{\mathrm{exp.}}}{\sigma}. \tag{4.1.1}$$

This is a well established way of separating tracks useful for a given physics analysis, by choosing the value of N for acceptance of a particle mass hypothesis. Both inclusive and exclusive cuts can be applied, where a mass hypothesis of a track is considered valid if its value is less than N (inclusive) or where a mass hypothesis is rejected if the value is larger than N (exclusive, veto). Inclusive cuts are most common and will be used throughout this thesis. The fact that the Bethe-Bloch curve only depends on $\beta\gamma$ leads to a horizontal shift of bands between different mass hypothesis when considering $\mathrm{d}E/\mathrm{d}x$ as a function of momentum (as this is the actual experimental observable). This is illustrated in figure (4.1.1). The figure shows the measured $\mathrm{d}E/\mathrm{d}x$ distribution as a function of rigidity p/Z, where Z is the particles charge. For all detectable particles until the proton mass, $|Z| = 1$ holds.

The resulting bands show clear separations in different ranges of the track momentum. One of the main quantities used for the assessment of reconstruction quality is the electron-pion separation at the minimum ionizing region of pions, between $3 < \beta\gamma < 4$ (pion mass hypothesis, $m_\pi = 139.6$ MeV/c: $0.42 < p < 0.56$ GeV/c $\rightarrow$ typical chosen range: $0.45 < p < 0.55$ GeV/c). Pions in this region are referred to as minimum ionizing particles (MIP). MIP pions, release the least amount of energy when traversing the TPC gaseous volume, while electrons at the same momentum already approach the Fermi plateau. Within this momentum range, both bands reach near constant mean values in $\mathrm{d}E/\mathrm{d}x$ with (near) Gaussian bands, making this region an excellent candidate to investigate the separation power. At this kinematic range the intrinsic resolution of the detector can be quantified. At low occupancies, the nominal $\mathrm{d}E/\mathrm{d}x$ resolution of the ALICE TPC is $\approx 7\%$ of the measured $\mathrm{d}E/\mathrm{d}x$ signal [12]. Secondary corrections refine the expected $\mathrm{d}E/\mathrm{d}x$ values and bandwidths over the multidimensional phase space spanned by p,

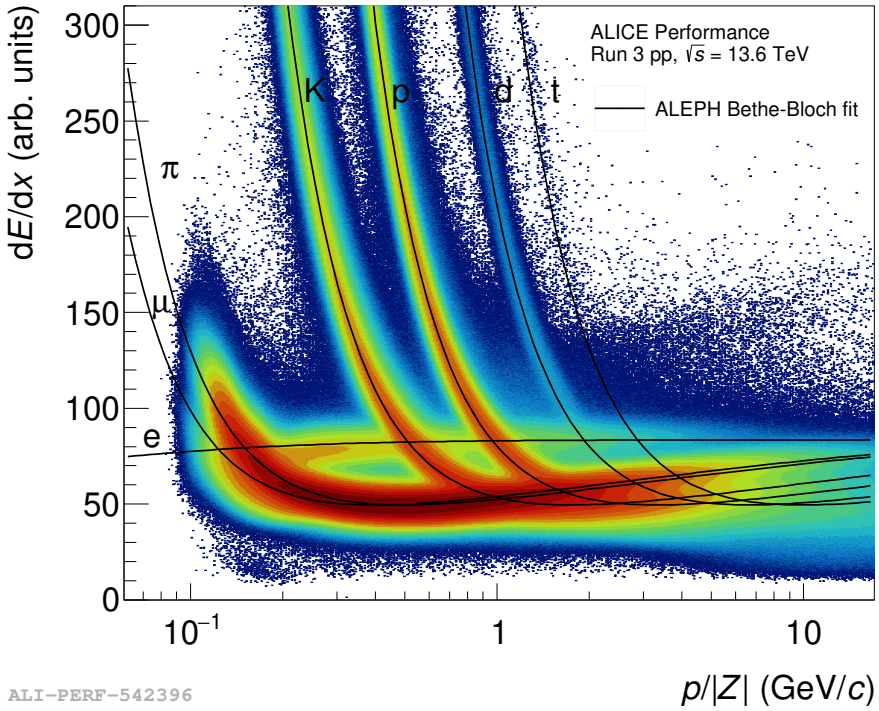


Fig. 4.1.1: Particle bands and fitted Bethe-Bloch curves against the rigidity ($p/|Z|$, where Z is the particles charge). Dataset: LHC22q, apass3, 1×10^9 events.

p_{T}, η, occupancy, and the number of TPC clusters. The derivation of optimal Bethe-Bloch parameters and the tuning of secondary corrections will be the goal of this chapter.

4.2 Clean sample selection and calibration of the Bethe-Bloch parameterization

The Bethe-Bloch curve delivers a theoretical prediction for the average energy loss of particles traversing the detector gas. However, the limited data points available after the measurement process (maximum 152) and the Landau-distributed $\mathrm{d}E/\mathrm{d}x$ signal limit the accuracy of the extractable signal. Especially for the TPC, this fact is the main reason for the use of the truncated mean estimation of the track $\mathrm{d}E/\mathrm{d}x$ from the individual clusters, as illustrated in the introduction (I). The necessity of using of the truncated mean estimation makes the direct application of the theoretical Bethe-Bloch function unfeasible. A tunable parameterization is chosen for practical purposes. The functional shape of choice is inspired by the ALEPH parametrization [47], written as a function of $\beta\gamma$, the charge z and the parameters $a_1, ..., a_5, f_z$ (the charge factor), f_{MIP} (the mean ($\mathrm{d}E/\mathrm{d}x$)-value of the minimally ionizing pions).

$$\langle \mathrm{d}E/\mathrm{d}x \rangle_{\mathrm{ALEPH}} = a_1 \cdot (a_2 - \log(a_3 + (\beta\gamma)^{-a_5})/\beta^{a_4} - 1) \cdot z^{f_z} \cdot f_{\mathrm{MIP}} \tag{4.2.1}$$

It is therefore the first step of this calibration to identify a set of particles for which the particle identity is known. The tuning of the parameters of this function is then done in the two-dimensional ($\mathrm{d}E/\mathrm{d}x$, $\beta\gamma$) plane.

4.2.1 Particle identification on V0 decays and γ conversions

To calibrate the corrections, both Monte Carlo data and real data can be considered as a feasible approach. While MC has the advantage of a priori available PID information, real data steers the calibration to the best possible approximation of corrections, as the training data is a direct reflection (subset) of the analysis data. However, in order to use real data for the training process, a clean set of tracks with known particle masses needs to be constructed. Two possibilities can be chosen for this process. The first one being tracks in kinematic ranges, in which one or multiple detectors show clear separation between the particle species (e.g. low momentum kaons and protons) without the need for secondary corrections. The second being geometrically identifiable decays and conversions of

- $K^0_S \longrightarrow \pi^+\pi^-$
- $\Lambda \longrightarrow \pi^- p$, $\bar{\Lambda} \longrightarrow \pi^+ \bar{p}$
- $\gamma + Z \longrightarrow e^+e^- + Z$ (photon: γ + massive scattering partner particle: Z).

These processes occur in abundance throughout the data-taking period and therefore accurately reflect the operational conditions. Furthermore, the resulting daughter tracks (e, π, p) already cover three of the four most dominant particle species in the measured spectrum, with only kaons (K) missing. Such a decay with a selection criterion on the DCA (distance of closest approach) of the decay is schematically shown in figure (4.2.2).

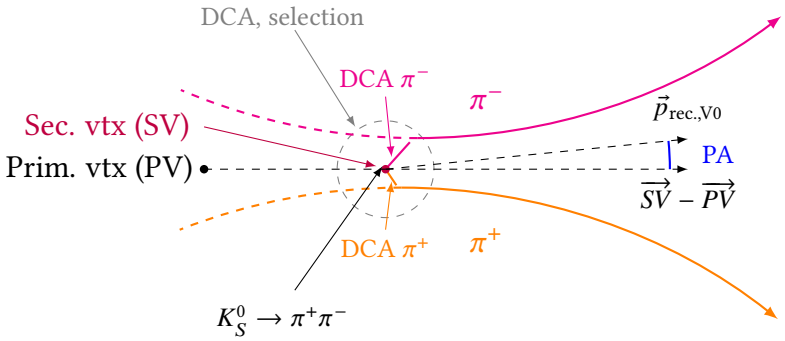


Fig. 4.2.2: Schematic view of the V-shaped topology of a reconstructed $K^0_S \longrightarrow \pi^+\pi^-$ decay with DCA selection criterion and pointing angle (PA) on the reconstructed daughter tracks.

While only charged particles release electrons (and ions) when interacting with the molecules of the detector gas, the charge-neutral mother particles traverse without electromagnetic in-

teractions. However, reconstructing the characteristic topology using the daughter particles can easily identify such decays, leading to particle identification without requiring highly granular calibrations. Since the mass of the mother particles is accurately known from previous measurements (K^0_S : 497.611 ± 0.013 MeV/c^2, Λ : 1115.683 ± 0.006 MeV/c^2)[1], it is possible to reconstruct the invariant mass of the mother particle from the four-momentum vectors of the daughter particles. It further quantifies the cleanliness of the applied selection and accuracy of the reconstruction process. Selections on the daughter tracks and the topology of the decay significantly reduce noise. While low-momentum protons already populate the low-$\beta\gamma$ range of the dE/dx spectrum (up to $\beta\gamma \approx 1$), electrons from γ decays reach into $\beta\gamma \approx 10^4$, covering the Fermi plateau until saturation. Pions contribute with a major dominance in the minimum ionization region and cover a kinematic range of $0.1 < \beta\gamma < 100$. This illustrates that it is feasible to extract the calibration in a data driven approach, as the full momentum range is covered. Kaons are additionally added from low momentum TPC + TOF selections, where particle bands are well separated.

The selection process for the daughter tracks is empirically driven. Topologically, the reconstructed angle between the momentum vector of the mother particle originating from the primary interaction and the vector connecting the primary (collision) and secondary (decay) vertex should coincide. The precision is limited by the detector resolution, accuracy of track reconstruction and combinatorial background (vertex creation of two uncorrelated particles merely by chance). Typically, the cosine of this angle, cos(PA) (PA = point angle), is one of the variables of choice, used to remove incorrectly reconstructed secondary vertices. A cos(PA) > 0.999 is typically required in physics analysis. To improve the clean selections for the PID calibration, a value of 0.9997 is used.
Due to four-momentum conservation, the daughter tracks exhibit an elliptic behavior in the plane spanned by $q_\mathrm{T} = p_\mathrm{T}$ and $\alpha = (p^+_\mathrm{L} - p^-_\mathrm{L})/(p^+_\mathrm{L} + p^-_\mathrm{L})$ (where + (−) indicates the positively (negatively) charged daughter tracks and L (T) the longitudinal (transverse) components) [48]. The characteristic shapes are then used to improve the clean selections. Figure (4.2.3) shows the Armenteros-Podolanski plot [48] in the (α,q_T) coordinate space, where the reconstructed mother particles exhibit the characteristic behavior.
Additional selections on detector measurements, such as the TPC dE/dx, the TPC track χ^2 and the TOF β allow further removal of contamination, even with wide cut values such as 5σ acceptance on the respective signal. For such initial selections, percent-level accuracy of the calibration is sufficient. The detailed list of selections are found in the appendix (table (C.1.2)). The selected daughter particles reveal the reconstruction quality and are representative of all data-taking conditions encountered during the run.

[1]Taken from https://pdglive.lbl.gov

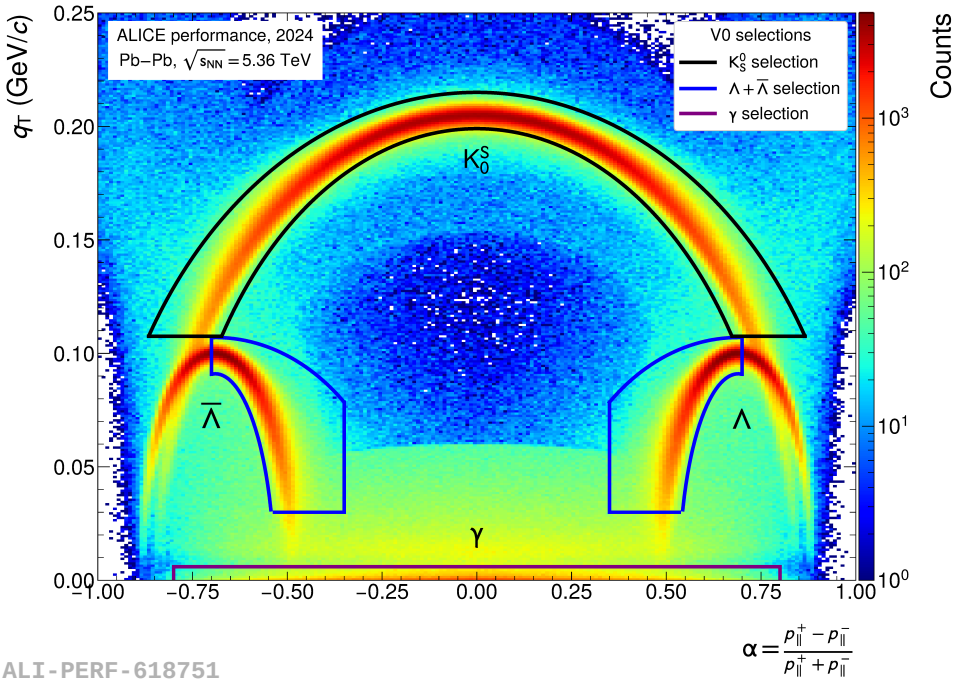


Fig. 4.2.3: Illustration of the selection criteria applied on the Armenteros-Podolanski variables for clean V^0 selection.

4.2.2 $\mathrm{d}E/\mathrm{d}x$ resolution and separation power

To investigate how the novel cluster finding algorithm affects the PID quality, the $\mathrm{d}E/\mathrm{d}x$ distribution is investigated. The bandwidth of the $\mathrm{d}E/\mathrm{d}x$ signal is a key component and central to the track selections performed at analysis level. The width of the $\mathrm{d}E/\mathrm{d}x$ bands is an intrinsic measurement of the TPC and is independent of secondary post corrections, applied at analysis time. It therefore signifies improvements or changes in the detector and reconstruction quality. A key measure for quality assurance is the measurement of the separation power between the $\mathrm{d}E/\mathrm{d}x$ bands of two particle species. With the aforementioned separation between pions and electrons, the minimum ionizing region ($0.45\ \mathrm{GeV}/c \leq p \leq 0.55\ \mathrm{GeV}/c$) is best suited to investigate the separation power. The separation power is calculated by the formula

$$S = \frac{|\mu_1 - \mu_2|}{0.5 \cdot (\sigma_1 + \sigma_2)}, \tag{4.2.2}$$

where it is assumed that both bands are well described by a Gaussian distribution. In the investigated Pb–Pb dataset, the pion band dominates the statistical yield of electrons to such an extent, that a double Gaussian fit does not converge to a sensible result, on the full distribution of tracks. Rather, the previously explained set of cleanly selected V0 samples are used to calculate the separation, since single Gaussian fits can be performed per particle species (using their assigned identity after selection). Figure (4.2.4) shows the two cleanly selected V0 samples for electrons and pions using the heuristic cluster finding algorithm.

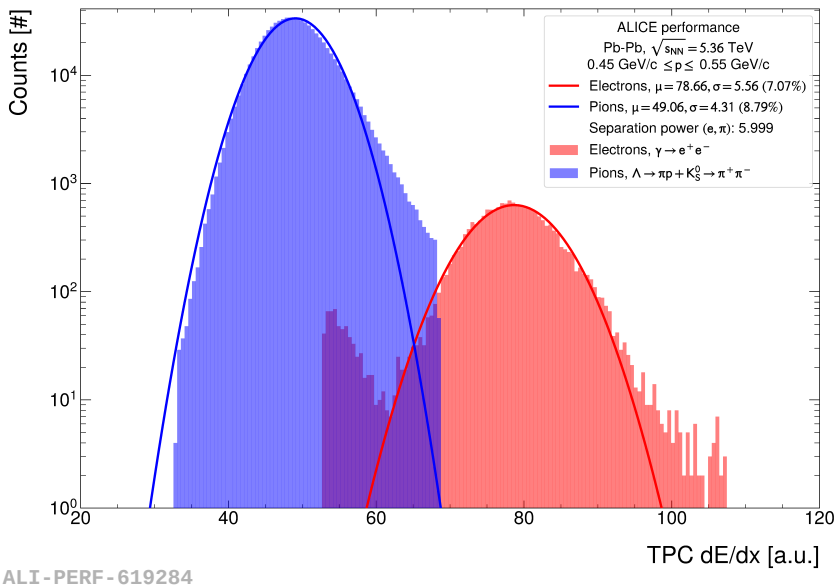


Fig. 4.2.4: Electron-pion band separation using the heuristic cluster finding method.

In the default reconstruction, the dE/dx of the minimum ionizing region is tuned with additional calibrations to be at $dE/dx \approx 50$ (arbitrary units). The separation power is found to be 5.999, a typical value confirmed by reconstruction on multiple other datasets, validating the reconstruction method and software used to produce the data. In contrast, figure (4.2.5) shows the separation between the clean selections with the same reconstruction settings for the neural network cluster finder algorithm. Multiple classification threshold settings were investigated, while only the setting of 0.05 is shown.

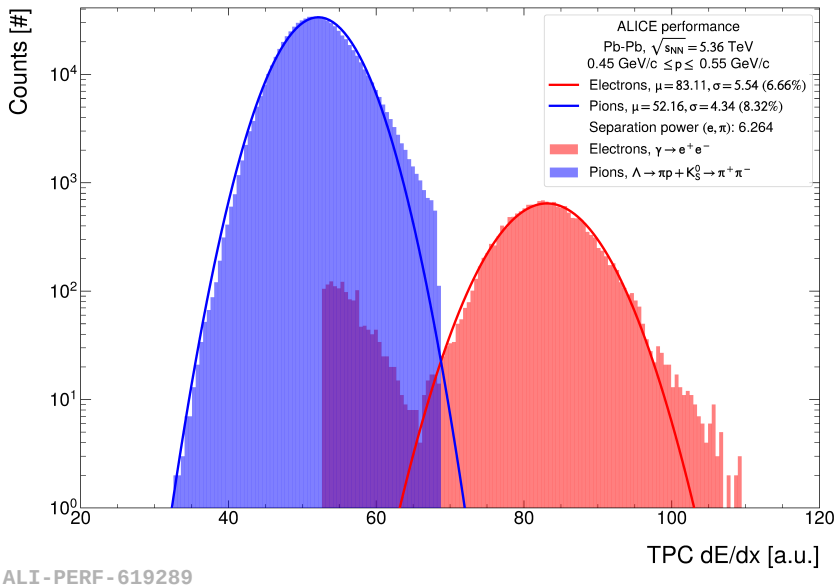


Fig. 4.2.5: Electron-pion band separation using the neural network cluster finding method. A threshold setting of 0.05 is chosen for the illustration.

Networks with an input charge array of (3,9,9) are used for both cluster classification and regression. The separation power increases to 6.264, which amounts to a 4.4% improvement over the heuristic cluster finding method. Furthermore, the higher charge estimate by the neural network leads to an overall increased $\mathrm{d}E/\mathrm{d}x$ response with the MIP value for the pion band found at 52.16 compared to 49.06 with the default cluster finding method. This is due to the larger charge accumulation window and the fact that the time gain calibration was not retuned on the new cluster finder method. Moreover, the neural network shows improvements in the relative pion and electron bandwidth of 5.0% and 5.8% respectively. Similar to the increased MIP position, this improvement is a direct consequence of the charge estimation and deconvolution capabilities of the regression network. Using the peak height and Gaussian width, the signal yield of the selected pions and electrons can be calculated. Using the normalized Gaussian

$$\mathcal{N}(\alpha;\mu,\sigma) = \frac{\alpha}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} := A \cdot e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \tag{4.2.3}$$

the integral in $[-\infty,\infty]$ is equal to $\alpha = A\sigma\sqrt{2\pi}/w$, where w is the bin-width of the histogram to which the Gaussian is fitted. The values of the integrals for multiple classification thresholds are stated in table (4.2.1).

Species	Setting	Gaussian integral (α)	Relative increase to CF [%]
e	Heuristic CF	1.75×10^4	-
	NN, thres. = 0.03, CF reg.	1.77×10^4	+0.90
	NN, thres. = 0.03	1.79×10^4	+1.93
	NN, thres. = 0.05	1.78×10^4	+1.79
	NN, thres. = 0.1	1.78×10^4	+1.61
π	Heuristic CF	72.3×10^4	-
	NN, thres. = 0.03, CF reg.	72.7×10^4	+0.50
	NN, thres. = 0.03	72.8×10^4	+0.69
	NN, thres. = 0.05	73.0×10^4	+0.90
	NN, thres. = 0.1	73.0×10^4	+0.78

Table 4.2.1: Integrals of Gaussian distributions of V0 daughter particles for various settings of the cluster finding algorithm.

While an increase is found when the heuristic regression algorithm is applied (case: NN, thres. = 0.03, CF reg.), additional increases are noticeable once the regression network is used. Particularly for the electron samples, an increase of the yield by 1.03% is found compared to the case using the classification with the heuristic regression algorithm and a 1.93% increase is found in comparison to the regular GPU cluster finder approach. This shows that tracks, previously skewed or not found due to charge overlaps are recovered using the neural network approach.

4.2.3 Bethe-Bloch parameter optimization

For the following calibration of corrections, additional particle samples are collected from regions of kinematic separation using the TPC and TOF detector. Particularly at low momentum, kaons ($p < 0.3$ GeV/c) and protons ($p < 0.6$ GeV/c) are well separated from other particle species and are included in the fitting procedure. The selection requires an approximation of the mean $\mathrm{d}E/\mathrm{d}x$, which is taken from the calibration of a previous dataset.

The functional shape of the Bethe-Bloch formula stated in equation (4.2.1) allows the tuning of seven parameters. Two parameters (a_1, f_{MIP}) are directly correlated, hence f_{MIP} is typically fixed to the mean position of the pions ($f_{\mathrm{MIP}} = 50$). Even with an increase of the MIP position by the neural network regression algorithm, the parameter f_{MIP} is not adjusted, as all scaling behaviors are absorbed in a_1. Since the fit is only performed for $|q| = 1$ particles, the scaling factor for the charge (f_z) is not adjusted. It is noteworthy that in the theoretical description of the Bethe-Bloch function shown in the introduction (1.1.1), the scaling law with charge is found to be exactly z^2. This parameter has not been retuned within the PID group since the start of Run 3, but is rather taken care of by dedicated analysis of light nuclei groups. The remaining set of adjustable fit parameters are $[a_1, ..., a_5]$.

For each particle species, the momentum is converted to $\beta\gamma = p/m$. Logarithmic bins in $\beta\gamma$ are set to cover the full kinematic range. In each bin, the $\mathrm{d}E/\mathrm{d}x$ values are histogram-binned and a Gaussian fit is performed. The mean values and their uncertainties result in a set of points, which is well described by the shape of the Bethe-Bloch parameterization. The parameters $[a_1, ..., a_5]$ are then fitted using a χ^2 minimization to the resulting set of values (μ_i, σ_i) for each $\beta\gamma$-bin. Figure (4.2.6) shows the resulting fits with their reduced χ^2/NDF values.

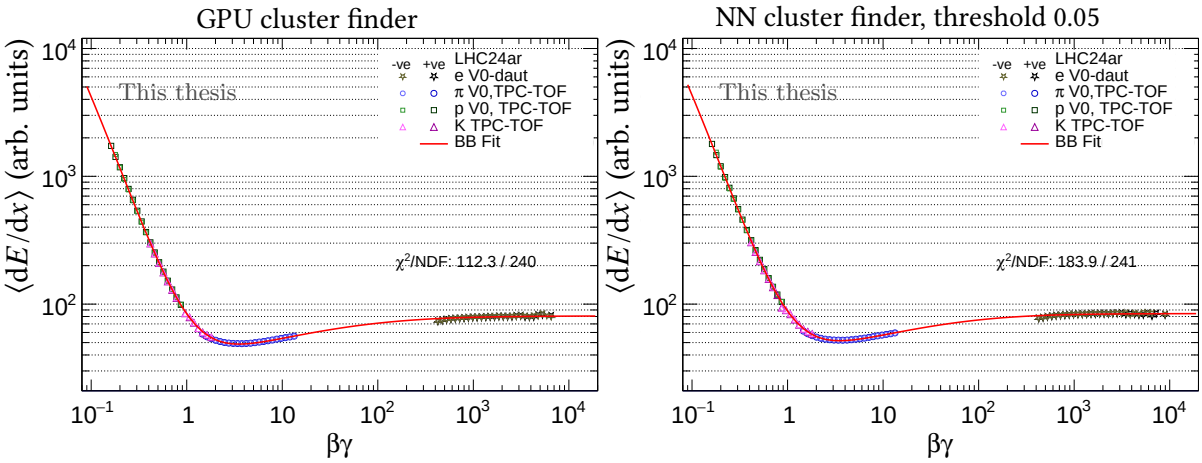


Fig. 4.2.6: Bethe-Bloch fit to resulting mean values of Gaussian fits for V0 daughter tracks, γ conversions and TPC-TOF selected particles for the heuristic cluster finding method (left) and the neural network cluster finding algorithm (right).

In comparison to the default cluster finding method, the fit performed on the dataset produced by the neural network cluster finder results in a higher χ^2/NDF. Major contributions arise at cross-over regions, mainly noticeable for selected kaons. This is a direct consequence of the initial selection of clean particle samples. Clean selections, particularly conducted with cuts on the TPC signal, use a default calibration object (parameterization) for the Bethe-Bloch curve, tuned to a MIP position of 50. As this is not accurate for the neural network cluster finder algorithm, it results in impurities and asymmetries in the selected samples. The resulting impurities however do not significantly deteriorate the fit and the result is considered sufficiently good for the initial calibration. Both methods result in a $\chi^2/\mathrm{NDF} < 1$, indicating that errors of the data points are overestimated. The resulting calibration, using only the Bethe-Bloch parameterization, describes the mean of each particle distribution to the $O(1\%)$ accuracy. With an intrinsic limit of the particle bandwidths to approximately 7% of the measured TPC $\mathrm{d}E/\mathrm{d}x$ signal, this calibration is only accurate to approximately 1σ of the signal. Particularly in cross-over regions and at low momentum, the calibration falls short of an accurate description. For the feasibility of a physics analysis, $O(‰)$ level accuracy ($\leq 0.2\sigma$) is required. The one-dimensional shape of the Bethe-Bloch parameterization cannot provide such accuracy. Secondary corrections on topological and kinematic track properties need to be applied where residual deviations are modeled by more complex functions.

4.3 Secondary corrections for particle identification

This section will concern the calibration of secondary corrections. At first, a motivation for the set of variables considered for such calibrations is given. Afterwards the procedure will be illustrated on datasets, reconstructed with different cluster finding methods and finally, analysis level quantities such as invariant mass resolutions are investigated.

For primary tracks, the ALICE TPC covers the range until $|\lambda| = \pi/4$ with a full detector readout. At higher inclinations a track will, at a certain radial distance from the beam-pipe, traverse outside the sensitive volume of the TPC with no further contributing clusters. High quality tracks can typically be reconstructed until $|\eta| < 1$ while even higher inclinations lead to reductions in quality and matching losses between reconstructed ITS and TPC tracks. Figure (4.3.7) illustrates the correlation between the inclination angle and the pseudo rapidity.

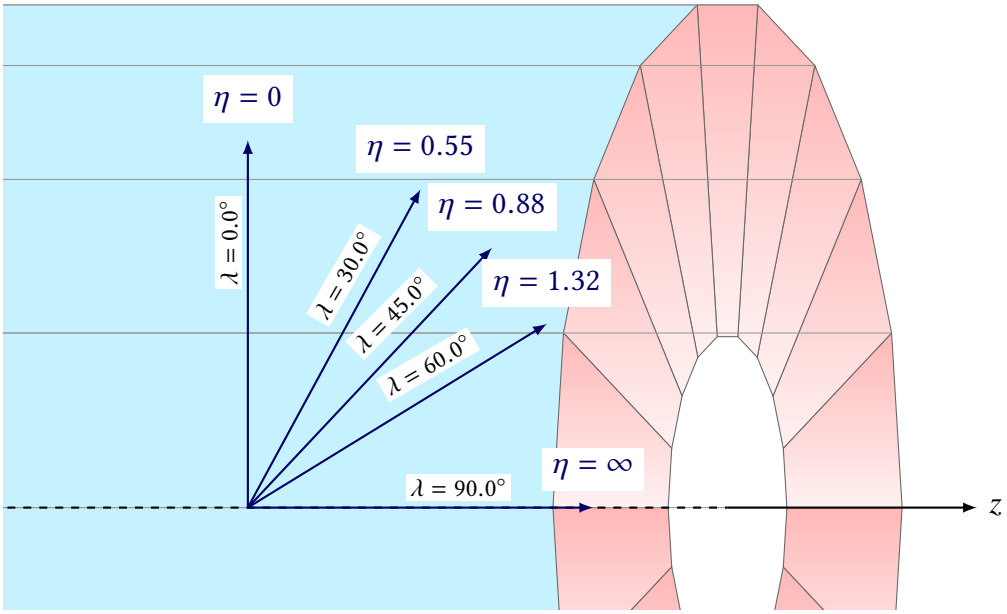


Fig. 4.3.7: 3D schematic of the ALICE TPC with 18 trapezoidal sectors per endcap, beam axis (z) and sections with λ and corresponding η values.

Aside the geometric acceptance of the detector, clusters of tracks at higher inclination angle lead to a larger charge deposition on the pad-plane, as the charge of a track seen per pad row is proportional to the length of the particle trajectory whose projection falls onto the pad row. The charge reaching one readout unit (pad) therefore increases with inclination, shown in figure (4.3.8).

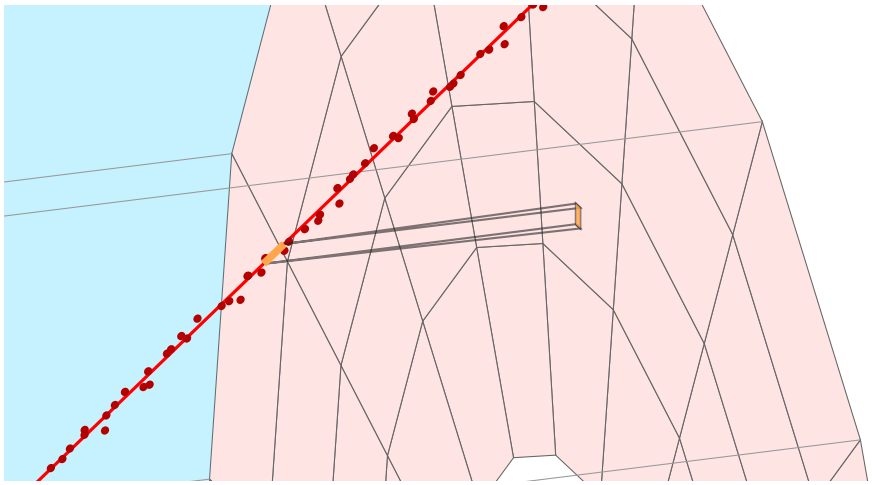


Fig. 4.3.8: 3D schematic of the ALICE TPC with one track at $\lambda = 50°$, clusters along the track, the IROC and OROC boundaries and a readout pad illustrated on the pad-plane (not to scale, for illustration purposes). The orange track-section shows the projected region of charge from a track on a readout pad.

This leads to a gradual change in charge measurements due to the track topology and impacts the reconstruction quality of inclined tracks. The track inclination, measured by η or $\tan(\lambda)$ ($\eta = -\ln(\tan(\frac{\pi}{4} - \frac{\lambda}{2}))$), is therefore one of the key observables to consider for secondary corrections.
Additionally, the number of clusters that are used in the tracking stage influences the track and reconstructed $\mathrm{d}E/\mathrm{d}x$ quality. After the application of the truncated mean (see introduction, (1.1.4)), the remaining shape of the $\mathrm{d}E/\mathrm{d}x$ measurements is approximately Gaussian. The bandwidth of the measured $\mathrm{d}E/\mathrm{d}x$ distributions per species therefore scales with $1/\sqrt{\mathrm{NCl}}$, where NCl is the number of clusters used for the $\mathrm{d}E/\mathrm{d}x$ computation. The number of clusters is therefore used as an input for the secondary corrections.
Kinematically, the reconstruction between particles and anti-particles (particles with the same mass and opposite sign of the electric charge) does not change. Topologically, these particles curve in the opposite direction in the applied magnetic field compared to their counterparts. Spallation processes, in which additional particles are released from interactions of particles from the collision with the detector material, can skew the measured particle to anti-particle ratios. However, this process mainly takes effect for particles heavier than the proton mass (light nuclei). Both effects are covered by $\mathrm{sign}(q)/p_{\mathrm{T}}$.

As a final piece of information concerning the track quality, detector occupancies play a major role for the reconstruction performance. Track densities decrease as $1/r$ (r is the radial distance from the interaction vertex) making the inner readout chambers of the TPC highly populated. Particularly in high-density environments such as Pb–Pb at a nominal interaction rate (50 kHz) with central collisions, clusters can suffer from major charge overlaps. Therefore, the TPC multiplicity (based on the number of TPC tracks) is added as an input to the secondary corrections. Since space-charge effects, increased detector noise and common mode effects can bias the computation of TPC occupancies, an additional occupancy estimator is introduced in Run 3, added from a high granularity timing detector, the FT0 occupancy. The FT0 detector [49], operating with quartz Cherenkov radiators optically coupled to photo-multipliers in the forward regime (high η, located at -19.5m and +17m from the primary vertex), is primarily used for initial vertex time determination and occupancy measurements. With a time granularity of $\sigma_t = 13$ ps, the detector can resolve individual particles stemming from multiple collisions and thus provides beneficial occupancy information for the particle identification response of the TPC [50].

The set of chosen variables for Run 3 is oriented by the choices made and experience gained throughout Run 2 and is summarized as

- p, the track momentum
- $\tan(\lambda)$, the track inclination angle

- sign(q)/p_{T}, the charge sign over the transverse momentum
- NCl, the TPC number of clusters used for the dE/dx computation
- TPC mult., the per-event multiplicity estimation in the TPC for a given track
- FT0 occ., the occupancy of the FT0 detector for a given primary vertex time associated to a track

A mass hypothesis is used as an additional input. At inference time, for each track, all mass hypotheses for which corrections should be provided, are evaluated (e.g. electrons, pions, kaons, protons, etc.). The output is thereby a table of size (num. tracks)×(num. mass hypotheses). The corrections are applied for every possible mass hypothesis. The selection is then performed at analysis time on the Nσ distribution of the particle of interest, closely resembling the selection process used throughout Run 2 data analysis. The full network input is therefore seven-dimensional with additional parameter sets being a constant topic of further development within the calibration group. Several considerations influence the choice of such variables, both from the physics point of view and the computational approach. The more variables are used, the more accurate the corrections will be across a wider range of the phase space. However, additional dimensions increase the sparsity of the dataset (curse of dimensionality), which can lead to a degradation of the fit accuracy or significant increases in training time. The variables mentioned above have shown great success in describing the correlations of the data and have proven their feasibility over time.

4.3.1 Neural networks for the TPC PID response correction

The tuning of secondary corrections to data taken during Run 3 of the LHC is fundamentally different in its approach compared to Run 2. A spline-based, per-dimension (inclination, momentum, number of clusters, etc.) calibration was used throughout Run 2 to correct the PID response of the detector. Each kinematic variable of choice, such as the track inclination angle, momentum or number of clusters was considered independent in this approach. Additionally, a functional parameterization of the dE/dx bandwidth was put in place to calibrate the signal width. While working at 1 kHz readout rate during Run 2 with comparably low track densities, Run 3 introduced a new challenge to the analyzers of the collaboration and the groups working on the detector calibration. Replacing the previous spline-based corrections, a neural network is trained to approximate the hypersurface of corrections which are applied to the initial Bethe-Bloch parametrization [45]. The approach is divided into three steps.
At first, a network (fully connected, 12 layers, 10 neurons per layer, tanh activation) is trained to approximate the mean of the dE/dx_{meas}/dE/dx_{exp} signal. The expected signal (dE/dx_{exp}) is the Bethe-Bloch parameterization tuned in the previous step. This network is applied to the initial training data set to correct it to a mean near 0 (within 1‰), from which a second network

(fully connected, 12 layers, 10 neurons per layer, tanh activation) is trained to approximate the bandwidth for each particle species. This corresponds to the σ used in the Nσ selection. At last, a third, smaller network (fully connected, 4 layers, 8 neurons per layer, tanh() activation) is trained on the output of the two previous, larger networks to be deployed in the O2Physics analysis framework. This network is used by the analyzers of the collaboration. The design is tuned to optimize physics performance and to allow for the large-scale inference application on the WLCG (Worldwide LHC Computing Grid) without exceeding memory and compute time bounds. The successful development, commissioning and deployment of this algorithm was part of the work conducted in long shutdown 2 of LHC, in preparation for Run 3. While data processing and the creation of training data for these networks remains a task performed by the PID calibration group, quality assurance of the results, continuous framework developments, computational optimizations and the complete production release of calibration objects for the data taken in Run 3 until the time of writing, were part of the work conducted throughout this thesis. The wide-scale use of this calibration over time marks a key achievement of this machine-learning-based approach. The continuation of this work after long shutdown 2 and successful calibration of all Run 3 dataset until the time of writing this thesis made the method state of the art in the field with several international conference contributions [46], [51], [52].

To validate the neural network cluster finding algorithm until analysis level, the calibration of the PID response marks one of the final goals of this work. The training data sample, described in the previous sections, is updated to include the values of the fitted Bethe-Bloch parameterization for each track and can afterwards be used to train the PID neural networks. Due to the increased total charge per cluster and the resulting increase in the MIP position, contamination enters the cleaned data when selecting particles using the TPC $\mathrm{d}E/\mathrm{d}x$. Particularly for electrons, the dominant neighboring pion band leaks significant contamination into the electron samples. This leads to initial failure of the tuning of secondary corrections. A tighter, symmetric cut of 3σ on the TPC signal around the electron mass hypothesis removes contamination and significantly improves the fit behavior. Figure (4.3.9) illustrates the electron band as a ratio to the tuned Bethe-Bloch parameterization with the applied selections (dashed lines) and without secondary post-corrections.

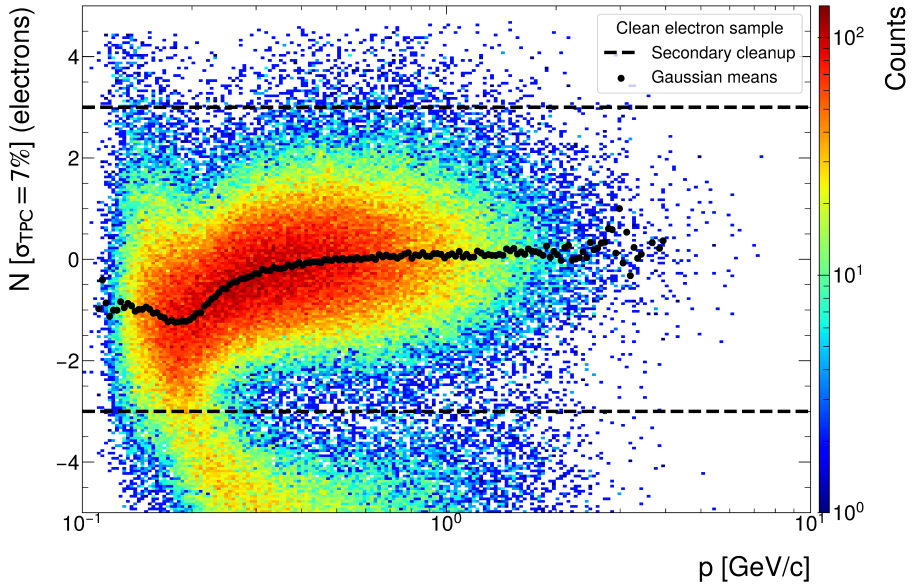


Fig. 4.3.9: Selected electron sample from γ conversions and additional post-selection at 3σ of the TPC signal (LHC24ar, apass2).

The pion contamination to the electron sample is clearly visible as a secondary band, merging into the electrons between $0.1 < p < 0.2$ GeV/c. To avoid this contamination in the subsequent fitting procedure, the cut at 3σ of the TPC signal is applied and results in sufficient noise suppression ($\sigma := 7\%$ of the $\mathrm{d}E/\mathrm{d}x$ signal is assumed for this first step). The cut does not remove significant contributions from the main density distribution of the electron signal. The chosen cuts are however necessary for the convergence of the fit procedure and do not alter the density distribution of the electron sample significantly.

The three-step procedure of training the neural networks is then carried out for all reconstructions performed with the various cluster finding algorithms. For the sake of brevity, only the results of default cluster finding method and the network setting of 0.05 are shown in figure (4.3.10).

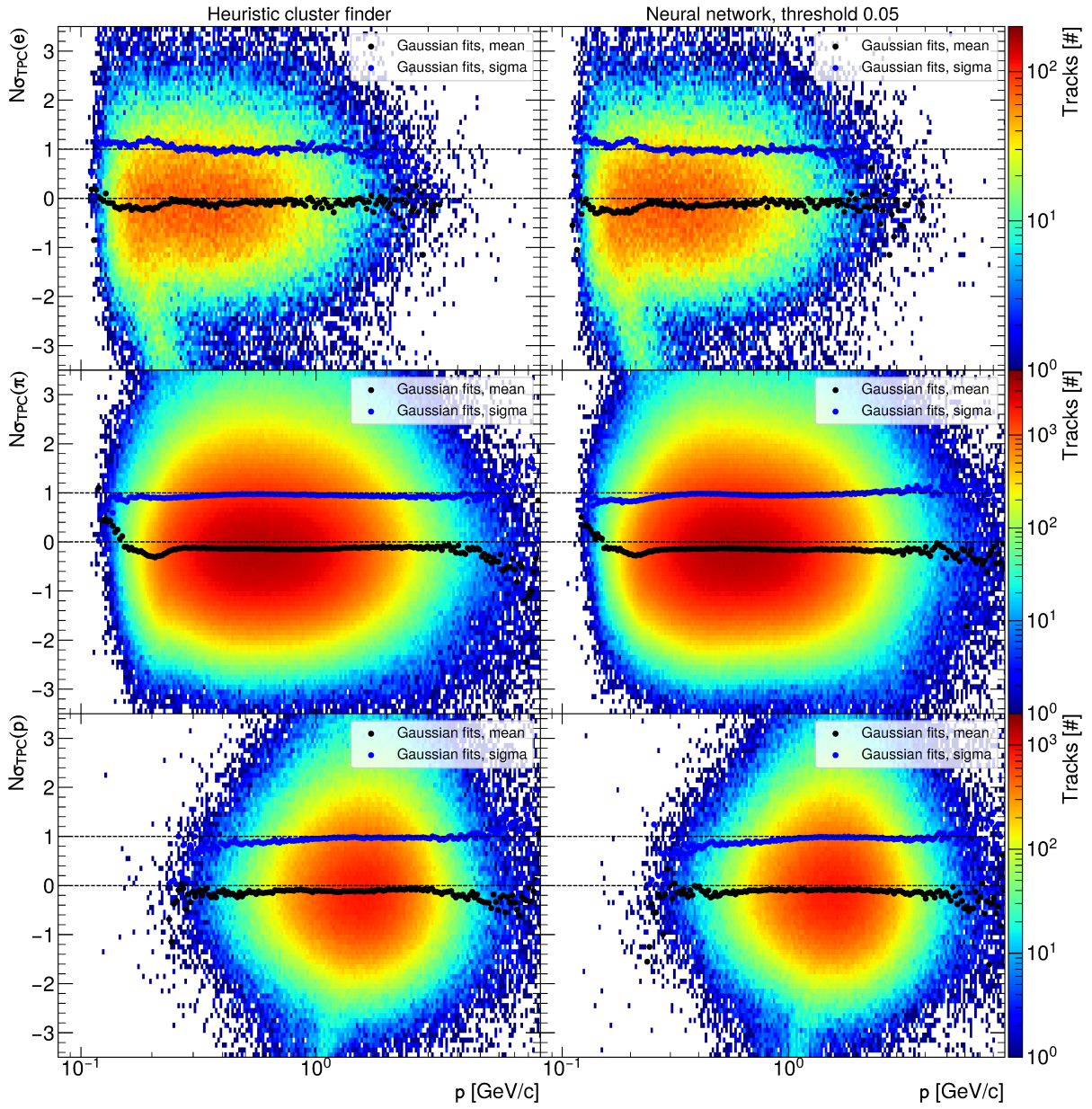


Fig. 4.3.10: Corrected Nσ distributions for the heuristic and neural network cluster finder algorithms on V0 samples for electrons (top), pions (middle) and protons (bottom). Dashed lines indicate the optimal value for the mean (black) and sigma (blue) per momentum bin.

No difference is found in the fit quality between the two cluster finding approaches, with satisfactory convergence of the learned secondary corrections. In both cases the mean correction is accurate to within $0.2\sigma \approx 0.2-0.3‰$ in regions of sufficient data density, which corresponds to the internal resolution of the utilized network size [45]. Similar accuracies are found from the fitted Gaussian widths. A general trend of the Gaussian mean positions is found towards negative values of Nσ, exemplary noticeable for the pion sample, with a shift by -0.1σ between $0.2 < p < 3$ GeV/c. This is a consequence of the residual non-Gaussian shape of the

distribution, stemming from the Landau distribution of charge per cluster in a single track. While the truncated mean corrects this behavior to a certain extent, the remaining $\mathrm{d}E/\mathrm{d}x$ distribution still shows a tail towards positive values in the Nσ distribution. The asymmetry is then reflected when training the first network (which learns the mean) using the MSE loss function (mean-squared error loss). It will not approximate the peak of a Gaussian, but rather the arithmetic mean

$$\mu = \frac{1}{N}\sum_{i=0}^{N-1}\frac{(\mathrm{d}E/\mathrm{d}x)_i^{\mathrm{TPC}}}{(\mathrm{d}E/\mathrm{d}x)_i^{\mathrm{exp.}}} \tag{4.3.1}$$

of the distribution in each local region of the input phase space. Therefore, the distribution is offset from 0 if the underlying dataset exhibits an asymmetric behavior. This will lead to an improved acceptance for a specified Nσ value, as more tracks towards the positive side of the Nσ distribution pass a given selection compared to a distribution centered on the Gaussian mean. A method for training a σ-value for the positive and negative half-side of the distribution was proposed and implemented in early stages of this work. However, this requires further investigation on the impact at analysis level and selection criteria and is therefore not used in production releases at the time of writing. Nevertheless, the applied corrections can correct additional effects and have shown improvements in many analyses [46].

4.4 Invariant mass distributions: K^0_S, Λ, D^0

Of primary interest from the physics point of view is the total yield, invariant mass mean position and width of reconstructed particle decay signals. The K^0_S, Λ and D^0 signals can be reconstructed even with the limited statistics at hand (8.8×10^5 analyzed events). The primary interest of this section will be effect of cluster rejection on the invariant mass spectra.

4.4.1 K^0_S and Λ mass resolution

The tuned corrections for particle identification allow a cleaner selection of secondary particles with the TPC. Both K^0_S and Λ particles benefit from the secondary corrections due to the improved selections of pions and protons of the resulting daughter tracks. The reconstructed invariant mass distributions are extracted using a five sigma selection for both pions and protons on the corrected TPC signal. Several fit functions are available to describe the shape of the peak. In order to evaluate the goodness of fit, the χ^2/NDF is extracted for each fit within a 3σ interval of the peak position. The results are shown in table (4.4.2).

K^0_S, χ^2/NDF	Single Gauß + pol3	Double Gauß + pol3	DSCB + pol3
heuristic CF	25.4	2.83	2.46
NN + CF reg., thres. = 0.03	26.0	2.80	2.45
NN, thres. = 0.03	24.5	2.97	2.51
NN, thres. = 0.05	24.1	2.88	2.42
NN, thres. = 0.1	24.9	3.08	2.56

Table 4.4.2: Goodness of fit for different fitting functions as measured by the χ^2/NDF for fits to the invariant mass distribution of the $K^0_S \rightarrow \pi^+\pi^-$ decay.

In all cases a third-order polynomial (pol3) was used to adjust for minor background contributions. The double-sided crystal ball (DSCB) fit performs the best overall. The χ^2/NDF calculation is performed for all datapoints (bins) within the 3σ range of the fit. Loosened cuts on the Armenteros-Podolanski variables were applied, compared to the previous step of clean sample selection in order to avoid artifacts in the resulting invariant mass distribution (see Appendix, table (C.1.3) for the detailed set of cuts). The resulting fit together with the mean and width of the DSCB + pol3 is shown in figure (4.4.11).

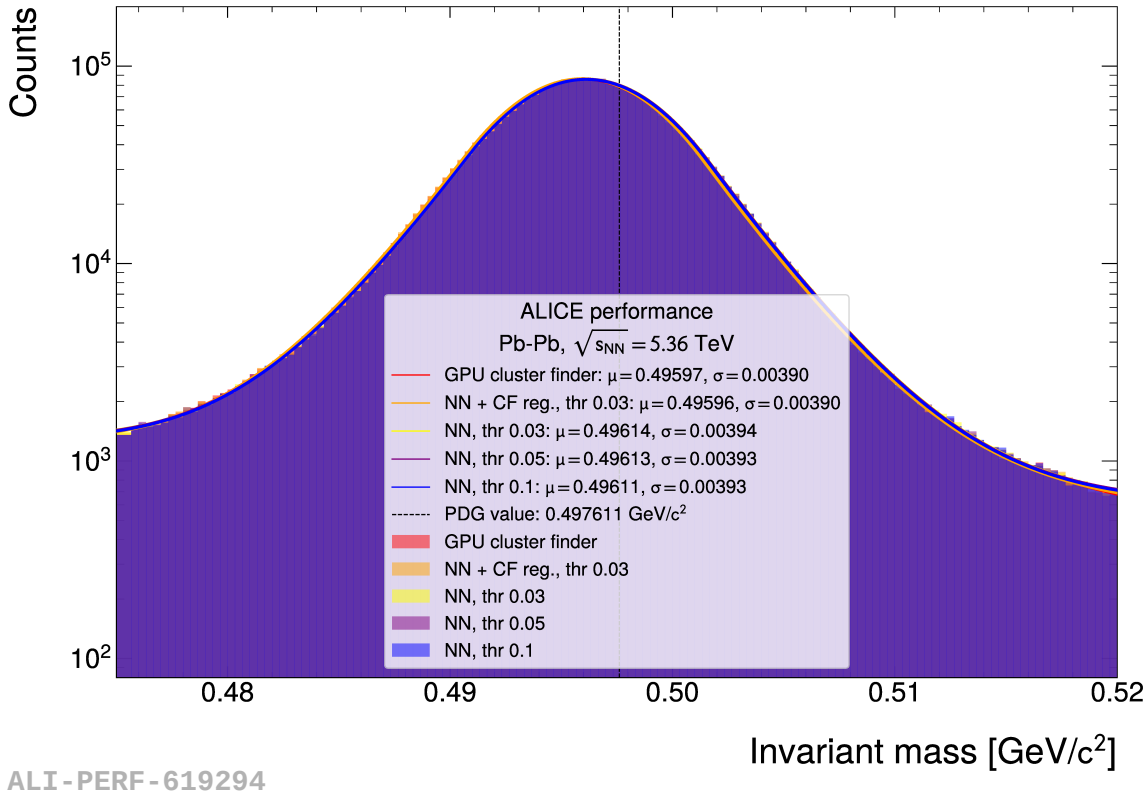


Fig. 4.4.11: Invariant mass peak of the reconstructed $K^0_S \rightarrow \pi^+\pi^-$ signal with double-sided crystal ball peak-fit.

It is a known behavior, under investigation at the time of writing, that both the K^0_S and Λ mass peaks show a mean value below the expected PDG value, indicated by the dashed line. The width for the K^0_S particle increases by approximately 1% while the mean value is improved by

up to 0.19 MeV/c^2 at the lowest rejection threshold (0.03), compared to the PDG value. The improvement in the mean position is also a direct result of the application of the regression neural network which is seen by comparing the cases "NN + CF reg., thr 0.03" (classification network with heuristic regression) and "NN, thr 0.03" (classification and regression neural network). A similar study is repeated for the Λ invariant mass peak. The results of using different fit-functions are shown in table (4.4.3).

Λ, χ^2/NDF	Single Gauß + pol3	DSCB + exp	DSCB + pol3
heuristic CF	55.3	19.4	1.12
NN + CF reg., thres. = 0.03	55.0	18.4	0.84
NN, thres. = 0.03	56.2	19.5	1.03
NN, thres. = 0.05	56.1	19.5	0.98
NN, thres. = 0.1	55.6	18.9	0.82

Table 4.4.3: Goodness of fit for different fitting functions as measured by the χ^2/NDF for fits to the invariant mass distribution of the $\Lambda \rightarrow \pi p$ decay.

Similarly to the K^0_S mass peak, the double-sided crystal ball fit performs best in describing the peak. The extraction of the width and mean invariant mass follow the same procedure as for the K^0_S decay and are shown in figure (4.4.12) for each investigated case.

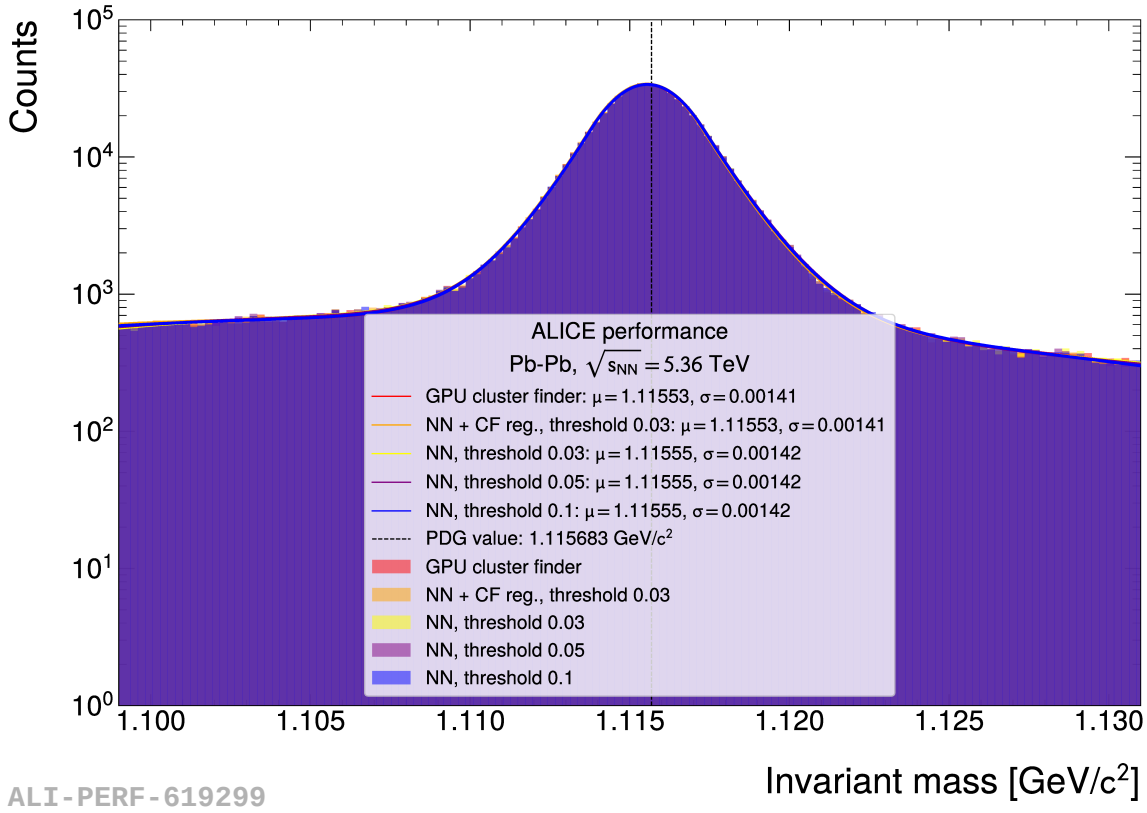


Fig. 4.4.12: Invariant mass peak of the reconstructed $\Lambda \rightarrow \pi p$ signal with double-sided crystal ball peak-fit and a third order polynomial for the background description.

In this case, no significant change is observable for the mean invariant mass or the peak width. The integral of the signal is extracted and compared between all algorithmic settings. The result is presented in table (4.4.4).

Signal integral, $(-\infty, \infty)$	K^0_S (DSCB + pol3)	Λ (DSCB + pol3)
heuristic CF	2.67×10^6	5.55×10^5
NN + CF reg., thres. = 0.03	+0.51%	+0.29%
NN, thres. = 0.03	+0.78%	+0.54%
NN, thres. = 0.05	+0.54%	+0.38%
NN, thres. = 0.1	+0.25%	+0.11%

Table 4.4.4: Relative difference of signal integrals of the invariant mass distributions for the reconstructed K^0_S and Λ mass peaks. Percentages indicate the difference to the heuristic cluster finding method.

Under all chosen threshold settings, the total yield of K^0_S and Λ particles increases as measured by the respective signal model. No difference is observed for the yields when applying the neural network or heuristic regression (compare: "NN + CF reg., thres. = 0.03" and "NN, thres. = 0.03"). All models are comparable and no reduction in the yield of either K^0_S or Λ particles is observed, even at the highest threshold setting.

4.4.2 Dependence of number of clusters and tracks on the neural network threshold

The above analysis has shown that the invariant mass yields and widths of the K^0_S or Λ peaks are maintained throughout the reconstruction. To put this into perspective, table (4.4.5) shows the reduction in number of clusters, evaluated on 200, randomly sampled timeframes for different threshold settings of the neural network.

	Total NCl red. (200 TFs) [%]	Total track red. (200 TFs) [%]
NN + CF reg., thres. = 0.03	7.083 ± 0.001	1.928 ± 0.016
NN, thres. = 0.03	6.992 ± 0.001	1.777 ± 0.016
NN, thres. = 0.05	11.213 ± 0.001	3.056 ± 0.016
NN, thres. = 0.1	18.115 ± 0.001	5.741 ± 0.016

Table 4.4.5: Number of reconstructed TPC clusters and tracks for different threshold settings. Percentages indicate the reduction compared to the heuristic cluster finding algorithm done on 200, 32-orbit timeframes of the Pb–Pb dataset (LHC24ar, apass2).

This shows a significant reduction in the number of clusters and tracks in the TPC. Both settings using the classification threshold of 0.03 return similar number of clusters and tracks. Regression therefore only has a minor influence on such parameters, validating the choice of noise threshold identified in the previous study (section (3.6)). Clusters and tracks are reduced

with increasing threshold, as expected from the MC studies. While the reduction in number of clusters is most relevant for final data size, the reduction in the number of tracks is equally significant and was investigated to ensure that no physics performance is lost. For this purpose figure (4.4.13) shows the number of reconstructed tracks as a function of their number of clusters from a set of 10 time frames (TF).

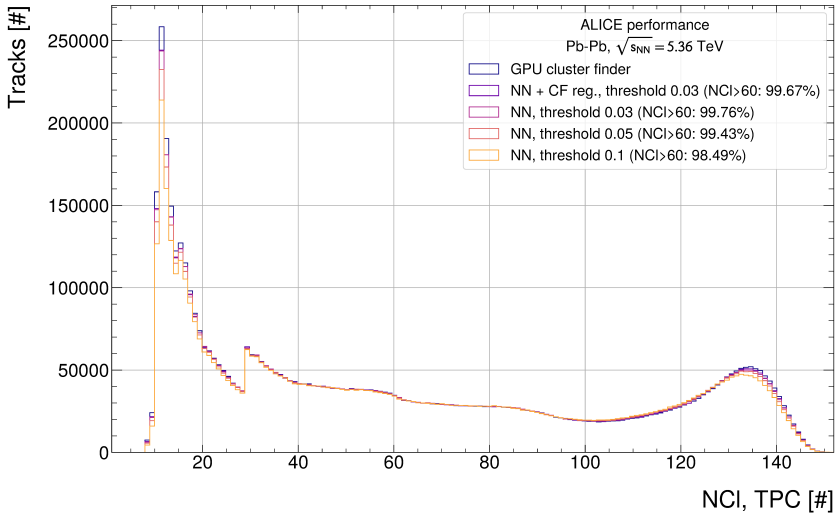


Fig. 4.4.13: Reduction in number of tracks as a function of number of clusters from 10 processed TFs.

Major track reductions occur at NCl < 20. At the highest rejection threshold, removing 18.1% of clusters in the TPC, the reduction in number of long tracks with NCl > 60 is found to be 1.5%. Such tracks are however not lost, but rather only have some clusters removed from their topology. The removal of such clusters from a track can hardly be avoided completely at any classification threshold setting. This is due to statistical variations in the cluster shape and the limited accuracy of training data (accuracy w.r.t. real data). Regardless, such clusters resemble the shape of bad clusters and are consequently not necessarily beneficial for the final physics performance. The improvements on the track fit (χ^2/NDF) shown on MC data are also verified on the real data sample. Instead of investigating the χ^2_{red} ($= \chi^2/\text{NDF}$ where $\text{NDF} = 2 \cdot \text{NCl} - 5$), a more typical analysis level variable is used, the χ^2/NCl, see figure (4.4.14).

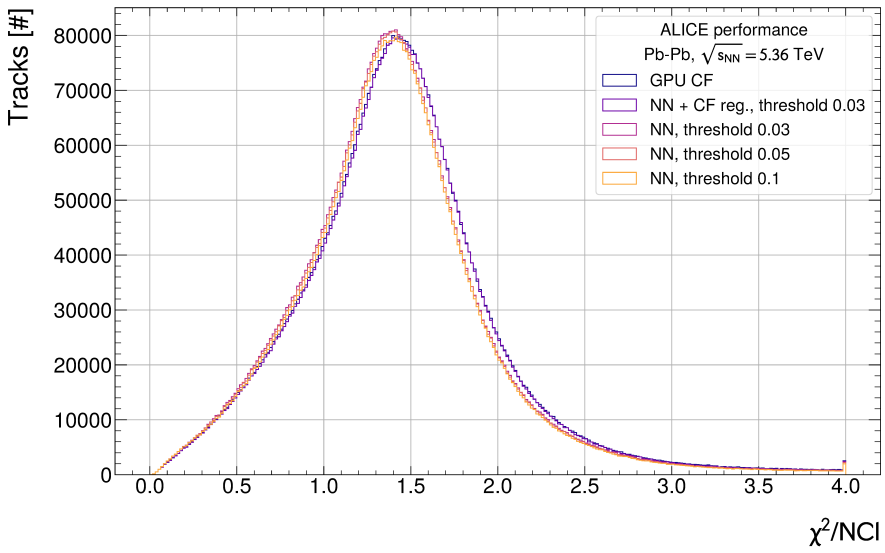
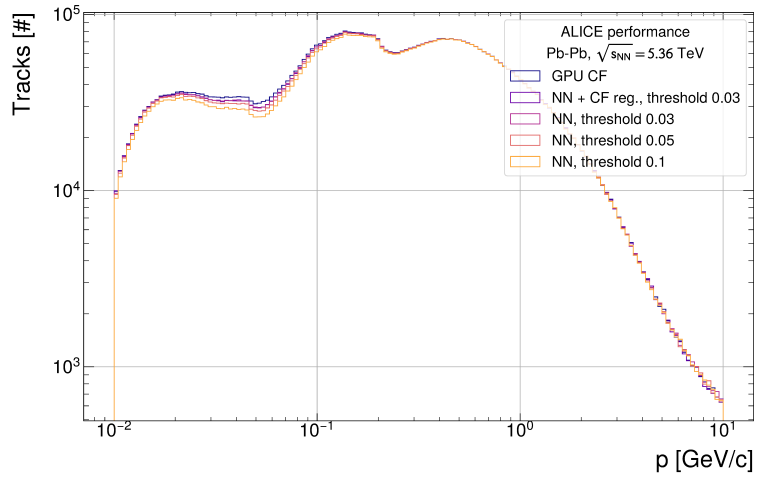
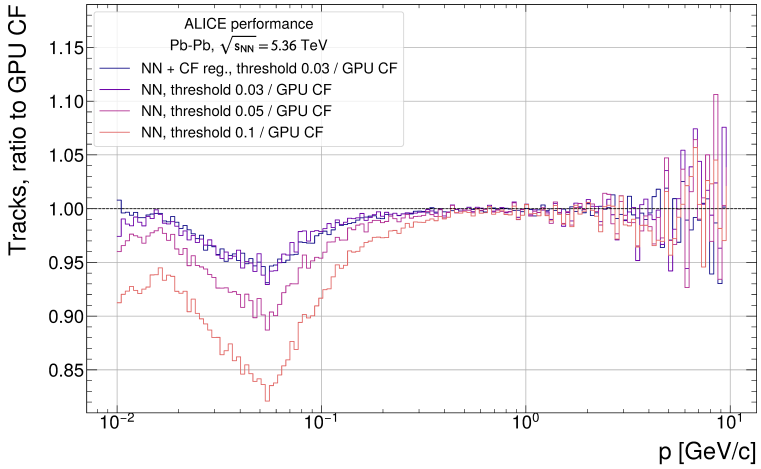


Fig. 4.4.14: χ^2/NCl distribution of tracks from 10 processed TFs.

In comparison to the previous study on MC data, cluster flags are used in the χ^2 computation to fully match analysis level performance. This reduces the track χ^2 as it increases the cluster error for all clusters marked by the split-flag during the reconstruction process. Improvements are seen in all cases where the neural network regression algorithm is applied. The improvement between the heuristic cluster finder and the neural network approaches is found to be $\Delta(\chi^2/\text{NCl}) \approx 0.05$. For the application using cluster rejection with the heuristic regression, a near identical distribution is found as with the default cluster finding method. This illustrates that improvements in the χ^2-distribution are produced by the neural network regression. Furthermore, at highest rejection threshold (0.1), the peak of the distribution is lowered by approximately 2.5% in total yield showing that also tracks at low values χ^2/NCl are removed. While this effect is undesirable, it is a direct consequence of cluster and track removal. It illustrates that such a rejection threshold is excessively strong and leads to losses of tracks with low χ^2 values. No increase in the peak position of the distribution is observed. Due to lower track losses at low χ^2/NCl, the settings of 0.03 and 0.05 are in this case preferable over the setting of 0.1, matching the analysis on MC data. The track reduction can additionally be investigated against momentum, shown in figure (4.4.15).



ALI-PERF-619314



ALI-PERF-619319

Fig. 4.4.15: Momentum distribution of tracks from 10 processed timeframes as raw yields (top) and as a ratio of the neural network algorithms with different threshold settings to the heuristic cluster finder approach (bottom).

Similar to the investigation on MC data, tracks are mainly removed at lower momenta up to approximately 0.1 GeV/c, mainly stemming from legs of short looping tracks. The reduction in tracks is found to be up to 17.9% between $0.05 < p < 0.06$ GeV/c compared to the heuristic cluster finding approach. In general a rejection of tracks can only be found until a momentum of $p < 0.4$ GeV/c, where high-p_{T} tracks are kept and are only subject to statistical variations due to the limited size of the available dataset. No measurable distinction can be made between the track-p_{T} distributions using the network regression and heuristic regression at same classification threshold (0.03). This illustrates that the track momentum is not significantly modified by the regression algorithm and reductions in tracks solely stem from the removal of clusters.

4.4.3 D^0 mass resolution

The previously analyzed Λ and K^0_S particles are identifiable by their decay topology and give a first indication of the reconstruction quality. The analysis of decays such as the D^0 meson requires a higher level of sophistication, as background suppressing models need to be employed in order to extract a measurable signal. The $D^0 \rightarrow K\pi$ is however still a common signal to be detectable with the very limited amount of data available for this analysis and one of the most common heavy-flavor decays analyzed in ALICE collaboration. It forms the bridge between pure reconstruction effects and analysis level investigations.

The collection of signal candidates requires detector calibration for both pions and kaons. Aside the test for reconstruction quality, the identification of hadrons containing heavy quarks is a key observable for the thermalization process of the quark-gluon plasma. Furthermore, it offers important constraints for perturbative QCD calculations at low transverse momenta due to the sensitivity of the gluon parton distribution function to the transferred momentum (Q^2) [53]. While the decay $D^0 \rightarrow K\pi$ accounts for $3.95 \pm 0.03\%$ of the decays (branching ratio[2]) the decay into $D^0 \rightarrow K\pi\pi^0$ adds noticeable contributions to the background distribution until 1.7 GeV/c^2. The spectrum of both D and $\overline{D}$ candidates are combined in the fit. Both exponential and polynomial functions were considered for modeling of the background. The polynomial function results in a lower χ^2 value of the background fit, hence the final shape of the invariant mass spectrum is approximated with a polynomial of order 3 and a Gaussian signal region. Signal yield (S) and background (B) are then extracted from the integration of the fitted functions, where signal is computed from the Gaussian integral as $A\sigma\sqrt{2\pi}$ (see eq. (4.2.3)) and background using simple integration rules for polynomial functions. The signal integral is computed over the $\pm 3\sigma$ range of the signal fit width.

A boosted decision tree model, applied in the default (official) reconstruction chain is used to select signal candidates based on track properties of the daughter tracks. This BDT model was not trained as a part of this work, but is taken from the condition and calibration database (CCDB). For this work, the same model is used to derive scores for the tracks reconstructed with the neural network cluster finding algorithm without retraining for each individual case. In contrast, the TPC particle identification response is tuned to each specific case to allow for near identical selection performance from the TPC calibration point of view. The selection criteria on the BDT scores are then optimized using a hyperparameter optimization framework called Optuna [54]. Additionally, the cos(PA) is added to the input parameters. The chosen

[2] https://pdglive.lbl.gov/BranchingRatio.action?desig=1&parCode=S032&home=sumtabM

score metric maximizes the signal yield (S) and significance ($S/\sqrt{S+B}$) using weighting factors w_1, w_2 and is written as

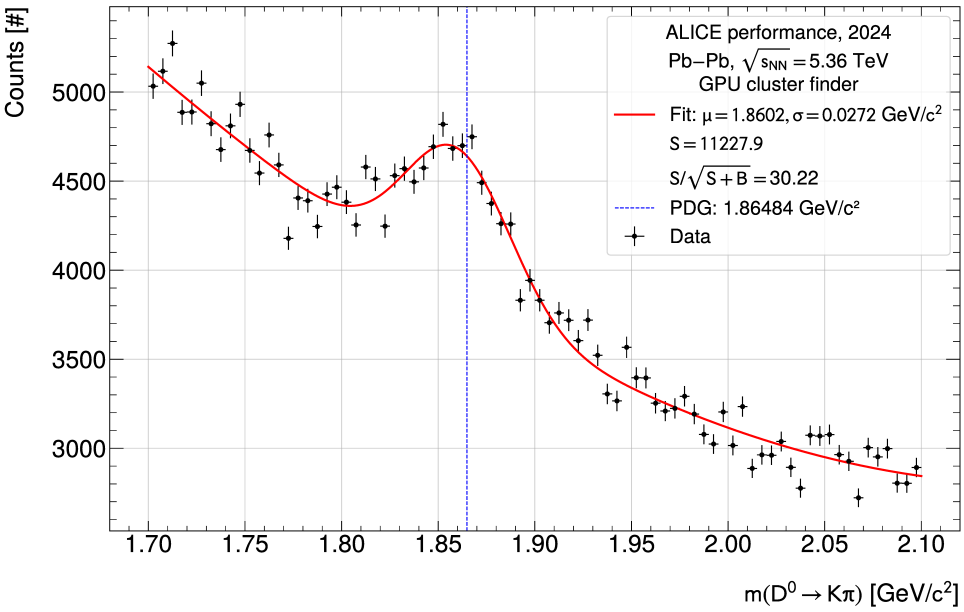
$$w_1 \cdot \frac{S}{\sqrt{S+B}} + w_2 \cdot \ln(1+S). \tag{4.4.1}$$

The Optuna framework then attempts to maximize this score under the Pareto-equal criterion specified by (w_1, w_2) using Bayesian optimization. This exploration of the phase space of possible cuts is substantially more robust than a manual tuning of parameters. Furthermore, the logarithmic function utilized in the second term of eq. (4.4.1) allows to keep optimization values in a similar range as the significance value while maintaining monotonicity with increasing signal yields. Weighting factors of $w_1 = 0.3$ and $w_2 = 0.7$ are chosen for the final optimization. 1000 trials are performed to optimize the signal yield and significance values. Tuning of the BDT scores and the cos(PA) is performed on the dataset with the neural network reconstruction algorithm and applied to both datasets identically. A cluster classification threshold of 0.1 was chosen for the dataset reconstructed with the neural network cluster finder to illustrate the signal extraction performance even in the case of the highest cluster rejection ($\approx$ 18% cluster reduction). The resulting distributions are shown in figure (4.4.16).

While the signal and significance of the current reconstruction algorithm are matched, the mass resolution improves by 1.5 MeV/c^2. The resulting differences are attributed to the large variety of detector calibrations which were not (and in many cases cannot be) recalibrate for the neural network cluster finder. The exemplary space-charge distortion calibration require $\approx$ 3s of continuous data for a reliable calibration, which cannot be provided with the recorded raw data. Additionally, the utilized boosted decision tree is trained on tracks of the standard reconstruction, which can have an impact on the selection process. Throughout the optimization many fluctuations of the result were noticed. To draw final conclusions needs dedicated BDT training and a larger dataset. Nevertheless, it illustrates that the novel reconstruction algorithm performs on par with the standard reconstruction and could have a beneficial effect on the reconstruction quality.

The explored phase space is visualized against the chosen Pareto-front using the signal and significance. It represents the path that optimization took over the iterations of the run. Figure (4.4.17) illustrates the trials taken by the Optuna algorithm.

The best value was achieved in trial 930 out of 1000. It is noticeable by the color gradient, that Optuna finds better solutions to the posed optimization problem in later iterations. The chosen trade-off between signal and significance is considered sufficient for the analysis conducted in this thesis. An optimization performed on the heuristic cluster finding method, where the



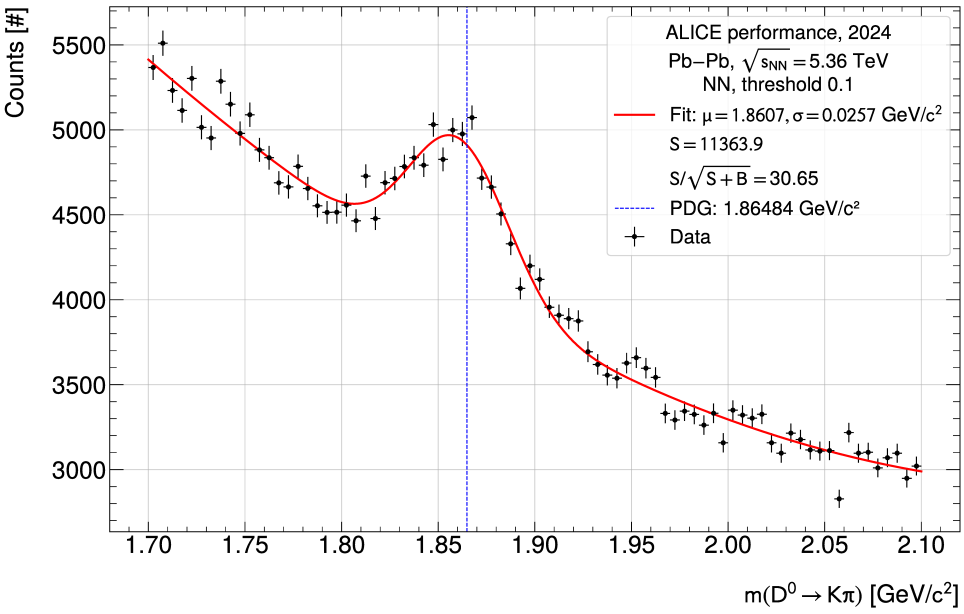


Fig. 4.4.16: Comparison of the extracted D^0 invariant mass peak with identical cuts used for the standard reconstruction (top) and the neural-network-based cluster finding method with threshold 0.1 (bottom).

resulting scores are applied identically to both datasets, shows similar comparability between the results and can be found in the appendix (C.2.19). Multiple other parameter tunings in reconstruction, training of the BDT model and selection of criteria can be conducted, which far exceed the scope of this project.

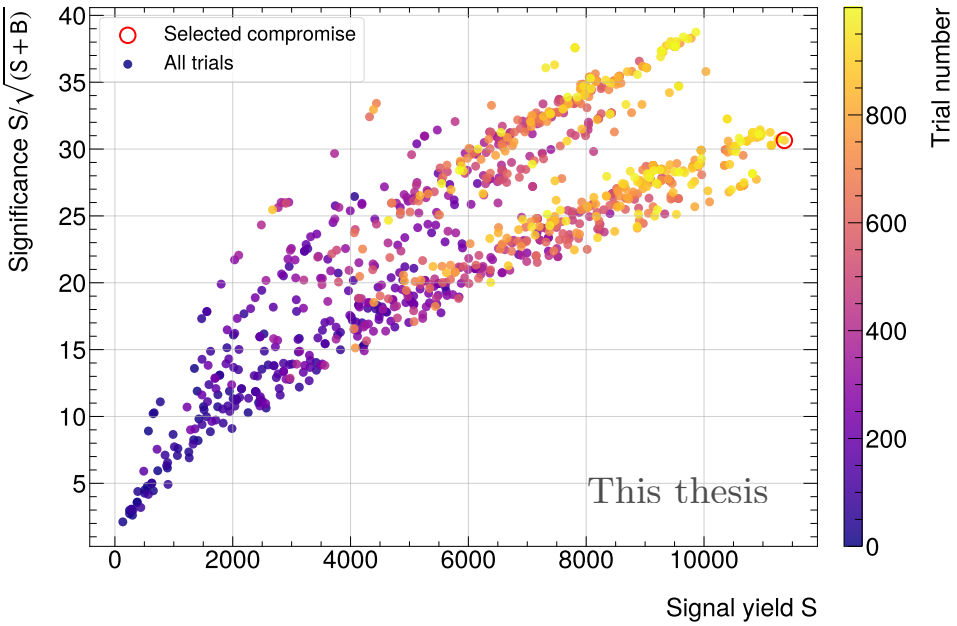


Fig. 4.4.17: Distribution of trials as a function of signal yield and significance for all trials in the Optuna optimization.

4.5 Commissioning runs and online reconstruction

The ultimate goal of this project is the application of the neural-network-based cluster finding algorithm during Run 4 of LHC and beyond. The successes of the algorithm were demonstrated in the previous sections and chapters and hold promising results for a feasible application, given sufficient computational resources. An application of this algorithm was not foreseen for Run 3 data taking due to computational limits. Additional testing and validation is to be conducted during long shutdown 3 of LHC (mid 2026 - mid 2030).

The algorithmic implementation succeeded by autumn of 2025, opening the possibility for commissioning runs in online data taking towards the end of Run 3. The first computational evaluation of the algorithm was tested using the full system test benchmark on one dedicated event processing node. It is a dedicated test that continuously processes a single timeframe of 50 kHz, Pb–Pb data N times (N is configurable) to uncover potential software issues. Timing tests and monitoring of compute resources, which will be outlined in the following chapter (V), resulted in a first estimate of the necessary computational resources required during online processing. The first estimate showed an increase of compute time by $\approx (1.3 - 1.6)\times$ (measured as throughput: number of processed timeframes per second). This allowed for a first set of synthetic (MC) and replay (real) data runs in the staging environment of the online farm, resulting in more precise estimates of the computational demands. While GPU and host memory were not a limiting factor, the increase in compute time was verified in this large scale test. However, for the low interaction rate runs occurring at this time of data taking, an

application of this algorithm was considered feasible due to otherwise unused resources in the pool of event processing nodes. A first set of online runs on synthetic and replay datasets was conducted on 3rd of September 2025 revealing software issues and the necessity for fixes to the data indexing within the framework. These first tests were conducted during a machine development (MD) period in which no beam is present in the LHC. This time is ideally suited for software tests, with the availability of the full computational hardware used during online runs. The promising results shown previously were presented in several meetings and finally brought forward to physics coordination of the ALICE collaboration, resulting in a permission for two runs of approximately 30 minutes each with the neural network cluster finder algorithm in online reconstruction. The first run is the most resource demanding as it uses the full neural network algorithm, with one network deployed for classification and one network deployed for cluster regression. The second run uses a classification network and the heuristic cluster finder regression and is therefore less demanding computationally. The environment delivered by LHC were 507 kHz (visible interaction rate), pp interactions at a center-of-mass energy of $\sqrt{s}$ = 13.6 TeV.

The two physics commissioning runs were taken on 3rd of October 2025 with a fully successful data taking throughout both of the 30-minute periods. No issues were encountered within the two runs. The configurations shown in table (4.5.6) were used.

Run number	#EPNs	Duration [mm:ss]	Int. luminosity [μb^{-1}]	q_{Tot} cutoff
566696	250	30:34	15888.085	8
566697	220	28:54	15094.528	8

Table 4.5.6: Metadata of physics commissioning runs with the full network configuration (566696) and the classification network + heuristic regression configuration (566697).

By accident, the total charge threshold of 8 was also utilized for the case when the heuristic regression (run 566697) was used. As shown in the previous chapter (figure (3.6.36)), this reduces the number of clusters by 2.8% and the number of tracks by 0.7% for a collision system of pp at 1 MHz interaction rate. The resulting physics performance is not significantly impacted by this choice. The total dataset sizes are 51.5 TB (run 566696) and 48.2 TB (run 566697) respectively. These sizes are compared to a run taken with the standard cluster finding method to compare the reduction in total data volume. A run with the same set of detectors is used for an apples-to-apples comparison. The sizes to be compared are the data volume (GB) per unit of integrated luminosity (μb^{-1}) as this already includes an appropriate scaling for run duration and number of events. Event pile-up is kept at a constant factor of 0.022 for all three runs, ensured by the ALICE-LHC interface. The comparison of the resulting data sizes are shown in table (4.5.7).

Configuration	Dataset size (GB)	Int. luminosity [μb^{-1}]	GB / μb^{-1}
default, full heuristic	395660.114	110754.709	3.572
NN class. + CF reg.	48187.350	15094.528	3.192 (-10.64%)
NN, class. + reg.	51450.699	15888.085	3.238 (-9.35%)

Table 4.5.7: Data sizes as measured by the total size of CTF files per unit of integrated luminosity. Percentages indicate the difference to the default reconstruction.

This shows the expected reduction in the number of TPC clusters, which is by far the dominant effect for the size of the dataset. The reduction in CTF size (dominated by TPC clusters) is found to be 9.35% for the full NN configuration and 10.64% for the configuration with the classification network and heuristic regression, compared to the standard reconstruction algorithm. The difference between both datasets of 1.29% is mainly a result of reduced number of clusters from the increased q_{Tot} threshold for the second configuration. It matches the expectation from offline reconstructions on 1 MHz pp data and allows for a physics comparison between all three settings. The data reduction also perfectly matches the expectation on the Monte Carlo-based evaluations (B.3.17).

Figure (4.5.18) shows a rendering of one of the first collisions taken with the neural network cluster finder (full configuration) in online processing (kindly provided by Felix Schlepper). Illustrated are tracks and associated clusters used in tracking for the ITS and TPC detector.

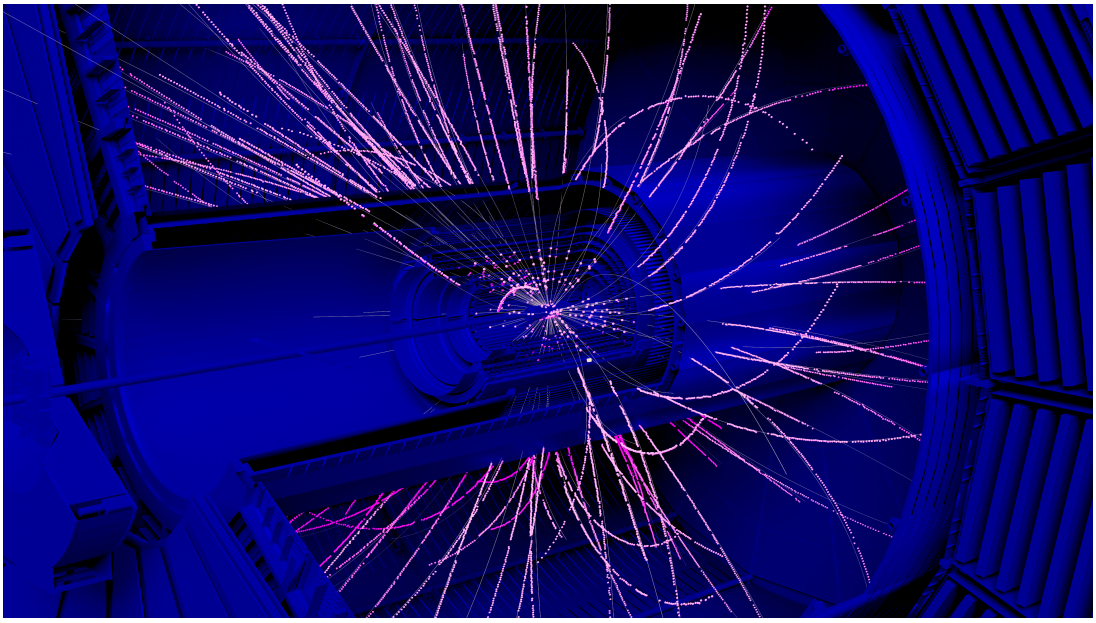


Fig. 4.5.18: Clusters and tracks from one collision collected with the neural network cluster finding algorithm in the online processing run (566696).

The validation of the physics performance using asynchronous quality control follows a concrete scheme and is centrally produced during the offline data reconstruction and calibration

using the available resources on the WLCG and the asynchronous part of the ALICE event processing nodes.

4.5.1 Analysis of commissioning runs and comparison with standard reconstruction

A first level analysis of the results is conducted directly after the reconstruction process. This mainly concerns the dE/dx signal produced by the neural network and the separation power between pions and electrons at the MIP region. Figure (4.5.19) shows the dE/dx distribution of the run which utilized the full neural network settings for both classification and regression.

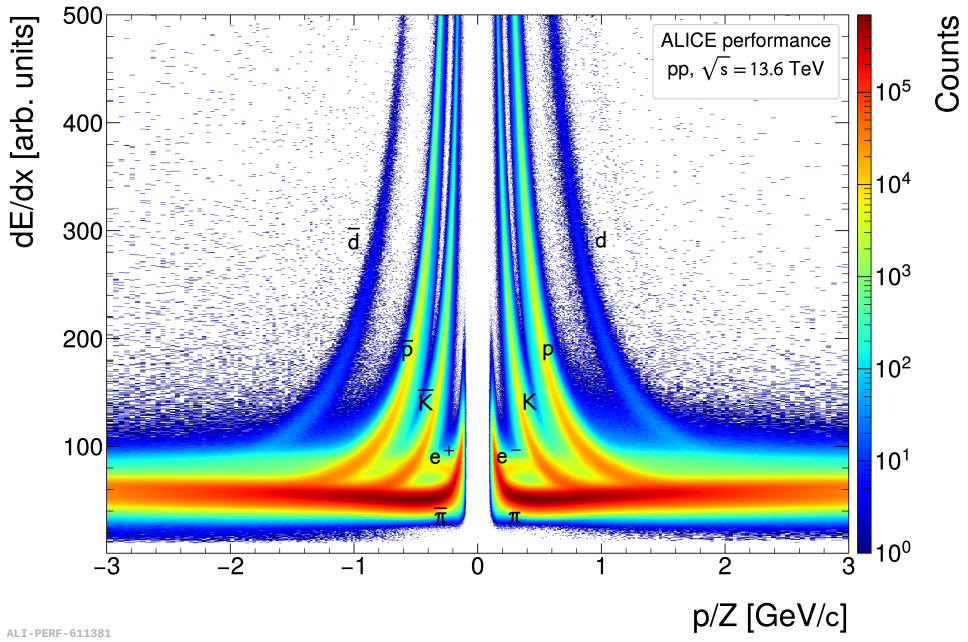


Fig. 4.5.19: dE/dx distribution as a function of p/Z after asynchronous reconstruction for Run 566696 using the full neural network settings (classification and regression).

The clear separation between the particle species is noticeable and allows for the calculation of separation power between pions and electrons at MIP. Unlike the previous analysis, the separation power in low interaction rate pp collisions is demonstrated directly on the dE/dx spectrum, as the pion band is far less dominant in comparison to Pb–Pb collisions at high interaction rate. As in the previous study, a range of $0.45 \leq p \leq 0.55$ GeV/c is investigated. As a reference, a run with the default reconstruction is used as a comparison. Run 566699 is chosen for this purpose, as it is taken in the subsequent LHC fill with the same machine settings, pile up factor, collision energy and interaction rate. Figure (4.5.20) shows the electron-pion separation in the MIP region for the full network configuration and the default, heuristic reconstruction.

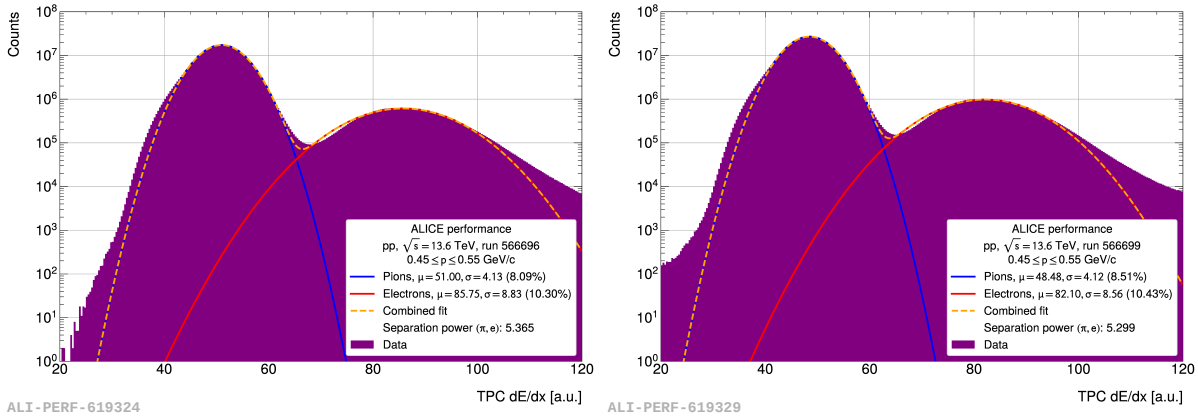
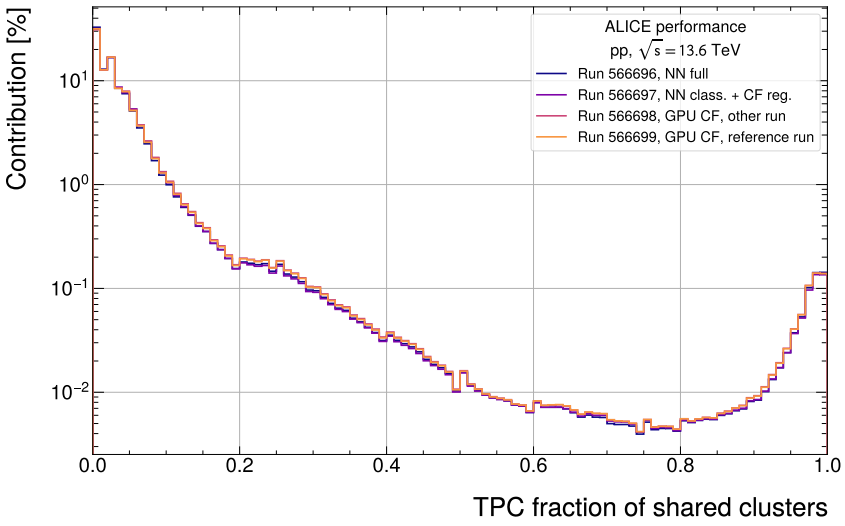


Fig. 4.5.20: Electron-pion separation in the MIP region for the full neural network configuration (left) and the default reconstruction algorithm (right).

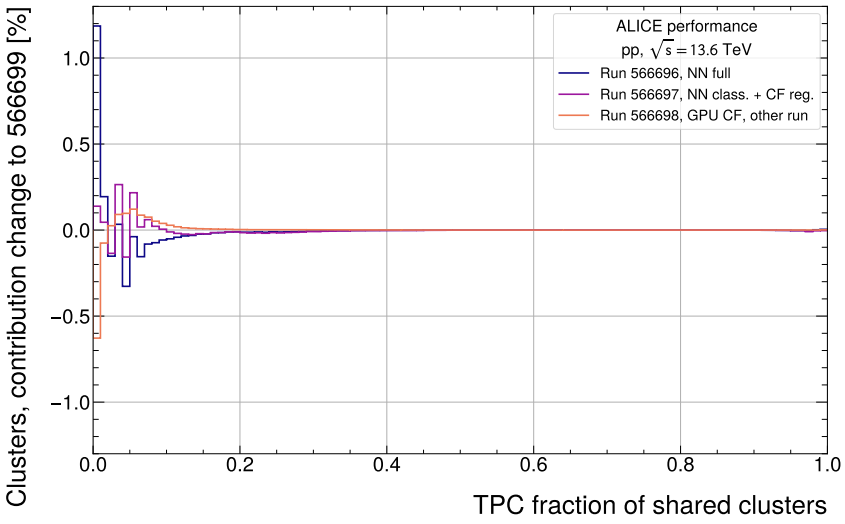
A double-Gaussian fit is used to approximate the observable pion and electron peaks. The cause of the observable non-Gaussian behaviors are the usage of the truncated mean for the $\mathrm{d}E/\mathrm{d}x$ computation and small contamination of the pion band from muons ($O(‰)$). Despite this contamination, the pion and electron peaks are well described with a clear improvement in the pion (4.9%) and electron (1.25%) bandwidth. This results in an improvement of the separation power of 1.25%. The observed improvements for the pion band are in agreement with the ones found in previous studies on Pb–Pb data, while the electron band shows a lower improvement compared to the Pb–Pb studies. This is most likely an effect of the significant reduction in track density from 38 kHz, Pb–Pb to 500 kHz, pp environments. Additionally, the improvement for electrons is only observable for the case where the neural network regression is used. In the case where the classification was used (and the incorrect noise threshold setting), the bandwidth of electrons increases (appendix, figure (C.3.22)), while the pion band still shows a small improvement (1.3%) and a decrease in separation power. Investigations of the shared cluster distribution further revealed that the regression network has beneficial effects. Such clusters can have significant charge overlap and contribute to noise in the $\mathrm{d}E/\mathrm{d}x$ distribution. Figure (4.5.21) shows the relative contribution of tracks in the dataset as a function of their fraction of shared clusters.



ALI-PERF-618816

Fig. 4.5.21: Shared cluster distribution for investigated runs.

In the recorded, low-rate pp environment, most tracks have low fractions of shared clusters with a peak of the distribution close to 0. This contribution amounts to more than 50% of all tracks having less than 5% shared clusters. The change in relative contribution to the total tracks and in relation to the reference run 566699 is shown in figure (4.5.22).



ALI-PERF-618821

Fig. 4.5.22: Change in relative contribution of tracks for a given fraction of cluster overlap as a ratio to run 566699.

This shows that changes made by the neural network cluster finding algorithm only have a significant influence to the total distribution of tracks until a cluster overlap fraction of 0.1. After this point, the contribution of tracks changes by less than 0.1% per bin. After a value of 0.03 in shared cluster fraction, the neural network lines stay consistently below the line of the reference run, which shows that the number of tracks with higher fractions of shared clusters

are systematically suppressed. These tracks are instead found close to bin 0 and account for approximately 1.4% of the integrated number of tracks. This means that shared clusters are systematically removed by the regression network, due to better cluster separation (by their center-of-gravity).

4.6 Summary

The validation of the neural network cluster finder is concluded from the physics point of view. Improvements on the D^0 mass with maintained invariant mass distributions of the K^0_S and Λ particle were illustrated on Pb–Pb data and the expected improvements on the $\mathrm{d}E/\mathrm{d}x$ bandwidth were found in both Pb–Pb and pp data. The physics performance was shown to be maintained even at highest cluster rejection thresholds of 0.1, corresponding to 18% reduction in number of TPC clusters. The commissioning of this algorithm marks a milestone for this project and explicitly demonstrates the feasibility for deployment in the online system. Additional improvements in the track fit (χ^2) and number of shared clusters were demonstrated even in the low track density environments of pp collisions at 500 kHz.

V Computing performance of the neural network cluster finding algorithm

The neural-network-based cluster finding algorithm has been developed for execution on parallel computing architectures, in particular GPUs. While its physics performance depends on tuning parameters of the simulation and reconstruction chain, the computational demands of the algorithm depend primarily on the size and structure of the neural network and on the available computing resources.
This chapter evaluates the computational performance of the full reconstruction chain and discusses the constraints relevant for both simulation and data analysis. The execution speed and memory usage of the newly developed algorithm are benchmarked on CPU- and GPU-based hardware. Particular emphasis is put on the comparison of different deployment strategies, including CPU versus GPU execution, fully connected versus convolutional network architectures, single precision (float32) versus half precision (float16) arithmetic and quantization, as well as performance differences across GPU hardware architectures using the ONNX runtime software for inferencing [55].

5.1 CPU-based computation

The online-offline software framework of the ALICE collaboration contains all necessary software components for the full reconstruction chain of ALICE data. Two separate applications need to be considered within this framework for the deployment of the neural network cluster finder: The application in offline processing and simulation (CPU-based) and the application in online processing (fully GPU-based). Both memory and compute speed requirements need to be considered for the application in either case.

For MC productions and local reconstructions it cannot be assumed that dedicated hardware acceleration (GPU) is present. The network is then only evaluated using CPU-based computations. For official productions, resources from the WLCG (Worldwide LHC Computing Grid) are used, which typically run on 8-core CPU machines. The task of cluster finding, is however not compute, but memory bound. Each network will therefore run single threaded. This avoids contention between resources, race conditions and thread stalling, but makes the evaluations potentially slower in terms of wall time. Therefore the measurements and application

will focus mainly on single threaded compute speeds and memory consumptions. A model size (number of layers and neurons per layer) as it is used for online deployment, is compared to the default reconstruction algorithm. To reduce memory usage, intermediate memory allocations and to keep a fixed size for the input vectors to the network, the input is loaded in a batched mode. Each batch contains a subset of cluster candidates found in the digits and is computed at once by the network with a single call to the ONNX runtime Run() function. All memory allocations for the NN input are specifically designed using memory-aligned pointers in order to keep resources free where possible and reduce memory loading operations. In the case of single-precision floating point layer operations and a batch-size of $2^{13} = 8192$ elements for a network with the (3,9,9) input ($= 3 \times 9 \times 9 + 3 = 246$ elements per input), this amounts to roughly 8 MB of memory usage for the input charge array allocation, excluding memory used for internal model and intermediate layer allocations. For larger batchsizes or input sizes, this number scales linearly with batchsize and input charge array size. Table (5.1.1) compares the peak memory usage for three different batchsizes chosen as powers of 2 over different orders of magnitude (32, 8192, 65536, 262144). Investigated is the private memory usage (dirty[1]), a measure for the shared memory that is utilized by the task and that has been altered during process execution. The same simulation of 25 centrality enforced Pb–Pb events (0-5%, 3.5×10^7 clusters with the default reconstruction) was evaluated 10 times with each algorithmic setting, providing a statistical error estimation of the total runtime.

Algorithmic setting, batchsize	32	8192	65536	262144
GPU CF			2.04 ± 0.01	
FC (class. + reg.), (3,9,9), 0.03	2.56 ± 0.08	2.43 ± 0.25	2.71 ± 0.17	2.96 ± 0.09
FC (class. + reg.), (3,9,9), 0.1	2.39 ± 0.05	2.34 ± 0.16	2.62 ± 0.09	2.82 ± 0.08
FC (class.) + CNN (reg.), (3,9,9), 0.03	2.08 ± 0.04	3.39 ± 0.64	2.75 ± 0.08	3.8 ± 0.17
FC (class. + reg.), (5,9,9), 0.03	2.56 ± 0.11	2.48 ± 0.09	2.91 ± 0.32	3.32 ± 0.06

Table 5.1.1: Peak memory usage (private, dirty) in GB for each investigated algorithmic setting using single-precision floating point networks (float, 32 bit).

For higher batchsizes, more memory needs to be allocated. The convolutional network outperform the fully connected networks only in the smallest batchsize setting. Residual fluctuations arise from the internal memory allocations for loading the model and storing intermediate layer computations. For the fully connected networks, the increase of memory usage of the cluster finding task is approximately 300-400 MB, and scales linearly with the batchsize (compared to the calculated size of 8 MB for the input charge array at a batchsize of 8192). This emphasizes the fact that the dominant contributions stem from allocations made by the ONNX runtime framework (like the model arena and buffer allocations) and intermediate layer com-

[1]Private refers to the set of memory pages that are owned by a process. Dirty refers to all memory addresses that have been altered during the execution of a process

putations. Figure (5.1.1) shows this memory consumption as a function of time from the start of computation, exemplary for the batchsize of 8192.

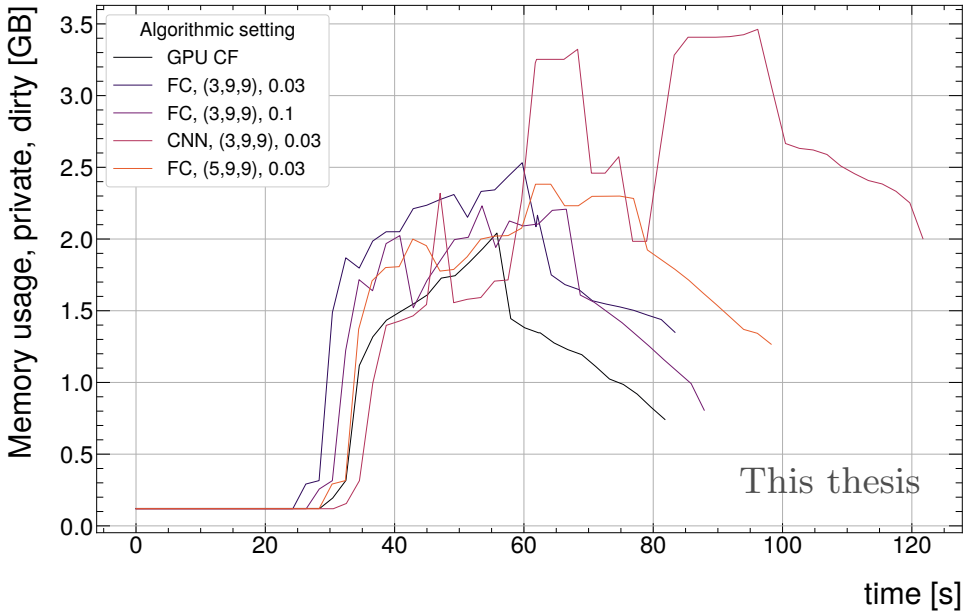


Fig. 5.1.1: Comparison of private memory (dirty) on one, centrality enforced (0-5%) simulation of 25 Pb–Pb events.

A major factor contributing to the computing demands beside the network is the filling of input data. Due to the increased input size of the network compared to the default reconstruction method, many memory (read/write) operations have to be conducted. Such memory operations are not a bottleneck on pure CPU-based systems, but will become more important in the following section on GPU computing. The total wall times of the reconstruction process for each setting and different batchsizes are reported in table (5.1.2).

Algorithmic setting	32 [s]	8192 [s]	65536 [s]	262144 [s]
GPU CF		26.1 ± 1.1		
FC (class. + reg.), (3,9,9), 0.03	53.5 ± 4.3	40.4 ± 2.6	41.1 ± 2.4	40.9 ± 2.0
FC (class. + reg.), (3,9,9), 0.1	54.1 ± 3.2	40.9 ± 2.1	41.3 ± 1.7	41.8 ± 2.1
FC (class.) + CNN (reg.), (3,9,9), 0.03	77.5 ± 1.8	65.4 ± 1.6	70.3 ± 2.2	71.2 ± 1.3
FC (class. + reg.), (5,9,9), 0.03	54.0 ± 3.9	49.7 ± 2.1	51.4 ± 2.0	53.4 ± 2.7

Table 5.1.2: Wall time of the reconstruction process for each investigated algorithmic setting using single-precision floating point networks (float, 32 bit) on 25 centrality enforced Pb–Pb events (0-5%).

The fully connected networks increase compute times by a factor of 2.1, while the convolutional network (applied only for cluster regression) increase it by a factor of up to 2.7. For the filling of the input data a dedicated kernel for CPU-based evaluations was designed, which

improves the speed by an approximate factor of 2 compared to using the GPU filling kernel. In comparison to the GPU-optimized kernel, the CPU-optimized version reduces the number of kernel calls and the number of internal index computations due to an increased set of sequential operations (nested for-loops) per kernel call. Concerning the network architecture, the convolutional network is most demanding in both memory (for larger batchsizes) and CPU time. For the evaluation of the fully connected networks, a speed-up of approximately 28% is gained when using a batchsize of 8192 compared to 32. This is achieved through reduced kernel calls and a resulting lowered instruction set, leading to more efficient CPU loading. In comparison to the case of 262144, no significant speed-up is found for the batchsizes of 65536 or 8192, showing that at this point, the floating point operations of each kernel dominate the kernel call overhead. Investigations on the individual evaluation of one convolutional network shows that it is approximately 7.4× slower than that of a fully connected network of comparable size and regression quality. For grid computations the convolutional network is therefore prohibitively expensive. Concluding this study, only the fully connected network is considered a feasible approach in terms of computational time with a near constant overhead in terms of memory usage.

To compare individual processing steps, the individual kernel times are summed up for the same dataset. The results for the components of the cluster finding task are shown in table (5.1.2)

Algorithmic setting	NN evaluation [s]	Input filling [s]	Cluster finding [s]
GPU CF	——— 6.2 ± 0.42 ———		18.2 ± 1.4
FC, (3,9,9), 0.03	10.29 ± 0.85	3.56 ± 0.43	31.7 ± 2.2
FC, (3,9,9), 0.1	10.9 ± 1.0	3.89 ± 0.75	33.5 ± 2.9
FC + CNN (reg.), (3,9,9), 0.03	37.67 ± 0.71	3.92 ± 0.57	61.0 ± 2.1
FC, (5,9,9), 0.03	17.75 ± 0.72	5.66 ± 0.28	41.9 ± 1.2

Table 5.1.3: Evaluation times of individual processing steps for CPU-based evaluations using different cluster finding settings for a batchsize of 262144.

The cluster finding time refers to the total time spent to produce the clusters from the digits. For the GPU cluster finder, the NN evaluation time and input filling time are replaced with the evaluation of the cluster finding (GPUCFClusterizer) kernel time. The NN cluster finder algorithm in the most feasible deployment configuration (FC, (3,9,9): table (5.1.3), rows 1 and 2) is approximately 1.8× slower. Besides the input filling and network evaluations, other kernels for publishing and flag setting make up the difference between the neural network and default cluster finding method.

5.2 Hardware accelerated computing

In contrast to fast, sequential executions, GPU-based systems thrive for problems that take advantage of parallelism. The inherent parallel design of a GPU allows for fast matrix-matrix and matrix-vector multiplications, ideally suited for neural network inference. Additionally, modern GPUs are equipped with dedicated hardware acceleration for lower precision or tensor core operations, such as half precision matrix-matrix multiplications. This saves compute time, increases efficiency and reduces energy consumption with a small to negligible reduction in precision.
The ALICE event processing nodes are equipped with 4 GPUs per NUMA domain, 2 NUMA domains per server and 350 servers in total. During Run 3 of the LHC, they operate heterogeneously on AMD INSTINCT MI50 and MI100 devices with 32 GB of device memory each. It will be the main focus of this chapter to investigate the feasibility of deployment, computational demands and optimizations put in place for the application of neural networks in the online processing of ALICE.

5.2.1 Computational limits for online processing

To investigate the feasibility for online computation, compute speed and memory consumption are the most crucial metrics to optimize. An initial estimate of the memory consumption was conducted using the rocm-smi toolkit[2] which measures the real-time memory occupation of the GPU. This revealed that for 16 neurons in each intermediate layer, the memory consumption increases by $\approx$ 2.5% for the reconstruction process on an MI50 card (= 0.8 GB) and scales linearly with number of neurons per layer. This poses no further hurdle for online deployment. The major optimization efforts will therefore focus on compute time and the choice of hyperparameters of the neural network and reconstruction framework. These include launch bounds for the number of threads per block and the number of blocks allocated for the computation of the network results, as well as the batchsize used for each individual evaluation of the network. Code-optimizations within the ONNX runtime framework are not considered.
Two optimization metrics are of primary interest. The first being the global wall time with parallel GPU compute streams for each evaluation of the networks, the second being the time measured per kernel for a serialized evaluation of the GPU kernels. The first metric is of global interest for the reconstruction time and hence deployment of the algorithm. In this case the results are convoluted with e.g. the tracking time, for which the choice of threshold of the classification network can influence the time used during the track finding and merging stage, due to the decreased number of clusters. The second metric is however of crucial interest for the optimization of the pure cluster finding task and evaluation of the individual kernel

[2] https://rocm.docs.amd.com/projects/amdsmi/en/latest/install/install.html

times, detaching the computational demands from other processing steps. Therefore, the optimization will first focus on the serialized kernel execution and show the effect on the global reconstruction time after that.

Before any optimization is attempted, it is already clear from the general structure and number of floating point operations of the heuristic GPU cluster finder, that the neural-network-based cluster finding algorithm will be computationally more demanding. While the default cluster finder uses a window of $1 \times 5 \times 5$, the filling of the input to the neural network already exceeds the number of memory accesses and floating point operations (e.g. charge normalization). This implies, that at every step computational demands must be kept as low as possible. As it was shown in the previous investigations, the input size of $3 \times 9 \times 9$ already results in significant physics performance increases compared to the two-dimensional, row-based cluster finding algorithm. Since no significant increase was observed for larger, three-dimensional input sizes (concerning mainly fake track reduction and efficiency gains), the input size for further investigations is chosen to be (3,9,9). The input is reused from the classification to the regression neural network to avoid further memory operations. The investigated networks are fully connected layers with the ReLU activation function, since all other non-linear activation functions come at computational overhead with multiple floating point computations per neuron (e.g. tanh(), GeLU(), sigmoid() all require the computation of exponential functions).

Using a fixed batch-size of 8192 elements (cluster candidates of size $3 \times 9 \times 9 + 3$ each), the network speed for different layer sizes is investigated in half precision. This provides a first estimate on the uncertainty of each measurement. The heatmap of compute times for different model sizes on one GPU kernel is shown in figure (5.2.2).

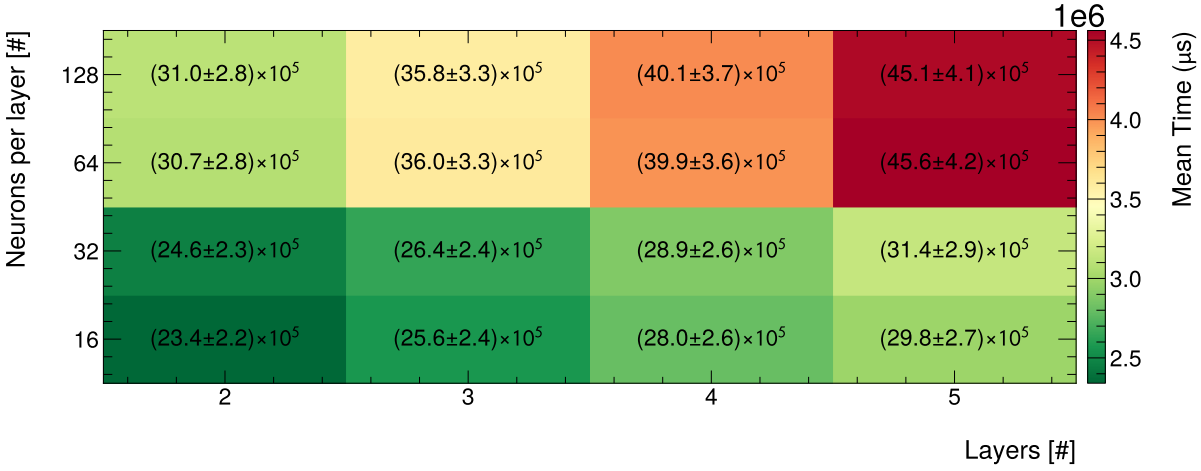


Fig. 5.2.2: Heatmap of the evaluation time for neural networks of different model sizes.

The batchsize was chosen as a power of 2 to maximally load all GPU threads (which are typically an integer multiple of 32). This illustrates that the timing precision of the kernel timers within the full system test and the chosen dataset size is given on the order of 1.5-3%. The test was conducted on 5 full system test evaluations of the same dataset (2 timeframes each, 32-orbits TF-length, $\approx 3 \times 10^9$ computed digit peaks). A different illustration shows the error-bands in relation to each configuration, see figure (5.2.3).

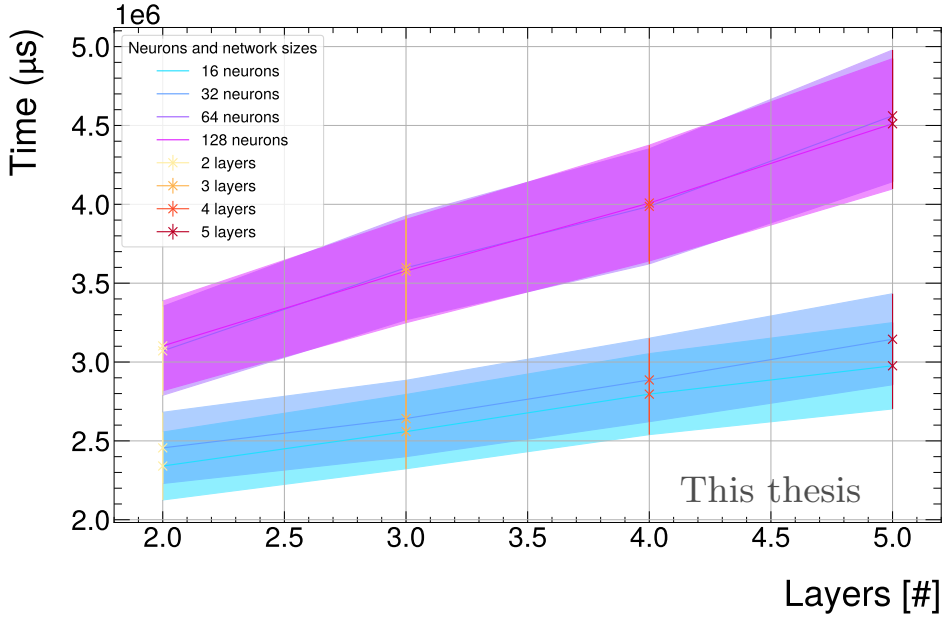


Fig. 5.2.3: Evaluation time and uncertainty for different model sizes as a function of number of hidden layers.

It is clearly visible that the hyperparameters of the network can have a significant influence on the compute time. Especially the neurons per layer have a noticeable effect for the given batchsize and launch bounds. For layers with n input and output neurons, the number of computations per layer scale with n^2. It is concluded that networks with at maximum 32 neurons are a feasible trade-off between computational resources and accuracy, while the wall time is dependent on the launch bounds of the kernel invocations. Likely, these numbers will also change between board specifications of the underlying hardware. This thesis will focus on the optimization on MI50 graphics cards. Increasing by one level of abstraction within the framework, figure (5.2.4) illustrates the computing time used in the cluster finding task in which the networks are evaluated in total numbers and as a factor to the default algorithm. To reduce the statistical error, 16 timeframes were evaluated for this investigation.

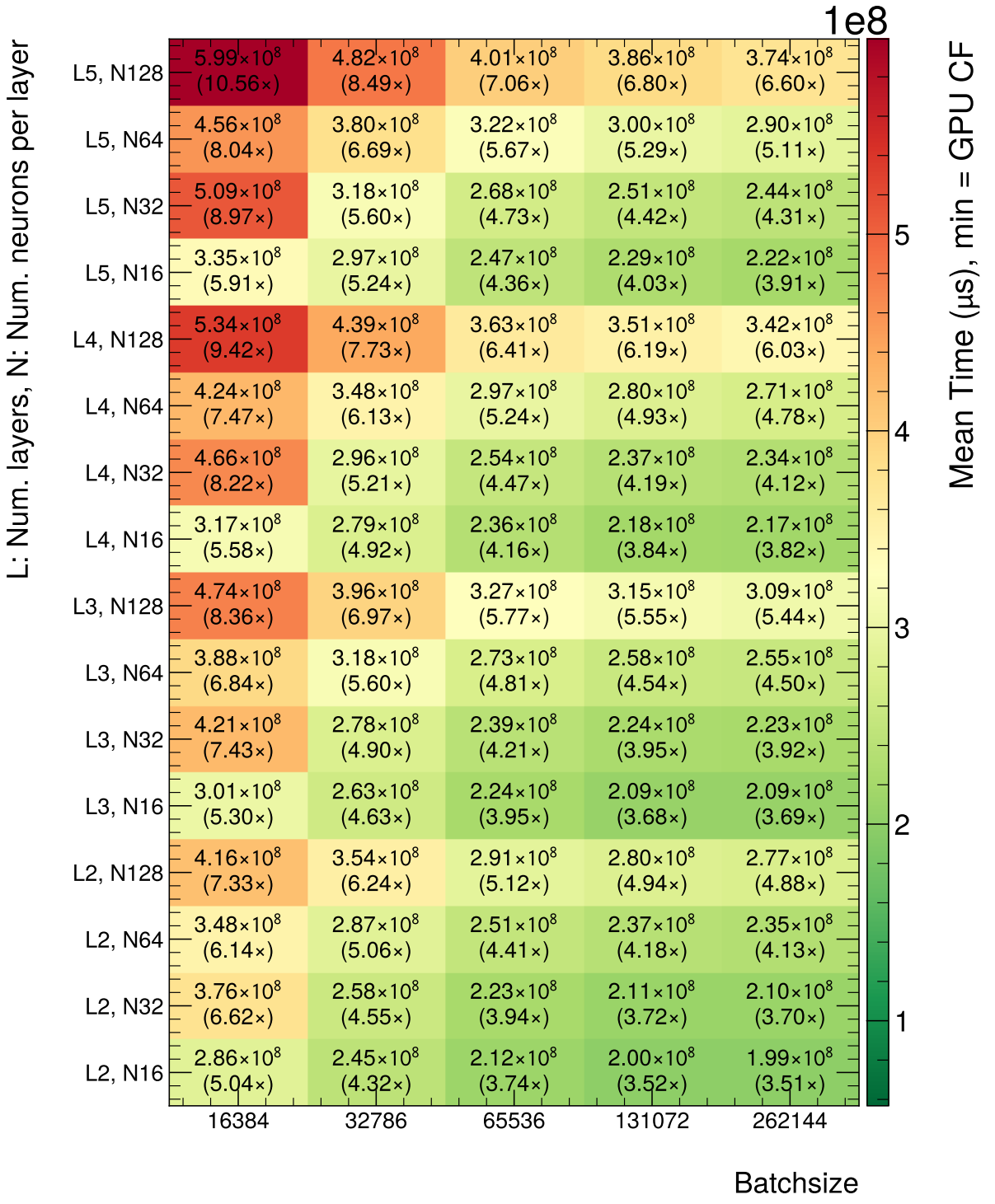


Fig. 5.2.4: Heatmap of the evaluation time for the cluster finding task with the neural networks in half precision for different batchsizes and different sized network sizes on one EPN node with 8 MI50 graphics cards, evaluated on 16 timeframes. Used abbreviations are L = number of layers and N = number of neurons per layer, for fully connected neural networks with input size (3,9,9) . The ratio to the default reconstruction algorithm (GPU CF) is stated as a factor in each cell.

It clearly demonstrates that large batchsizes are in general preferable over smaller batch-sizes for all network sizes. To fit within computational demands, the largest chosen batch-size for this investigation is the one used for deployment ($262144 = 2^{18}$). The total input vector for the neural network computation will therefore be of size $262144 \times (3 \times 9 \times 9 + 3) \times \text{sizeof(float16)} \approx 129$ MB. With 3 parallel compute streams (CUDA streams) this results in 387 MB of additional memory usage on each GPU. It is also noticeable that computation times increase noticeably when using more than 32 intermediate layer neurons. This is assumed to be a result of the hardware architecture of the MI50 GPUs on which these benchmarks were conducted: The hardware provides 64 threads per warp (or in AMD-specific terminology: 64 threads per wavefront) with 60 compute units ($= 60 \cdot 64$ individual threads). For two adjacent layers of the network, 64×64 parallelizable computations need to be done for a network with 64 intermediate neurons, exceeding the number of available hardware threads. Furthermore it is noticeable that the increase of computation time between (L=2, N=16) and (L=4, N=32) is only a factor $1.17\times$ which is already reached between (L=2, N=16) and (L=2, N=64). This suggests that no large compute overhead is used when using deeper networks with $\leq$ 32 neurons per layer compared to 16 neurons per layers. The step size for the number of neurons per layer was explicitly chosen as a power of 2 for this test to efficiently load all warps (wavefronts). Overall the cluster finding task is between $3.5\times$ and $4.5\times$ slower with the neural networks. Given the trade-off with performance on cluster regression (see figure 3.4.15), networks of L=4 and N=32 was chosen for deployment purposes. This amounts to a 40% increase in wall time of the full reconstruction process in the NN configuration (classification + regression network applied) for a batchsize of 262144 (see appendix, figure (D.0.23), configuration: (262144, (L4, N32))). Additionally, the times for all GPU kernel evaluations can be found in the appendix (table (D.0.4)).

5.2.2 GPU kernel optimizations

A major focus was put on the optimization of GPU kernels utilized for the filling of input data and publishing of final cluster data from the network output. While the publishing of cluster data only requires 10 read and write operations (5 cluster properties) and a minor set of floating point computations, the filling of the input data is significantly more demanding. In the recommended case, it requires $2 \times 3 \times 9 \times 9 + 3$ read/write operations of scaled uint16 or float precision (depending on the chosen setting). Additionally, floating point computations need to be conducted to find the correct indices from which data is read and to which data needs to be written, as well as one floating point operation for charge normalization per element in the input array. Coalesced memory accesses are crucial for this task and can significantly increase performance. The accesses of the memory are optimized for coalesced read operations from the charge array that exists at runtime in the GPU memory. Host-to-device copies (CPU-to-GPU or vice-versa) are strictly avoided in this phase as the memory bandwidth for such copy

operations is orders of magnitude slower than device-to-device copies in the internal shared or thread local memory. The ultimate benchmark for the filling kernel is the limit of memory bandwidth specified by the GPU model. In the investigated case an MI50 GPU is used to evaluate the kernel speed. With a maximum possible bandwidth of 1.02 TB/s the optimization goal are > 500 GB/s. Anything higher than this rate is in many cases not achievable unless data is specifically designed for the task of testing bandwidths. The kernel was optimized for coalesced memory accesses over the row and time coordinates, avoiding further overhead for memory loading. The layout of the kernel invocation requires a total of 43 bytes (read and write) in half-precision floating point operations and 61 bytes in single-precision floating point accuracy. The kernel is tested standalone with a batchsize of $2^{21} = 2.1 \times 10^6$ for each fill and 27 threads performing the dominant read-write operations, saturating the GPU thread capacities. A bandwidth of 400.6 GB/s (576.2 GB/s) was achieved in half (single) precision floating point accuracy. The desired goal of > 500 GB/s was only achieved in single-precision floating point accuracy, as the half precision read/writes require additional floating point operations for scaling and bit-shifts. In wall time of the kernel, both single and half precision are nearly equally fast, however a significant speedup is gained when using half-precision floating point operations for the neural network. An increase by factor of $(1.96 \pm 0.08)\times$ was found on the MI50 cards which matches the expected speed-up by a factor of 2 for half-precision floating point operations within the uncertainty. Due to the significant number of memory operations performed by the kernel, the time spent for filling the data is approximately $0.7\times$ the time as for a single network evaluation and thus not negligible.

5.2.3 Single vs. half precision: Accuracy and resource usage

The choice of using half-precision floating point (float16) precision for the weights and biases of a neural network is a standard method to reduce both compute time and memory usage. The dedicated hardware on MI100 GPUs can significantly speed up the computations performed in half precision. While older GPU models such as the MI50 do not contain dedicated hardware acceleration for float16, their threads still allow a speedup by a factor 2 for each individual operation. The conversion between float32 and float16 is performed using a method named *quantization* [56]. Weights of the neural network are scaled by passing a set of input data and scaling weights and biases of the network using the results of this computation, such that the computation in float16 best matches the computation in float32. Scaling the network's internal parameters does not change the model architecture. It is however required, as the resulting reduction in precision uses a reduced representation of the single-precision floating point numbers and should cover the range of values encountered in the network.

Single to lower precision floating point / integer conversion

Int8 quantization

A simple example is made using two single-precision floating point numbers: $[-2.6, 3.1]$.

Naively casting FP32 values to int8 (8 bit integer representation) by rounding the values within the range $[-128, ..., -3, -2, -1, 0, 1, 2, 3, ..., 127]$ would result in $[-2.6, 3.1] \rightarrow [-3, 3]$ and therefore a significant loss of precision. The precision loss can be mitigated as the full range of int8 is not utilized. Therefore, a scaling is applied which defines the "step width" between integers to convert the full range available in int8 to the required range for the representation. The scaling rule is written as

$$q = \text{clip}\left(\text{round}\left(\frac{x}{s}\right) + z, \text{q}_{\text{min}}, \text{q}_{\text{max}}\right) \tag{5.2.1}$$

where q is the stored integer representation within the new range of numbers (e.g. int8 within the range $[\text{q}_{\text{min}}, \text{q}_{\text{max}}]$), s the scaling factor to be determined to optimally cover the range of values (defines the "step width" in high precision range between two numbers), x is the number to be converted in high precision and z the zero-point (offset) which is also tuned for optimal range coverage. For uniform min–max quantization of a set of values S using b bits, the scale is commonly chosen as

$$s = (\max(S) - \min(S))/(2^b - 1). \tag{5.2.2}$$

Half-precision floating point

For half-precision floating point, no explicit quantization scheme is applied. Weights and biases of the neural network are simply rounded to the nearest representable FP16 number by truncating the mantissa and adjusting the exponent if needed. This comes with minimal losses in precision of the network prediction. The precision degradation (Δx) for a single precision value x scales as $\Delta x \sim x$.

Due to the normalization of input and output, the network weights are found to be within a total range of $[-3, 3]$, shown in figure (5.2.5).

The largest weights are found for the input layer, where scale adjustments of the input charge array are made. Otherwise, the network weights are mainly found within a range of $[-1, 1]$ and allow for efficient scaling. Using the quantization provided by the ONNX runtime framework, the final distribution of output values between the original, single precision and quantized, half precision network is compared. Figure (5.2.6) shows the relative difference between the model outputs as a function of the absolute difference of values.

Due to the reduced mantissa precision of half-precision floating point arithmetic, larger output values result in larger absolute rounding errors ($\Delta x \sim x$). Therefore, the illustrations show

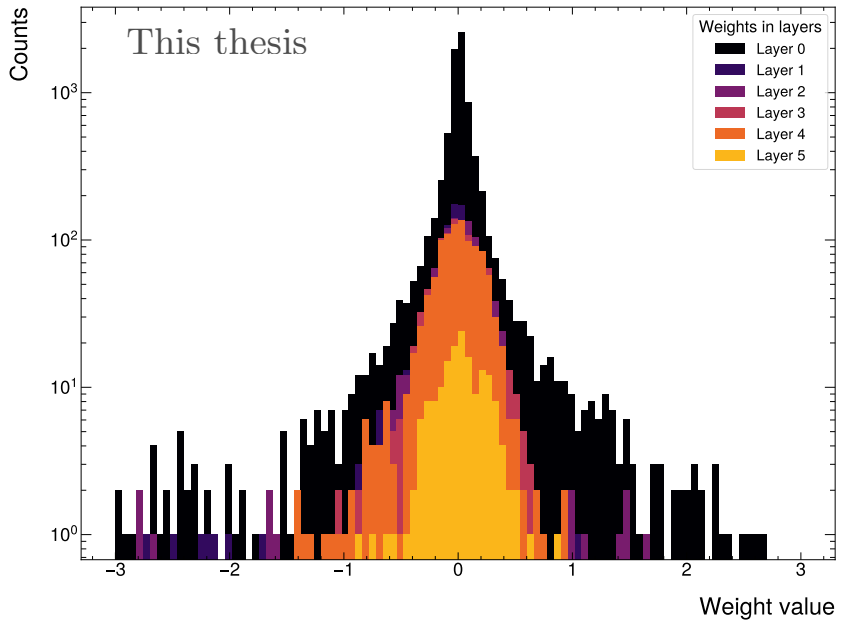


Fig. 5.2.5: Weight distribution per layer of the deployed neural network for online processing in float32 precision. Input and output layer are counted as layer 0 and N-1 here.

the residuals between the half and single-precision floating point values for each output neuron. The right-hand side of figure (5.2.6) clearly shows that highest floating point errors are made for the $q_{\text{tot}}/q_{\text{max}}$ and σ_{time} estimate, while the center of gravity and σ_{pad} estimations are well described with minor absolute deviations in comparison. This behavior is expected, as floating-point rounding errors scale with the magnitude of the represented value, and the $q_{\text{tot}}/q_{\text{max}}$ and σ_{time} outputs span the widest dynamic ranges due to the asymmetry of the cluster in the time direction and the overall cluster size.

The losses due to conversion in precision are well within the limits of accuracy of the neural network estimation of the quantity of interest. Differences of larger than 1% in relative deviation are only observable for values with an absolute difference of $\Delta_{\text{abs}} < 0.002$ for all output values of the network and will therefore only influence the innermost percentiles of the distributions shown in the MC studies. The effect is however entirely negligible compared to the extent of the residual distributions for each of the output values (see chapter III). Due to the available hardware acceleration for half-precision floating point operations on modern GPUs, half precision is chosen for the final deployment strategy. Using Int8 precision was not considered for the current investigations, as the prevalent hardware accelerators do not support additional computational speeds for this quantization compared to half precision. Therefore, Int8 quantization is considered an aspect for future investigations.

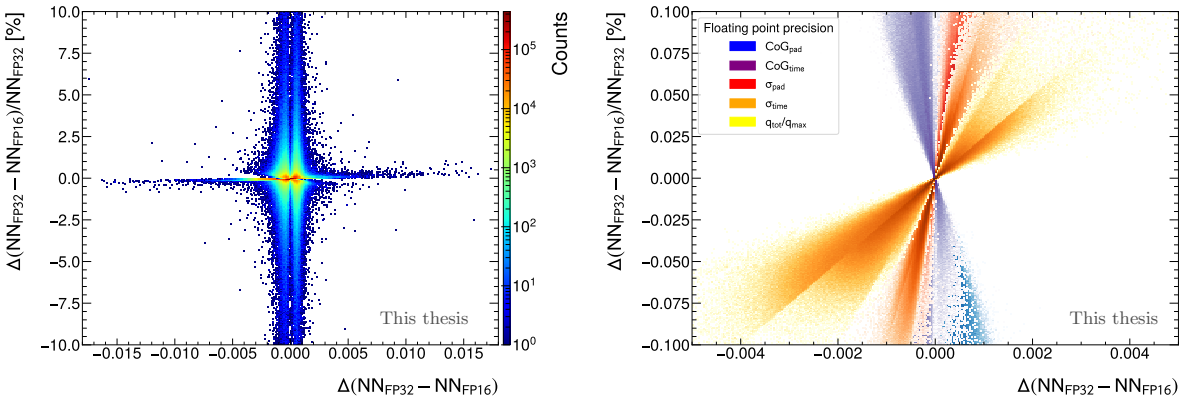


Fig. 5.2.6: Relative difference in percent as a function of the absolute difference for the full flattened array of outputs (left) and enlarged for the individual outputs (right).

5.2.4 Convolutional versus fully connected layers

As shown in the MC studies, no significant improvement in physics performance is found between the fully connected and convolutional architecture. The computational speed between both model architectures is compared in order to draw conclusions for the availability for online productions. The choice of architecture and size rely on several iterations and (physics) performance investigations on data. The model architecture which showed most promising results for the CNN-based architecture is shown as a graph in figure (5.2.7) and is benchmarked against the fully connected architecture.

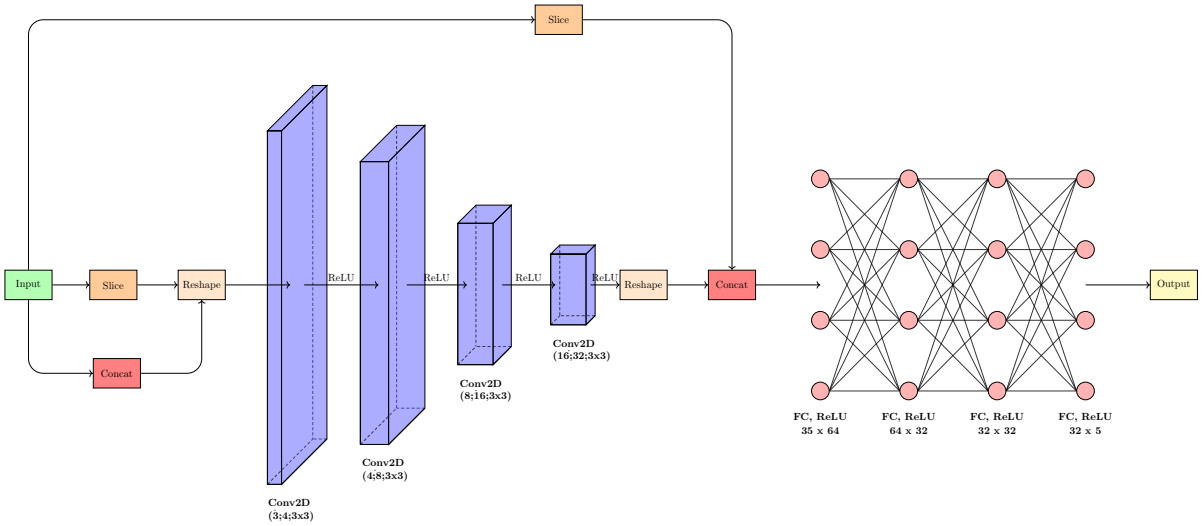


Fig. 5.2.7: Network architecture of the convolutional neural network. Convolutional layer captions are labeled as (input channels; output channels; kernel width×kernel height).

An architecture similar to the VGG networks for image classification is chosen as the baseline model. The CNN layers can only process regular size matrices ("images"). Therefore, fully connected layers need to be incorporated after the CNN layers to take the normalized (sector,

row, pad) coordinates into account. The performance of this model is then compared to the fully connected model shown in figure (5.2.8).

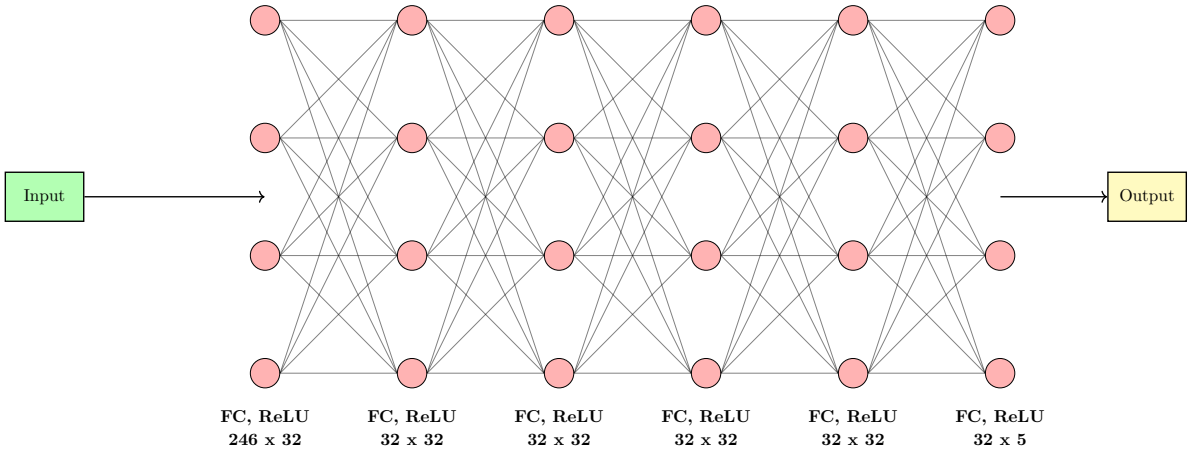


Fig. 5.2.8: Network architecture of the fully connected neural network.

Structurally, the fully connected network shows a significantly reduced complexity compared to the CNN graph. While the (in-)difference in physics performance was illustrated in the previous chapters, the difference in compute performance is of major interest for the final deployment strategy. Both models are benchmarked on the same input tensor batchsize of 262144 (= 2^{18}) for a set of 1000 runs on several GPU models. The results on GPU compute time, memory usage and power consumption by each model are listed for both single and half-precision floating point operations are listed in table (5.2.4).

	FC (FP32)	CNN (FP32)	FC (FP16)	CNN (FP16)
Compute time per batch [ms]				
(GPU) NVIDIA RTX3090	2.2 ± 0.2	10.6 ± 0.6	0.94 ± 0.04	8.3 ± 0.4
(GPU) NVIDIA RTX5080	0.934 ± 0.002	7.696 ± 0.007	0.467 ± 0.002	7.56 ± 0.01
(GPU) NVIDIA H100	0.76 ± 0.05	4.89 ± 0.03	0.42 ± 0.06	4.71 ± 0.03
(GPU) NVIDIA L40S	0.98 ± 0.07	4.21 ± 0.06	0.46 ± 0.07	4.36 ± 0.07
(GPU) AMD MI50	1.477 ± 0.011	-	1.054 ± 0.013	-
(GPU) AMD MI100	0.969 ± 0.005	-	0.585 ± 0.023	-
(GPU) AMD Radeon VII	1.99 ± 0.02	-	1.238 ± 0.016	-
(GPU) AMD Radeon Pro W7900	1.11 ± 0.02	4.22 ± 0.14	0.48 ± 0.03	1.80 ± 0.45
(GPU) AMD MI300X	0.357 ± 0.004	3.73 ± 0.04	0.202 ± 0.003	2.62 ± 0.06
(CPU) AMD EPYC 7452	62.8 ± 1.3	88.8 ± 6.7	163 ± 32	1887 ± 11
Memory usage (H100) [MB]	132	1028	66	514

Table 5.2.4: Measurements of compute time and memory usage on different GPU architectures[3] for fully connected (FC) and convolutional (CNN) networks in single (FP32) and half (FP16) precision, measured over 1000 batches with the batch size used for online deployment (262144) and an input size of (3,9,9). The color code is normalized per column and indicates the fastest (green) to slowest (red) models.

To reduce the error on the measurement, 10 warm-up runs are performed before each measurement is started. Additionally, the first measurement point is excluded from the calculation as it typically shows the largest deviation. IO-binding is used to avoid additional data copies. The evaluation of GPU time only accumulates the time used by the network evaluation. Memory transfers of the input data are excluded from the calculation but can significantly increase the computing time.

The memory usage is measured as the difference between the minimum and maximum memory values measured over all runs. It amounts to the internal allocations performed by each layer to store intermediate results. It is seen that the fully connected graph achieves a significantly improved compute speed and memory usage compared to the CNN architecture. Limiting factors for the convolution layers are memory operations, such as reshaping, slicing and concatenations of the outputs, as well as sparse thread allocations. With increased compute performance for float16 tensor operations, modern server-grade GPUs (e.g. NVIDIA H100, NVIDIA L40S, AMD MI300X) show clear performance benefits from single (float32) to half (float16) precision, typically by a factor of approximately 2. While the MI300X is the most modern card of the tested ones, it is expected that newer models of NVIDIA graphics cards, which were not available for the test, will achieve similar results (e.g. H200, B100). Likewise, there are measured differences between cards on different hosts that should give identical results by the manufacturers specifications (MI50 and Radeon VII are identical in core count, memory and memory bandwidth). These differences could either stem from ONNX internal allocations / computations or differences of the host system. While memory copies are excluded from this measurement (using prior host-to-device copies and I/O binding features of the ONNX runtime framework), internal operations of the framework are not known or further investigated as they go beyond the scope of this work. With the increased flexibility of the fully connected model concerning the input tensor, the improvement in speed and near identical physics performance (within the margin of error), the fully connected model is chosen for the final deployment strategy in online processing.

[3] NVIDIA RTX3090: https://www.nvidia.com/en-us/geforce/graphics-cards/30-series/rtx-3090-3090ti/
NVIDIA RTX5080: https://www.nvidia.com/en-us/geforce/graphics-cards/50-series/rtx-5080/
NVIDIA H100: https://www.nvidia.com/en-us/data-center/h100/
NVIDIA L40S: https://www.nvidia.com/en-us/data-center/l40s/
AMD MI50: https://www.amd.com/en/support/downloads/drivers.html/accelerators/instinct/instinct-mi-series/instinct-mi50-32gb.html#amd_support_product_spec
AMD MI100: https://www.amd.com/en/products/accelerators/instinct/mi100.html
AMD Radeon VII: https://www.amd.com/en/support/downloads/drivers.html/graphics/radeon-rx/radeon-rx-vega-series/amd-radeon-vii.html
AMD Radeon Pro W7900: https://www.amd.com/en/products/graphics/workstations/radeon-pro/w7900.html
AMD MI300X: https://www.amd.com/en/products/accelerators/instinct/mi300/mi300x.html
(CPU) AMD EPYC 7452, 32-core: https://www.amd.com/en/support/downloads/drivers.html/processors/epyc/epyc-7002-series/amd-epyc-7542.html

A 32-core CPU execution is shown to demonstrate the optimized parallel throughput on GPUs for neural network calculations. Compared to the GPU evaluation in single precision, the parallel CPU evaluation is approximately 176 times slower than the fastest GPU in the benchmark (AMD MI300X) and approximately 29 times slower than the slowest GPU (NVIDIA RTX3090). The CPU benchmark was performed on 50 samples compared to the 1000 samples for the GPU cases. It can also be seen that half-precision computation drastically reduce throughput on CPU compared to single precision evaluations due to the lack of native float16 support on the used AMD CPU (this will only be available from the AMD EPYC Zen6 generation onwards). The average utilization of the CPU threads was observed to be only 10-20% during the half precision execution.

While the used software stack does not support model compilation for the CNN on older graphics cards, comparison of the fully connected networks is more relevant for deployment purposes. Modern hardware accelerators, like the server-grade NVIDIA H100 graphics card, offer additional compute speed improvements due to dedicated hardware acceleration for tensor operations and dense matrix-matrix multiplications. With the overall increased number of CUDA cores (threads) in modern GPUs, the throughput (elements/s) is even more efficient when the GPU is fully loaded (i.e. all / most threads perform concurrent operations).

5.2.5 Compute speed evaluations in online processing

The standard to which the new cluster finding scheme has to adhere to is the evaluation in online processing. The networks chosen in previous analyses for real data evaluations and execution in the online runs are also chosen in this case to perform evaluations on the full processing farm. These tests were conducted in the machine development phase before the online runs (beginning of October 2025) and represent a direct measurement of the compute speed under full load. At the time of execution, compilation issues (unrelated to this work) only allowed for serialized execution on MI100 GPUs (single compute stream). Hence, only the MI50 compute speeds are reliably comparable. The runs were conducted on replay, pp (500 kHz) and Pb–Pb (50 kHz) data. In nominal data taking conditions, certain margins of compute capacity are kept to account for data size fluctuations or otherwise unexpected changes of the data taking conditions. However, to reliably benchmark the computing speed of a given algorithm, all nodes must operate under full load (to the best possible approximation). This is ensured by reducing the number of compute nodes until input buffers fill up and input data inevitably needs to be dropped. Both configurations (pp and Pb–Pb) were therefore run with a reduced number of event processing nodes to artificially create such backpressure. CPU overhead is shadowed by the compute load on the GPUs, which is the main bottleneck. The results are summarized in table (5.2.5).

Beamtype	Configuration	#EPNs	TF processing rate	×default reco.
pp, 500 kHz	heuristic (default)	45	3.482	× 1
	NN class. + CF reg.	101	2.574	×1.35
	NN full	101	2.068	×1.68
Pb–Pb, 50 kHz	heuristic (default)	242	0.939	× 1
	NN class. + CF reg.	233	0.764	×1.23
	NN full	233	0.679	×1.38

Table 5.2.5: Measurements of compute time on MI50 graphics cards in online processing for two configurations in comparison to the default cluster finding method.

The evaluation was performed on the networks with an input charge array of (3,9,9) with a batchsize of 262144. Statistical errors for this measurement are taken as the standard deviation of all processing rates over all EPNs and are ≤ 0.001 for the timeframe processing rates reported in the table. With an increase of compute time by 38%, the results match well with the previous evaluations on a single EPN node (see appendix, figure (D.0.23), configuration: (262144, (L4, N32))), showing that the initial tests scale identically to the full farm. In all cases the network cluster finding increases the computational load. The resulting effect on the reconstruction time is up to 68% increase in the low-rate pp runs. While this is not a bottleneck in this case, as these configurations require less computational resources than the dense Pb–Pb environments, the decrease in Pb–Pb processing speeds is the major bottleneck for the application of this algorithm during Run 3 of the LHC. From the measurement it can clearly be stated that the network evaluation benefits from denser environments as the evaluation is more efficient when all input data is filled and evaluated with all GPU threads maximally loaded. The network evaluation therefore does not become more resource demanding in absolute terms, but operates in comparison less efficient with pp data than with Pb–Pb data, due to the substantially reduced occupancy per unit time (sparsity of data).

5.2.6 Compute time evaluations of individual processing steps

The investigations in online processing only allow for an estimation of the wall time and overall compute time increase. However, the reduction in number of total clusters results in differences for all steps outside the cluster finding process. An overview of the different steps is shown in table (5.2.6). The quoted times are the average evaluations per timeframe, the errors are taken from the standard deviation of the recorded times.

	GPU CF [s]	NN (FC, (3,9,9), 0.05) [s]	Ratio: NN / GPU CF
wall time	22.28 ± 0.03	31.3 ± 0.2	1.40 ± 0.01
cluster finding	3.54 ± 0.01 (15.9%)	14.0 ± 0.2 (44.8%)	3.96 ± 0.05
sector tracking	6.60 ± 0.01 (29.6%)	6.24 ± 0.01 (20.0%)	0.95 ± 0.01
track-fit & merging	8.73 ± 0.02 (39.2%)	7.88 ± 0.01 (25.2%)	0.90 ± 0.01
compression	3.39 ± 0.01 (15.2%)	3.11 ± 0.01 (10.0%)	0.92 ± 0.01

Table 5.2.6: Times per TF for the two algorithmic settings and as a ratio. The measurement is performed on 8 identical time frames of minimum bias, 50 kHz Pb–Pb simulated data on one EPN equipped with 8 MI50 GPUs.

Wall time refers to the total processing time and cluster finding to the accumulated time of all cluster finding steps including the neural network evaluations. Due to the reduced number of clusters with the neural network, tracking (sector tracking and track-fit & merging) and data compression times improve, while the cluster finding step strongly increases in compute time compared to the default reconstruction case (GPU CF). The increase in time of the cluster finding step then results in the overall increased wall time of the reconstruction process. The measurement of the wall time coincides with the measurement on the full farm (see table (5.2.5), case: Pb–Pb, 50 kHz, NN full).

5.3 Loss landscape of the neural network

For stable operations and adaptability of the neural network to changing conditions it must be ensured that training the algorithm results in similar performances, stable over several training runs. While it is not expected that the algorithm needs retraining, it should be a feature of the designed framework to cope with frequent demands on retraining or adaptation, especially in the first phase of deployment. This is ensured by conducting the Monte Carlo studies from the previous sections after each training, but also by investigating the convergence behavior of the neural network itself. For this purpose the shape of the loss landscape of the network is studied. This study orients itself on the visualization of the loss-landscape approximations conducted in different fields of research on machine learning, most prominently for ResNet [57]. Of particular interest is the property of convexity of the loss landscape. While this property is not directly accessible due to the high dimensionality of the input space, the loss landscape can be visualized by using a subset of the training data. For this purpose the NN with input (3,9,9) is analyzed using a dataset shifted by the first two principal components of the training data. The optimization problem for principal component analysis (PCA) consists of finding the vectors in $\mathbb{R}^D$, where D is the dimensionality of the phase space under investigation (here $D = 3 \cdot 9 \cdot 9 + 3 = 246$), such that the variance of the data projected onto the principal component is maximized.

Annotation: Principal Component Analysis

Problem Statement: Given a centered data matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$, PCA finds directions $\mathbf{w} \in \mathbb{R}^d$ that maximize variance under a unit-norm constraint.

$$\arg\max_{\mathbf{w}} (\mathbf{w}^\top \mathbf{S}\mathbf{w}) \quad \text{subject to} \quad \|\mathbf{w}\| = 1, \tag{5.3.1}$$

where $\mathbf{S} = \frac{1}{n}\mathbf{X}^\top\mathbf{X}$ is the sample covariance matrix.

Solution: This is a constrained optimization problem that leads to an eigenvalue equation:

$$\mathbf{S}\mathbf{w} = \lambda\mathbf{w}. \tag{5.3.2}$$

The top k eigenvectors $\mathbf{w}_1, \ldots, \mathbf{w}_k$ corresponding to the largest eigenvalues $\lambda_1 \geq \cdots \geq \lambda_k$ define the principal directions.

PCA Projection: Project data to lower dimensions:

$$\mathbf{Z} = \mathbf{X}\mathbf{W}, \quad \text{where } \mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_k] \in \mathbb{R}^{d \times k}. \tag{5.3.3}$$

This method is a common, unsupervised machine learning technique to describe feature importance and correlation in a dataset and to reduce its dimensionality to the most descriptive dimensions. In case the first $n \ll d$ principal components have significantly larger associated eigenvalues compared to the rest, the dataset is well described with a lower dimensional phase space while maintaining a significant portion of the information contained in the original dataset. This technique is also commonly used for an initial data compression before utilizing neural networks, by projecting the dataset to a lower dimensional space and evaluating more complex algorithms only on the subspace. However, in most cases this is not a lossless compression leading to potentially suboptimal results. Each principal component is a vector in the vector-space spanned by the input dimensions and is typically not directly interpretable. At first, the principal components of the dataset are determined and sorted by the size of their eigenvalues in descending order. The eigenvalues are afterwards normalized and give a measure of the "explained variance" for each component. The higher the eigenvalue, the better the associated principal component describes the dataset. Figure (5.3.9) shows the explained variance histogram of the first 20 principal components of the utilized dataset.

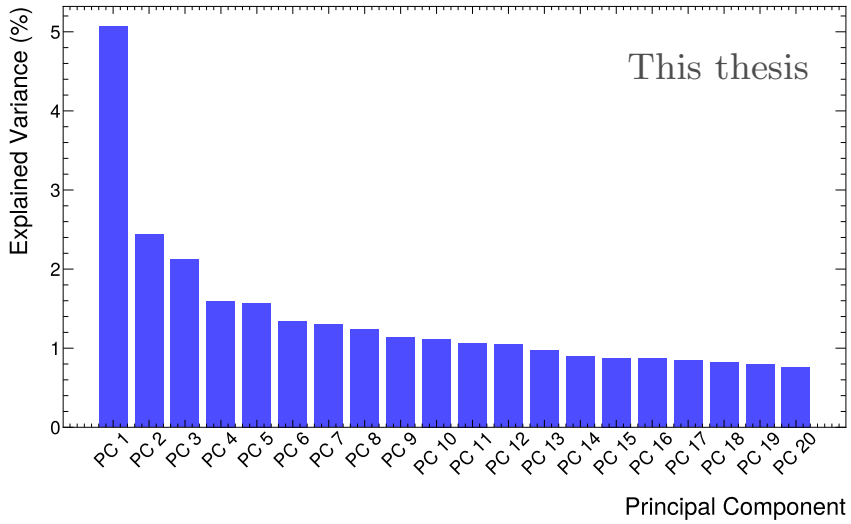


Fig. 5.3.9: Explained variance for the first 20 principal components of the full training dataset.

The first principal component accounts for $\approx 5\%$ of the explained variance in the dataset, while each of the principal components after PC5 explain $< 1.5\%$ of the variance. This mild decay in the explained variance over increasing principal components illustrates, that the dataset cannot easily be compressed into a lower dimensional subspace. Accumulated, the first 2 / 5 / 10 / 20 principal components account for 7.5% / 12.8% / 18.9% / 27.9% of the explained variance. This implies that the dataset is intrinsically high-dimensional (i.e. many of the digits of the input charge array carry important information and are hence not obsolete).

This information is directly used to investigate the behavior of neural networks in the two-dimensional subspace. A subset of 1024 elements, randomly sampled from the training dataset is linearly transformed using the first two principal components to explore the loss surface, which the network optimizes in the training procedure. The linear transformation is conducted as

$$x_i^{\text{new}} = x_i^{\text{orig}} + \alpha_1 \cdot \vec{v}_{\text{PCA},1} + \alpha_2 \cdot \vec{v}_{\text{PCA},2} \tag{5.3.4}$$

where $x_{\text{new}} \in \mathbb{R}^{1024 \times D}$ is the transformed data-matrix of the selected data and x_{orig} the original dataset without PCA transformation. Figure (5.3.10) illustrates a wide angle view of the loss surface with large scale-factors for α_1 and α_2. The color and z-axis indicate the value of the logarithm of the mean squared error loss function. The scan was performed for the regression neural network with an input size of (3,9,9) without momentum vector estimation.

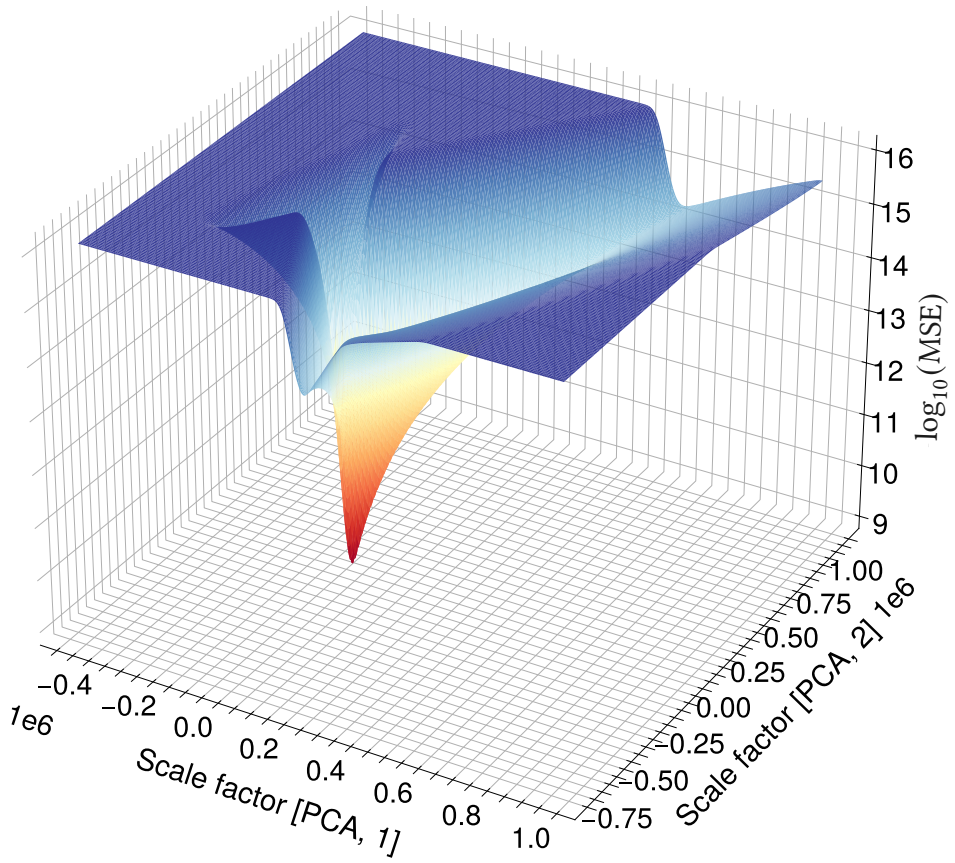


Fig. 5.3.10: Loss surface of the regression neural network along the first two principal components of the training dataset with cubic spline interpolation between points.

The fine-grained grid structure used to investigate the loss surface indicates that no major fluctuations are observable at this scale. This is beneficial for the gradient descent computation as it makes the descent into the minimum of the loss surface both smooth (i.e. without encountering many local saddle points or minima) and the result reliable. A localized scan of the loss surface is illustrated in figure (5.3.11). Cubic splines are utilized to interpolate the loss landscape to visualize the data as a continuous hypersurface.

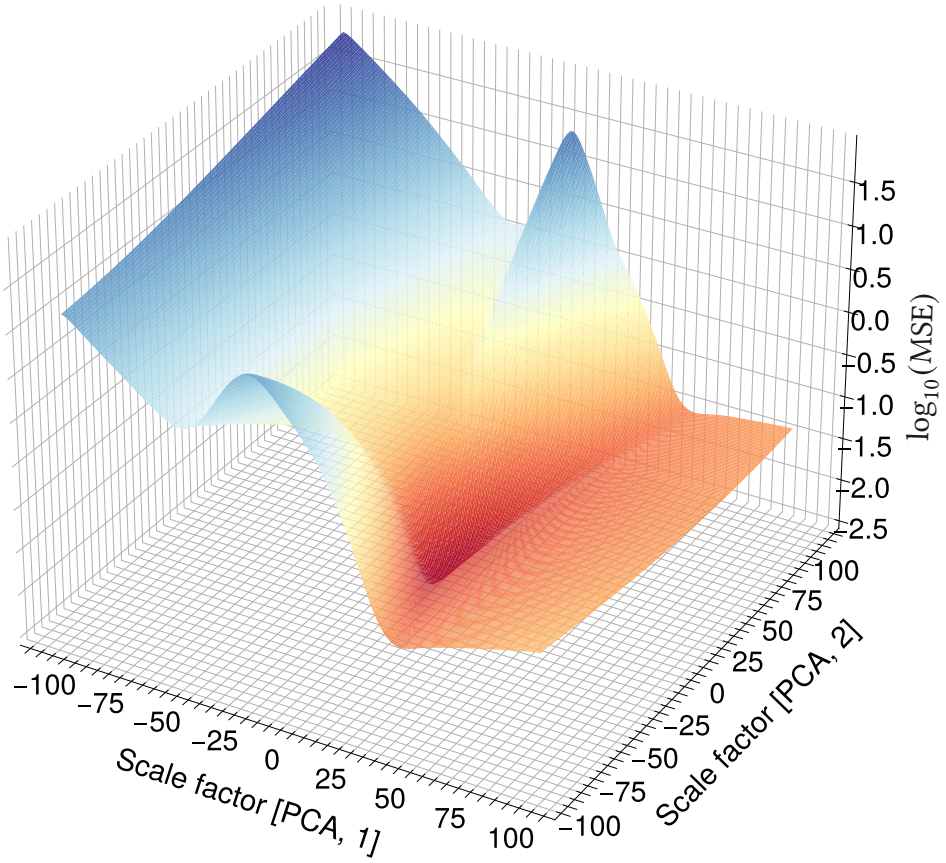


Fig. 5.3.11: Localized scan of the loss surface of the regression neural network with cubic spline interpolation.

This illustrates that the descent into the global minimum of the loss surface obeys local smoothness visually (in the two-dimensional subspace). The structure of the loss landscape does not significantly change with even smaller scale factors for the principal components and stays smooth even at smallest scales of 10^{-2} for both scale factors (α_1 and α_2). The main explanation for such well-behaved optimization landscapes is the fact that the utilized neural network is rather small compared to state-of-the-art, real-world networks (such as the ones used for the ImageNet challenge and beyond). With only four hidden layers of 32 neurons each, the network minimizes the computational cost for online processing, memory consumption and exhibits a locally smooth optimization behavior. The change of local gradients can then further illustrate the existence of local saddle points or non-convex areas.

Annotation: The Hessian matrix

Convexity: $f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y) \quad \forall x, y \in \mathbb{R}^n,\ \lambda \in [0,1]$.

If f is differentiable, then the convexity statement is equivalent to

$$f(y) \;\geq\; f(x) + \nabla f(x)^{\top}(y - x) \quad \forall\, x, y. \tag{5.3.5}$$

Therefore the function always lies above the tangent plane defined by the function value at x and the derivative at that point. If f is then twice differentiable, this statement is converted into

$$\forall\, z \in \mathbb{R}^n, \quad z^{\top}\nabla^2 f(x) z \geq 0. \tag{5.3.6}$$

Hence the Hessian matrix (using the notation $f_x = \partial f / \partial x$)

$$\nabla^2 f(x, y) = \begin{bmatrix} f_{xx} & f_{xy} \\ f_{xy} & f_{yy} \end{bmatrix} \tag{5.3.7}$$

is used to identify the local curvature behavior in both x and y directions. A positive definite hessian matrix (all eigenvalues are positive: $f_{xx} > 0 \wedge \det(\mathrm{H}) > 0$) implies convexity of the function f at the evaluated point. Negative definiteness implies concavity (all eigenvalues are negative: $f_{xx} < 0 \wedge \det(\mathrm{H}) > 0$). If $\det(\mathrm{H}) \geq 0$ the behavior of f is semi-definite (either convex, concave or flat). In case $\det(\mathrm{H}) < 0$, the function is indefinite and assumes a saddle point shape. In other words, the determinant of the hessian measures if the curvature directions agree ($\det > 0$) or disagree ($\det < 0$).

For an analysis of the sign and magnitude of the hessian matrix, the loss surface is interpolated using smooth bivariate splines. The sign of the determinant of the hessian matrix then indicates the curvature behavior, while the magnitude quantifies the magnitude of local curvature (i.e. the steepness of the spline interpolation). In this way many more details are highlighted in comparison to the surface plot of the loss values. Figure (5.3.12) illustrates the absolute magnitude of the determinant of the hessian matrix determinant on the spline interpolation and figure (5.3.13) the sign of the determinant of the hessian matrix indicating the definiteness.

While the figures show that close to the desired minimum (scale-factors < 5 units in each direction), the determinant of the hessian matrix is greater than 0 (and therefore definiteness is guaranteed), further away, many fluctuations are observed in the loss surface than are visible from the 3D surface plot (figure (5.3.11)). The smoothness of the figures above also depends on the regularization of the spline interpolation, while the overall magnitude of the determinant of the hessian is $\leq 10^{-4}$ indicating only very small variations in the local gradients. Repeating this study with various smoothness settings for the splines did not remove the visible structures completely. The small saddle points encountered in the loss-surface are easily overcome by the step width (learning rate) of the gradient descent algorithm and by the stochastic nature of the batch sampling. The plateau at positive values of the scale factor of the

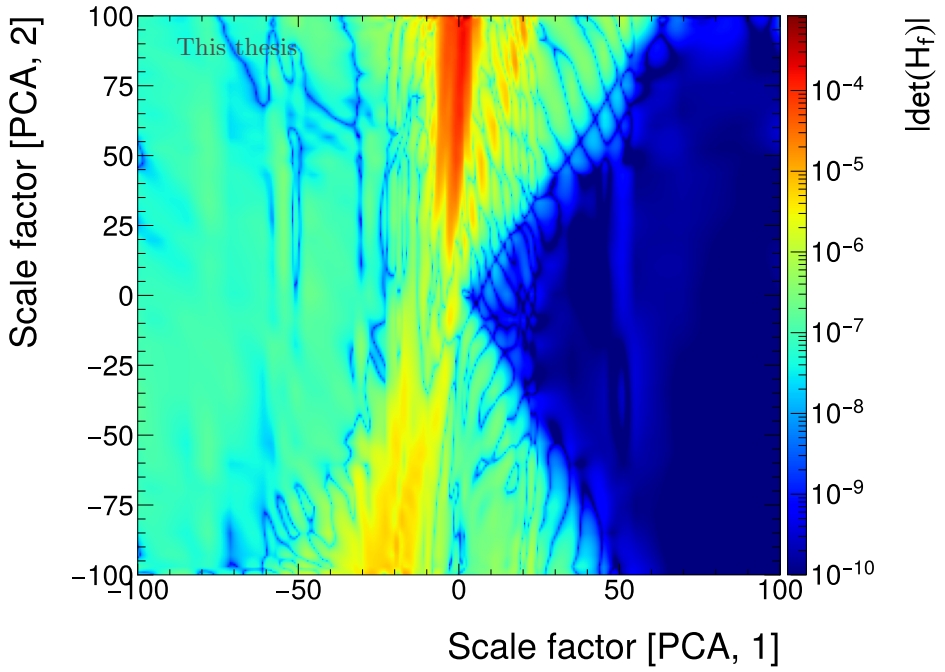


Fig. 5.3.12: Absolute value of the determinant of the hessian matrix evaluated locally on the smooth bivariate spline interpolation.

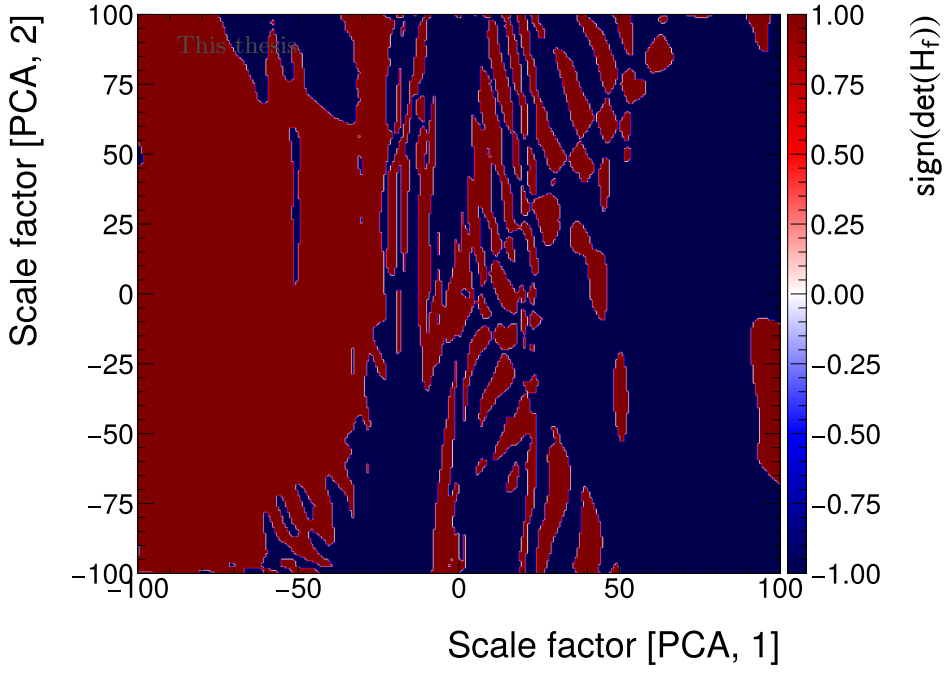


Fig. 5.3.13: Sign of the hessian determinant evaluated on the directions spanned by the principal components.

first principal component assume very small values, indicating that the hessian is (to the best approximation) indefinite indicating a near constant behavior of the curvature in both direc-

tions. Batchsize scheduling was applied at various epochs throughout the training process to ensure a consistent descent of the algorithm into the global minimum, minimizing the likelihood of convergence on a saddlepoint. As a result it can be stated that, if the network structure is as such sufficient to perform the task at hand (i.e. cluster regression), then the optimization will result in a loss near the global minimum. This makes the training procedure robust and the result of the network stable in case of retraining, provided a model of comparable size is used, and the phase space is filled (sampled) similarly.

5.4 The economic aspect

While it is not a typical part of a physics thesis to have monetary interests in mind, the newly developed algorithm holds high potential for the reduction of cost and energy by reducing the number of stored clusters. This estimate neglects the costs for event processing nodes entirely as it is not clear how much the neural networks and the reconstruction process (in terms of wall-time) can profit from the hardware acceleration that is to be expected in modern GPUs. It relies on cost estimates made in accordance with experts upon request to the CERN IT department

- Cost of 1 PB of disk-space per year:
  - Industry standard (example: AWS, Amazon Web Services 2025, standard plan): $\approx 2.76 \times 10^5$ USD/PB/year[4]
  - CERN internal storage: 22 CHF/GB (CERN IT department, November 2025) for the lifetime of disk (typically 4-5 years) $\rightarrow \approx$ 5000 CHF/PB/year
- Disk buffer allocated in 2025: 180 PB

For the lowest cluster rejection scenario of 10% saved disk space, the monetary savings per year (C) are therefore calculated as

$$C = 0.1 \times 180 \times 5000 \text{ CHF/year} = 90000 \text{ CHF/year}. \tag{5.4.1}$$

In the best case scenario, this number doubles to 1.8×10^5 CHF/year. This calculation hides opportunity costs associated with storage maintenance and electricity cost, but represents an estimate on the final budget that has to be allocated by (or in this case is saved for) the collaboration. Additionally, the deployment of this algorithm also comes with an increase in physics performance. Cost increases of hard drives in the future are not taken into consideration here.

[4]Taken from https://www.cloudzero.com/blog/s3-pricing

5.5 Summary

This chapter investigated many, sometimes orthogonal, topics concerning computing infrastructure and design choices for the feasibility of deployment of the neural network cluster finding algorithm. On CPU-based systems, mainly relevant for Monte Carlo productions, fully connected networks increase compute time for the full reconstruction process by a factor of 2.1 compared to the previous cluster finding method. The comparison of fully connected and convolutional layers showed that only fully connected layers are feasible for deployment. Additionally, the half precision networks (float16) increase CPU compute time drastically. The float32 models are preferable in this case.
On GPU the computing time of the reconstruction time of the full system test benchmark is increased by a factor 1.4, for the case of nominal 50 kHz Pb–Pb collisions, in the online farm. Different model sizes were tested, where the fully connected network with 4 layers and 32 neurons per layer was chosen for deployment. Half-precision floating point operations further increased compute efficiency with a minimal trade-off in precision. For future considerations, different GPU models were benchmarked with the AMD MI300X being overall the fastest tested hardware. Finally, a qualitative analysis of the neural network loss landscape was conducted to investigate training stability. Additionally, cost saving, due to the reduced number of clusters in online processing, was estimated which is relevant as an operational aspect. This implies that the neural network cluster finder is computationally feasible for low interaction rate Pb–Pb runs within Run 3 and becomes a strong candidate for Run 4, under certain hardware and precision constraints.

VI Summary and conclusion

This project aims to introduce a novel cluster finding algorithm for the TPC reconstruction in the ALICE experiment at CERN. All steps of the production procedure were illustrated in this thesis, with an emphasis put on the strengths and weaknesses of the algorithm and the challenges that were encountered and solved throughout the work performed in the ALICE reconstruction software. Dedicated implementations and studies for the creation of ideal Monte Carlo clusters were explained in the first part of this thesis (see chapter (II)). The study of its effect on matching efficiencies and fake rates illustrate how efficient training data is created and which sampling steps were taken to ensure an improved phase space coverage for the training process of the neural network. The trained networks were then rigorously tested against Monte Carlo ground truth and the current reconstruction algorithm for simulated (chapter (III)) and real data (chapter (IV)). Improvements in the efficiencies and fake-rates were demonstrated on primary and secondary particles (figure (3.4.18)) in high-density environments of central (0-5%), Pb–Pb collisions. In addition, the usage of the regression neural network demonstrated an improvement in the χ^2/NDF distribution (figure (3.4.23)) and on the $\mathrm{d}E/\mathrm{d}x$ separation power of pions and electrons at the MIP position (figure (4.2.5)). Ultimately the physics performance was shown to be maintained for identified V^0 samples (K^0_S: figure (4.4.11), and Λ: figure (4.4.12)) and improved for the mass resolution of the D^0 decay (figure (4.4.16)). Studying the classification threshold setting of the cluster classification neural network revealed major data savings of up to 18% on reprocessed real data (table (4.4.5)), which was verified in two commissioning runs of 500 kHz pp collisions conducted in the online reconstruction of the experiment (table (4.5.7)). This not only validated the feasibility of deployment of this algorithm, but also marks the first application of machine learning in online data taking for ALICE and one of the first of the four major LHC experiments. These commissioning runs provided a comparison for the performance of the algorithm on centrally reconstructed, low-rate pp data, which showed beneficial effects on the $\mathrm{d}E/\mathrm{d}x$ separation power (see figure (4.5.20)) and shared cluster maps (see figure (4.5.22)) compared to the default cluster finding method. This marks the major achievement of this project, exceeding the scope of deployment which was intended for Run 4 and beyond. The successful commissioning and validation of the expected performance increase and data reduction validate the prior studies and complete the implementation of the algorithm.

The technical implementation was a significant part of the work performed within this project. Challenges were substantially encountered when implementing the ONNX runtime framework with GPU support for NVIDIA and AMD GPUs into the O2 software stack, including inference time and kernel optimizations. Compute performance was further increased by changing the precision of the networks to 16-bit floating point operations with native hardware acceleration in most modern GPUs. Performance benchmarks of different neural network model types were conducted on a variety of hardware accelerators (see table (5.2.4)) to assess the computing speed and memory consumption. Such benchmarks are part of the decision-making process for hardware accelerators considered for future upgrades of the EPN farm in long shutdown 3.
Many tangents to this project exist which rapidly increase the scope and scale for investigation. The following improvements are of interest, but are only partially answered in this work and require substantially deeper investigations.

Impact of this new cluster finding algorithm on all other calibrations (mostly prominently the space-charge distortions)
This can ultimately only be answered using real data, taken with the experiment due to insufficient knowledge and understanding of the effects and their application in simulation. However, the space-charge distortion calibration typically requires at least several seconds of continuous data taking in high-density environments to allow for a recalibration, which is not available at the time of writing. The effect of cluster regression is not considered to be large on the fitted distortion maps but the reduction in number of clusters and noise tracks has the potential to improve performance.

Additional effects observed in physics analysis
Using the new cluster finding algorithm a brief analysis on the D^0 spectrum was shown in the thesis (see figure (4.4.16)). The question must be asked if other probes, sensitive to the reconstruction process, benefit from this new algorithm and the improved separation power with TPC or if there are probes which require further tuning of the algorithm. This requires a collaboration wide effort of validation and ultimately needs to be conducted on real data. Necessary steps in this direction are taken in accordance with physics coordination of the experiment but are not expected to converge before the end of Run 3.

Optimization of computational resources
As it was experienced throughout the conducted work in this thesis, an optimization of the computational utilization can be done in almost all cases, even when it is thought that a given function is implemented in the most optimal way. This project is no exception. With a significant interest for this algorithm within the collaboration, the increase in manpower by the

central reconstruction team is very beneficial to uncover minor issues and optimization of computational resources. While it is topic of investigation how the algorithm could be optimized, the application within Run 3, especially for the heavy-ion run in 2026 is unlikely. However, the design of the new EPN farm dedicated to the online reconstruction in Run 4 and beyond will take machine learning and neural networks into its focus. Especially modern GPUs with tensor processing units can speed up the inference significantly (see table (5.2.4)). A full-scale application of this algorithm for Run 4 and beyond is highly probable.

The future possibilities of this cluster finding method are widespread and not yet explored in all its details (see section (3.5)). The local inclination vector estimation and potential improvements to cluster flags or the cluster error parameterization are only two of the many outlined directions that could form the basis of further investigation. The code implementation further paves the way for future GPU-supported machine learning applications within the ALICE reconstruction framework, representing a methodological shift away from the widely employed heuristic approaches.

To conclude, the scope of this project was achieved and exceeded. The algorithmic implementation is finalized with continuous work conducted to optimize compute and physics performance. This marks the beginning of a new era in ALICE, the era of machine learning in online reconstruction.

A Appendix

A Algorithmic approaches to cluster finding

Kernel / function	Description
Synchronization and transfers	
SynchronizeStream	GPU barrier synchronization.
TransferMemoryResourcesToHost	Transfers GPU buffers to host memory.
Preprocessing and cluster finding	
GPUTPCCFChargeMapFiller	Fills the internal flat buffer with the digits.
GPUTPCCFPeakFinder	Finds local charge peaks (see subsection 2.2.1).
GPUTPCCFNoiseSuppression	Initial stage of detector noise suppression for subthreshold charges.
GPUTPCCFDeconvolution	Assigns split flags and deconvolutes charges according to the presence of peaks in the local 5×5 neighbourhood.
GPUTPCCFClusterizer	Builds clusters using a heuristic method in a local 5×5 neighbourhood of digit charges.
Neural-network path	
fillInputNNCPU (fillInputNNGPU)	Fills the NN charge input array from the flat digit buffer.
mModelClass.inference	Runs inference on the input array using the classification model provided via an ONNX file.
mModelReg1.inference	Runs inference on the input array using the regression model provided via an ONNX file.
determineClass1Labels	For a runtime-adjustable threshold, publishes per-digit-maximum labels: build cluster (1) or not (0).
publishClass1Regression	If a cluster should be built according to determineClass1Labels, publishes the regression output to the internal native cluster buffer.
runCfClusterizer	Wrapper that calls GPUTPCCFClusterizer functions when CF regression is requested.

Table A.0.1: Summary of kernels and helper functions used in the cluster-finding pipeline.

B Monte Carlo

Figure (B.0.1) shows the color-coding of different networks used in the Monte Carlo analysis.

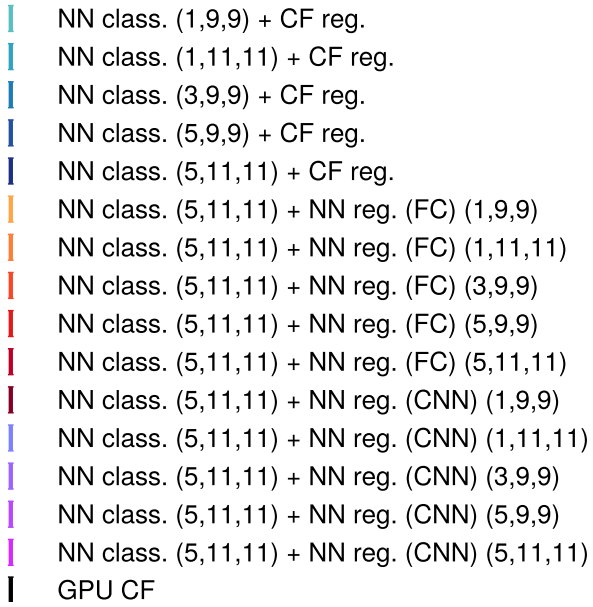


Fig. B.0.1: Legend color coding

B.1 Analysis on pp data

Some excerpts of the analysis on simulated pp data are shown here. The network trained on Pb–Pb data was used to evaluate this dataset: pp, LHC24af apass2, anchor run 550824, 505 kHz interaction rate, 3000 minimum bias events.

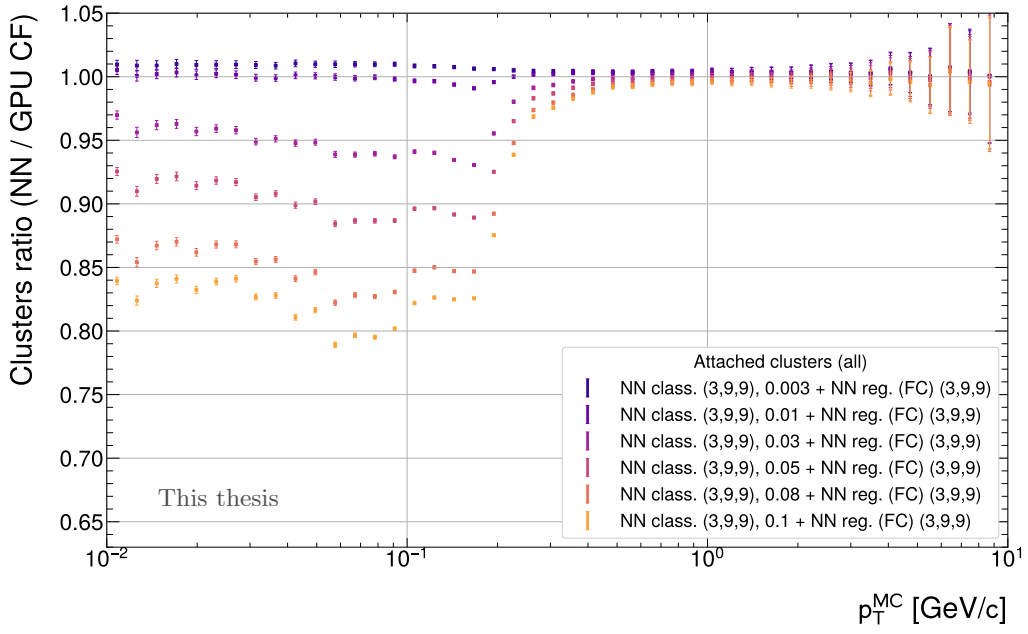


Fig. B.1.2: Ratio of removed clusters using different classification thresholds.

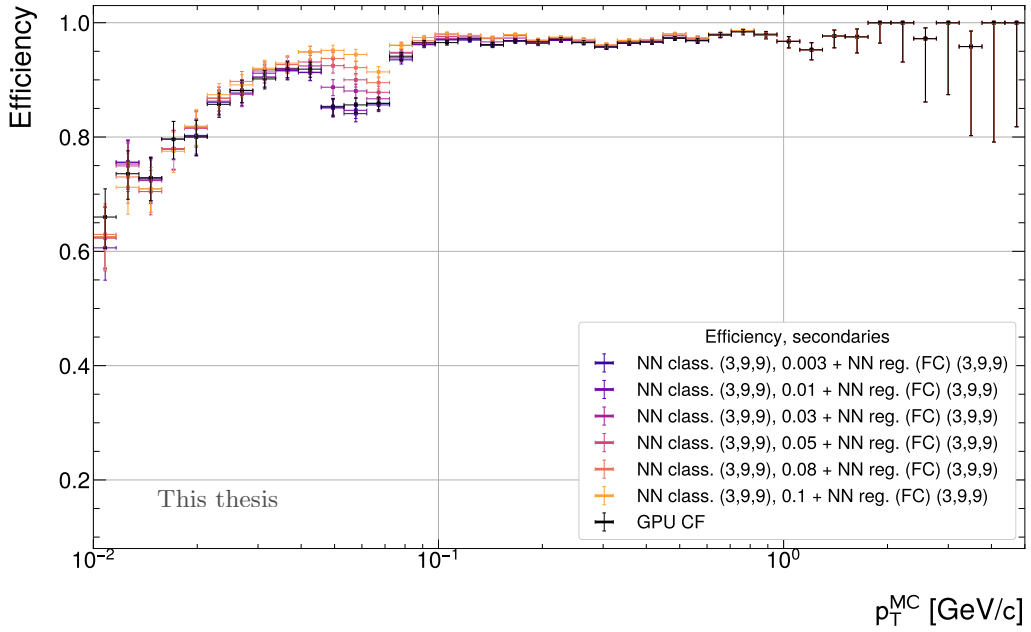


Fig. B.1.3: Tracking efficiency for secondary particles for different classification thresholds.

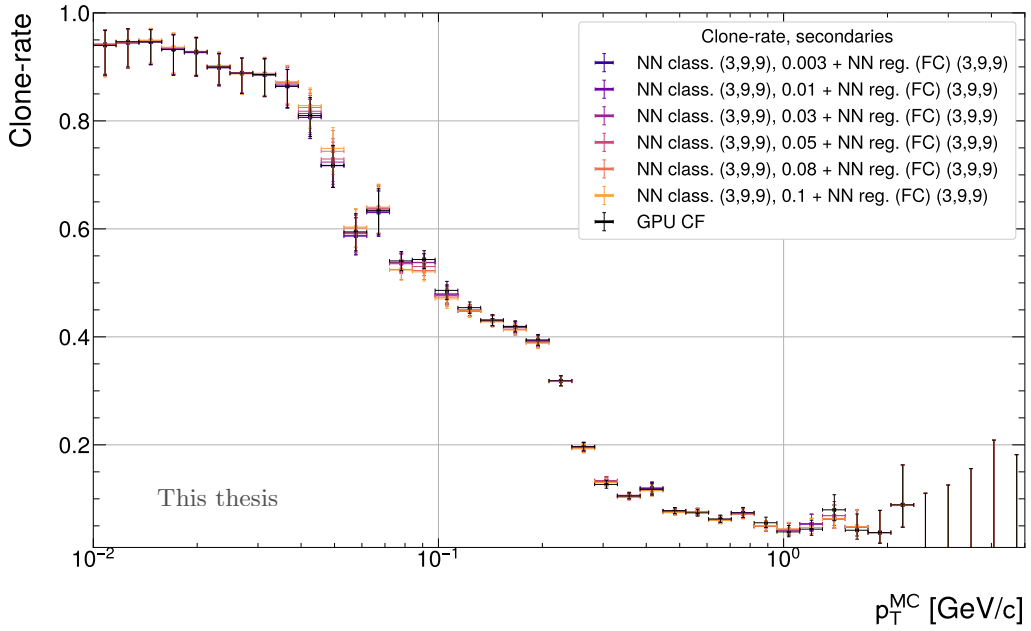


Fig. B.1.4: Clone rate for secondary particles for different classification thresholds.

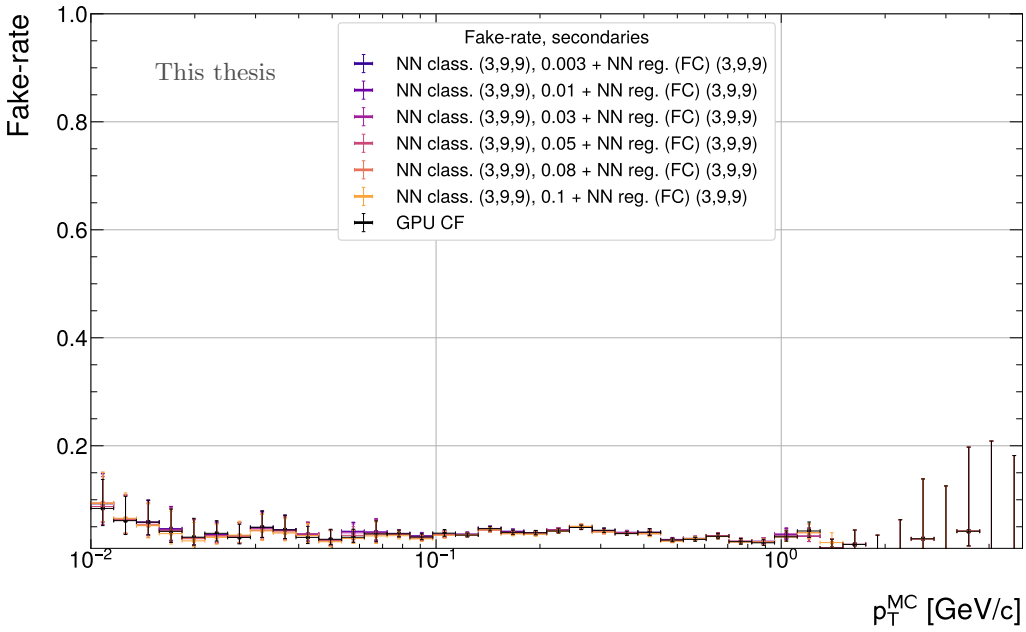


Fig. B.1.5: Fake rate for secondary particles for different classification thresholds.

For low interaction rate pp data, the improvements on efficiency, clone rate and fake rate are significantly less enhanced compared to the investigated cases of Pb–Pb. An enhancement in the efficiency of secondary particles is noticeable which indicates an improvement of tracks by removal of fake clusters. Fake rates and clone rates are compatible within the statistical uncertainties.

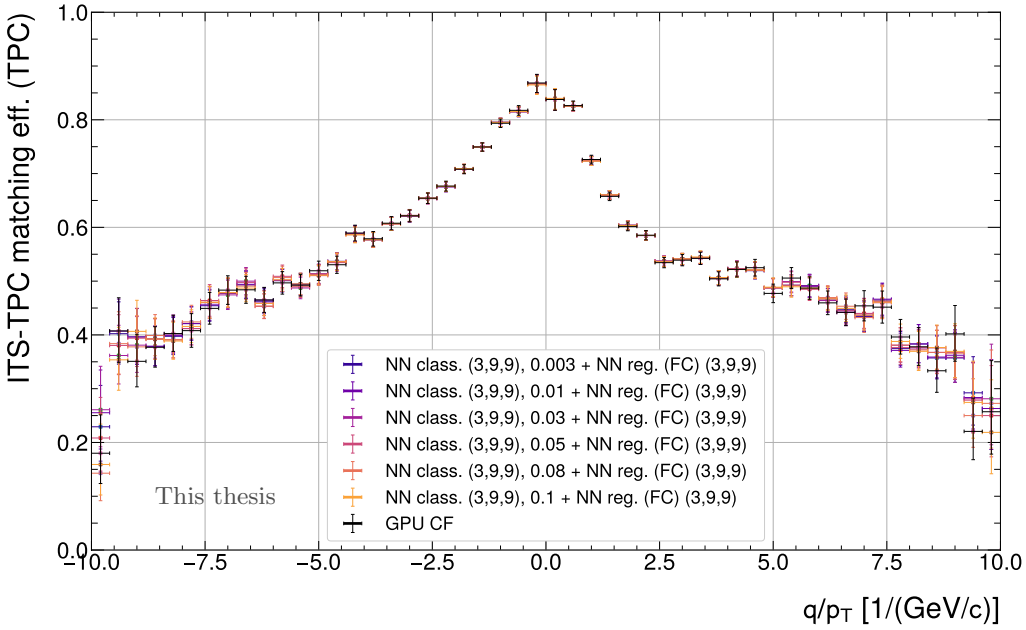


Fig. B.1.6: ITS-TPC matching efficiency (ratio to TPC tracks) as a function of q/p_T.

The matching efficiency shown in figure (B.1.6) between ITS and TPC is maintained with no performance change at higher thresholds.

B.2 Cluster residuals without space-charge distortion effects

Simulation setting: Pb–Pb, 38 kHz anchored, minimum bias, without space-charge distortion effects. Illustrated below are percentile distributions with mean offsets (in case they are present) for all investigated networks.

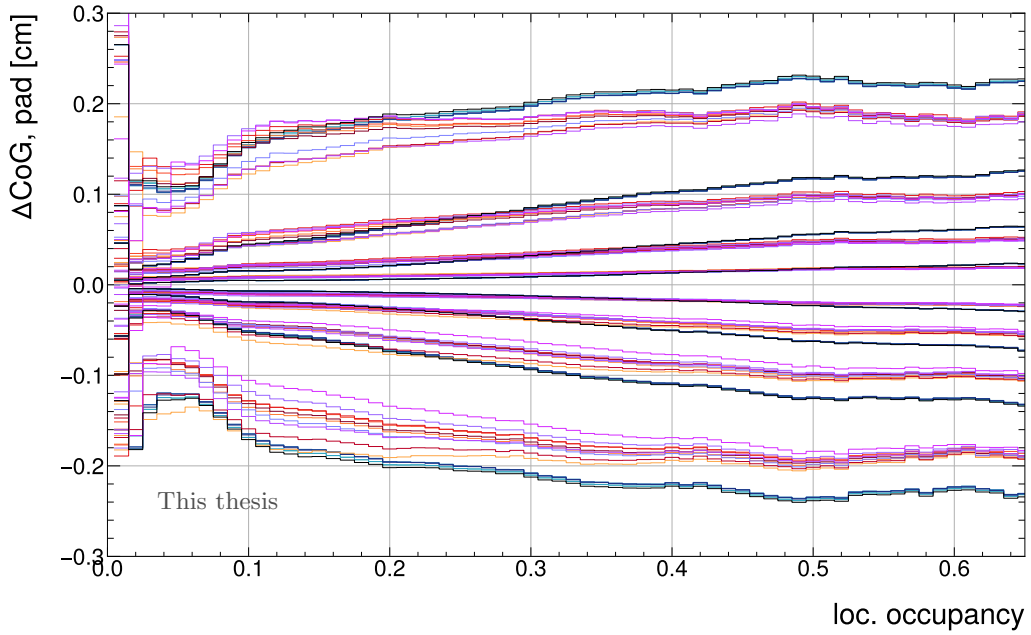


Fig. B.2.7: Center-of-gravity resolution in pad direction with mean offsets as a function of local occupancy.

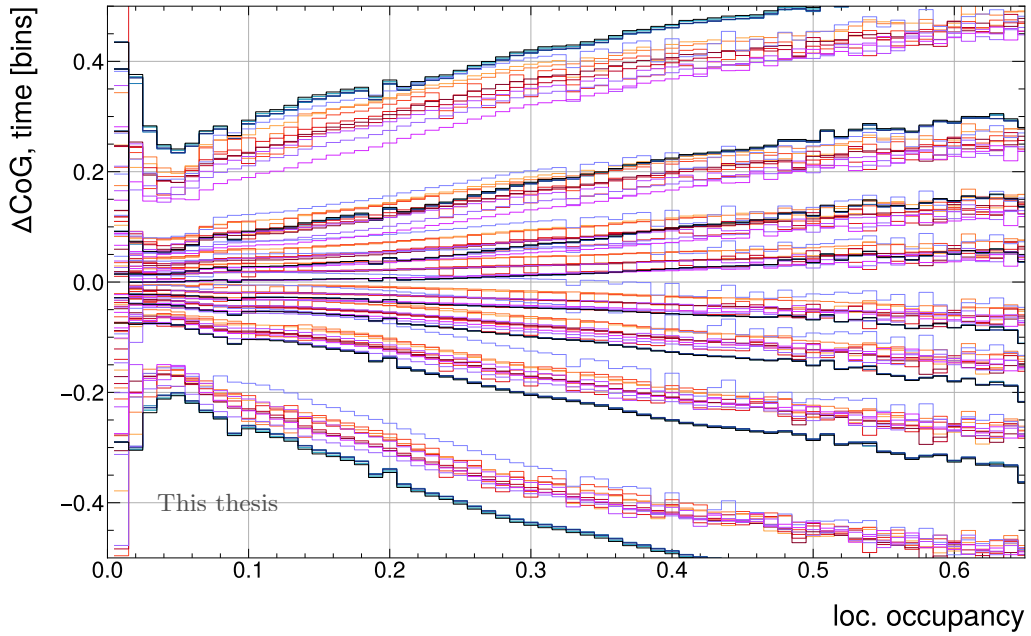


Fig. B.2.8: Center-of-gravity resolution in time direction with mean offsets as a function of local occupancy.

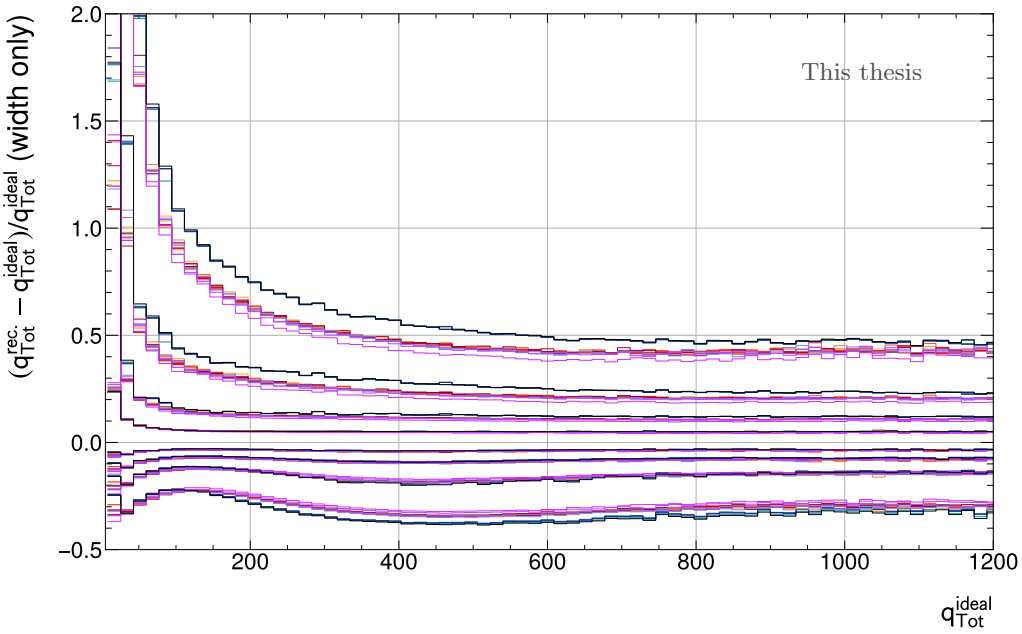


Fig. B.2.9: Percentile distributions of the total charge estimate as a function of the total charge of the cluster (width only). For clusters with low total charge, contamination deteriorates the charge estimation.

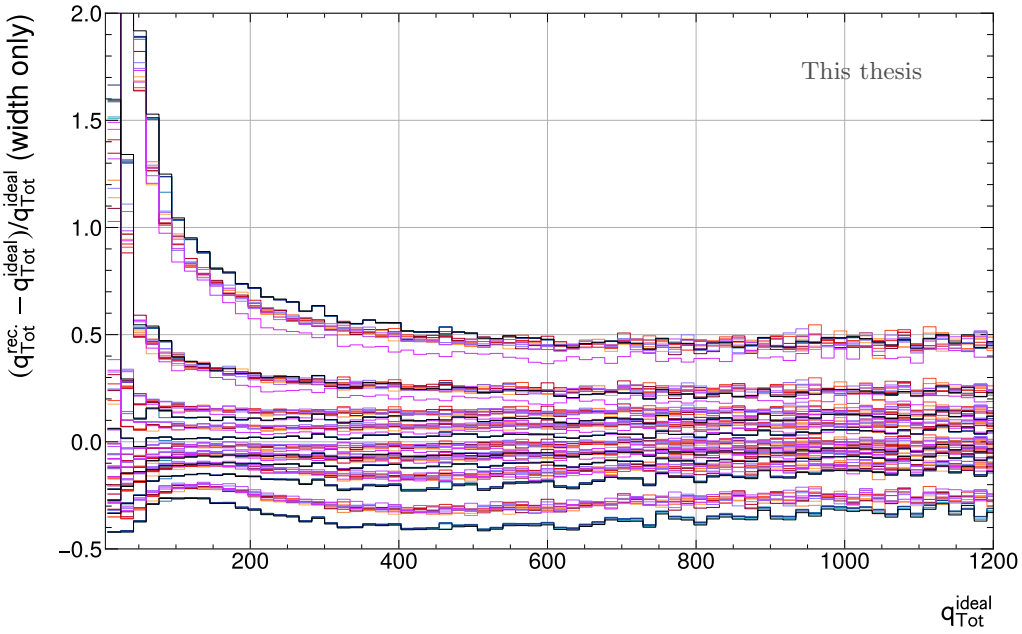


Fig. B.2.10: Percentile distributions of the total charge estimate as a function of the total charge of the cluster with adjusted means. This shows the fluctuation that the mean can have w.r.t. figure (B.2.9).

B.3 Efficiency, clone and fake-rates with space-charge distortion effects

Simulation setting: Pb–Pb, 38 kHz anchored, minimum bias, with space-charge distortion effects.

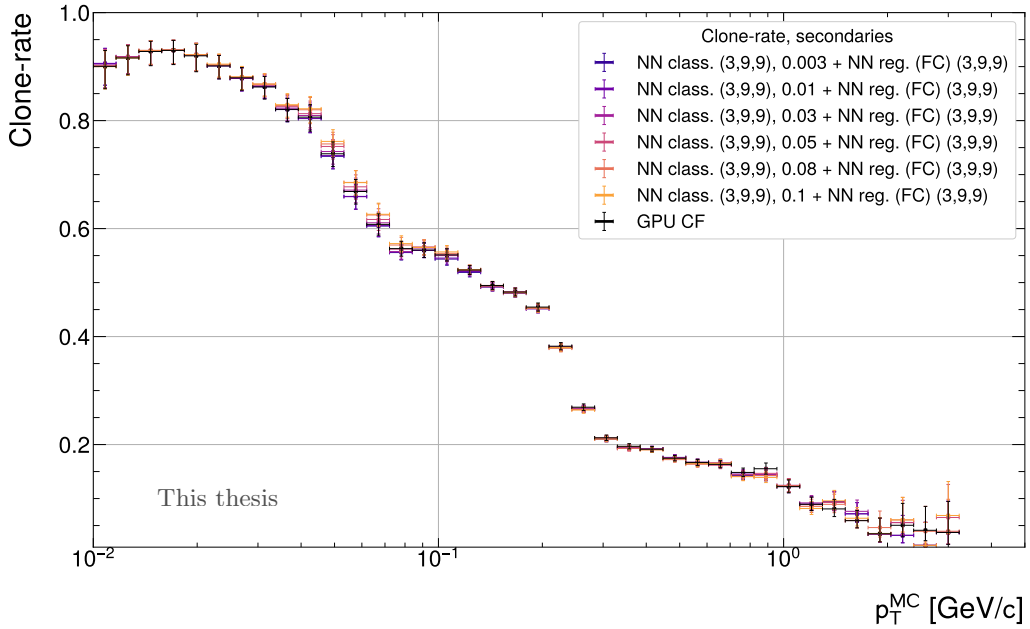


Fig. B.3.11: Clone-rate of tracks as a function of p_T with increasing threshold of the classification network.

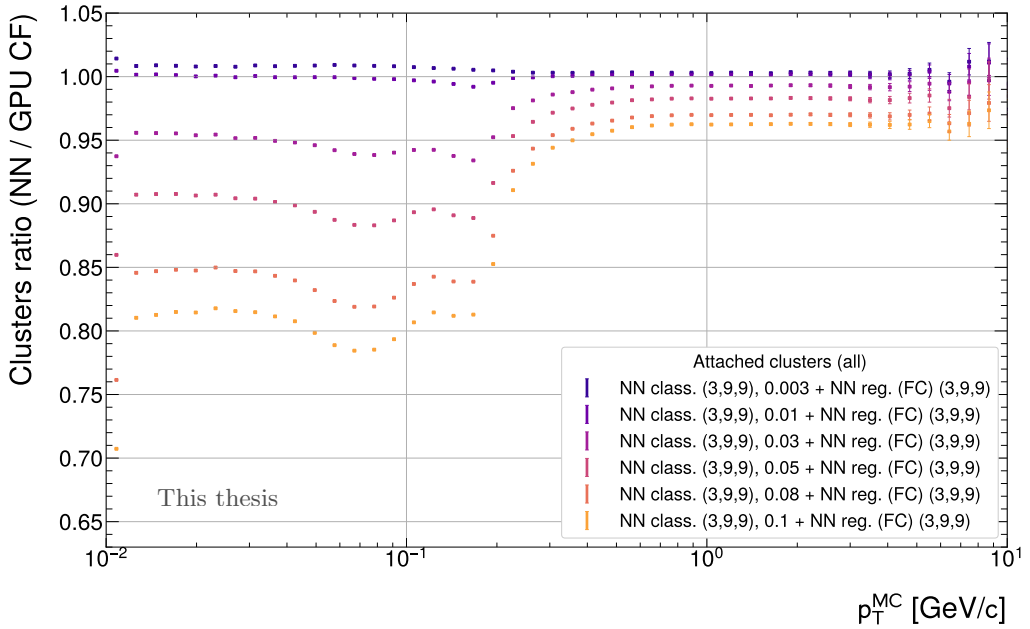


Fig. B.3.12: TPC clusters attached to tracks as a function of p_T from all reconstructed tracks, as a ratio to the number of clusters from the default reconstruction algorithm in each bin.

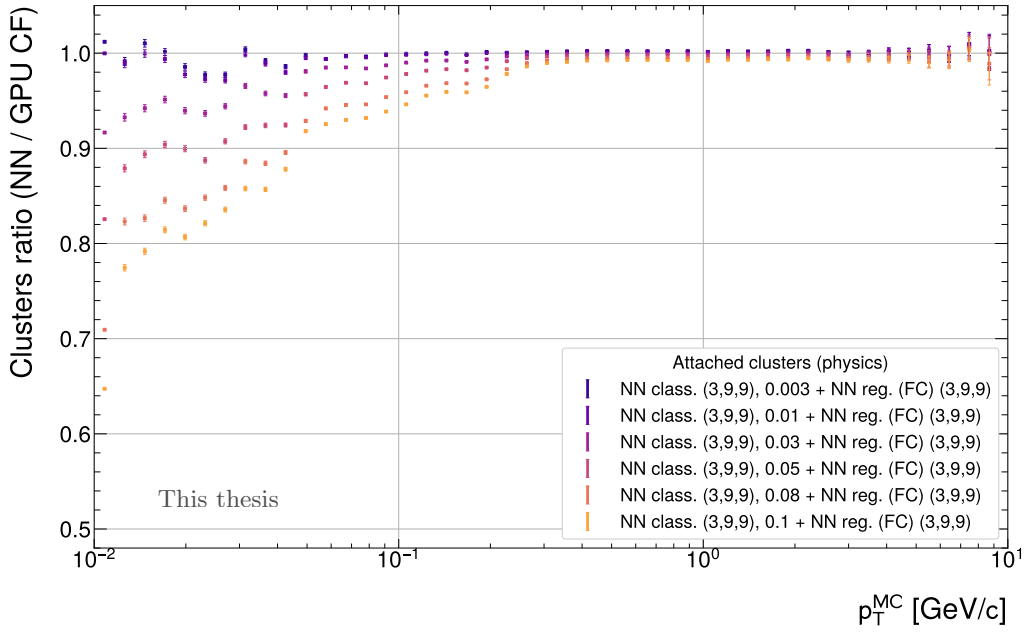


Fig. B.3.13: TPC clusters used in physics tracks as a function of p_T from all reconstructed physics tracks ($\geq$ 60 TPC clusters), as a ratio to the number of clusters from the default reconstruction algorithm in each bin.

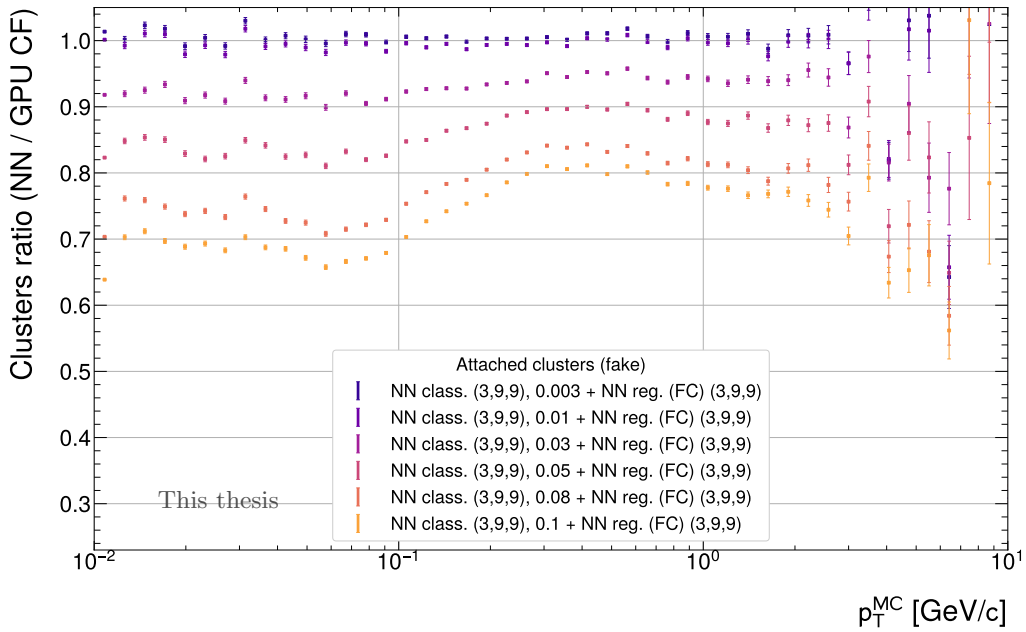


Fig. B.3.14: Fake TPC clusters as a function of p_T, as a ratio to the number of clusters from the default reconstruction algorithm in each bin.

Simulation setting: Pb–Pb, 38 kHz anchored, centrality enforced (0-5%), without space-charge distortion effects. Illustrated below are percentile distributions with mean offsets (in case they are present) for all investigated networks.

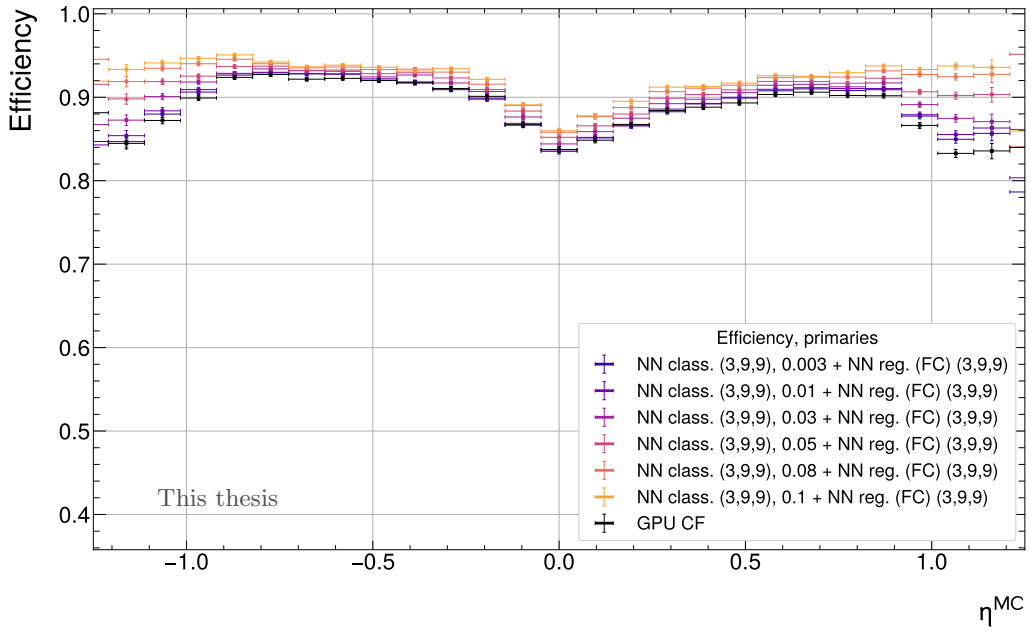


Fig. B.3.15: Tracking efficiency as a function of pseudo-rapidity (η) for primary particles (by MC label).

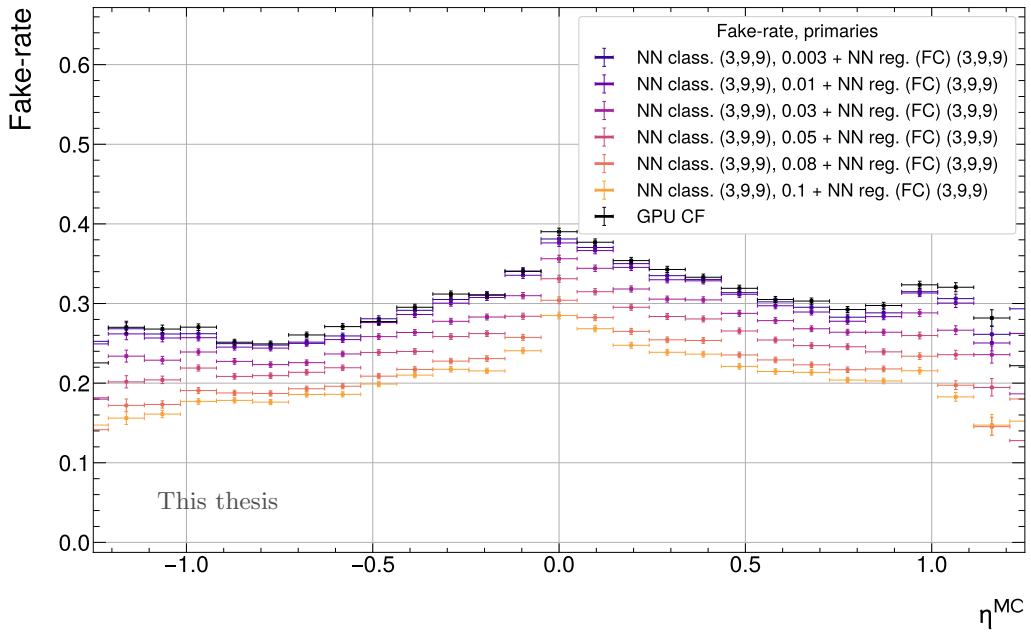


Fig. B.3.16: Track fake-rate as a function of pseudo-rapidity (η) for primary particles (by MC label).

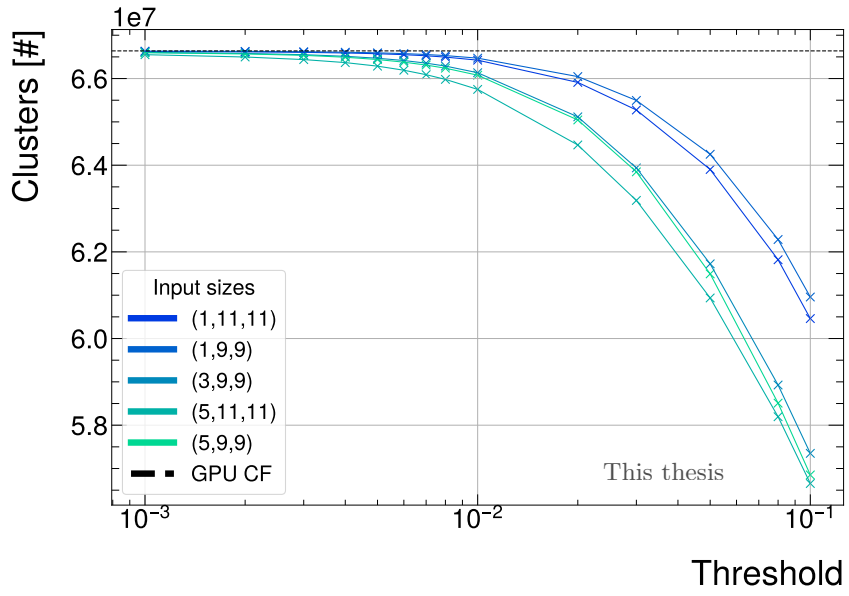


Fig. B.3.17: Relation of clusters as a function of chosen threshold for classification networks with different input sizes.

B.4 Studies with different peak finder algorithm and 3D convolutional networks

A different peak finding scheme was evaluated in which each digit is considered a peak for which all nearest neighbours (NN) suffice to $q \geq q_{\mathrm{NN}}$. This increases the number of peak candidates by $> 20\%$ in 50 kHz Pb–Pb simulations. A correspondingly high efficiency is generated with a significantly reduced fake-rate when using high thresholds.

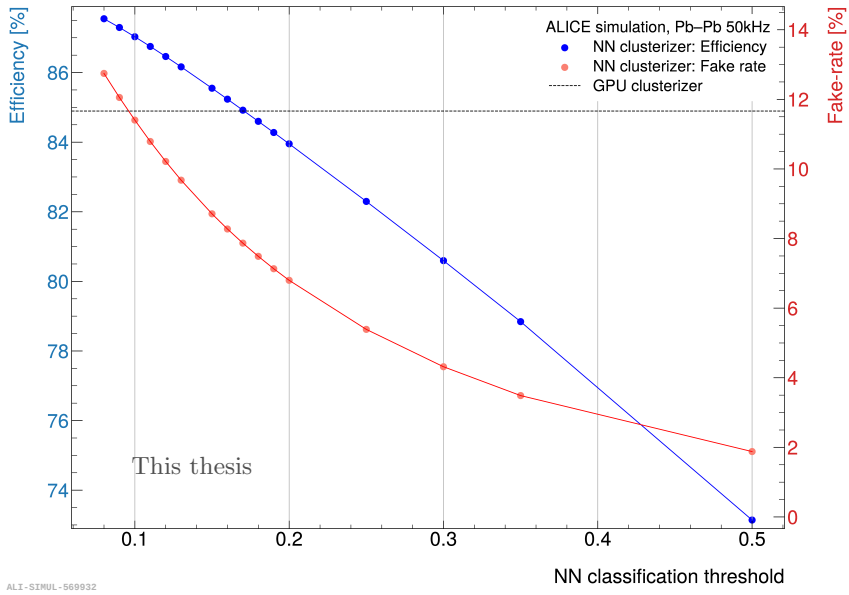


Fig. B.4.18: Efficiency and fake-rate for MC matched clusters using a different peak-finding scheme and 3D convolutional neural networks.

C Real data analysis

C.1 Real data analysis: LHC24ar

The following selections were performed on to select clean V0 samples for the runing of the PID calibration on the LHC24ar dataset.

- Nσ, TPC (all particles) < 5, no Tsallis downsampling
- Post selection: Electron cut Nσ, TPC < 5

O2Physics, v0-selector task, clean sample selection	selection value / threshold
cutAlphaGLow	0.4
dcav0dau	0.3
cutQTG2	0.006
cutAlphaALow	-0.7
cutNsigmaPrTPC	5
cutAlphaAHigh	-0.35
cutAlphaG	0.4
maxpsipair	0.8
cutAPK0SHigh	0.8
cutAPK0SHighTop	1.0
cutQTL	0.03
v0cospa	0.9997
mincrossedrows	70
cutAlphaGHigh	0.8
cutQTG	0.006
maxchi2tpc	4
dcamin	0
v0Rmin	0
cutAPK0SLow	0.199
v0max_mee	0.015
cutAPL2	-0.69
cutAPL1	0.107
dcamax	10^{10}
cutAPL3	0.5
v0Rmax	90
cutAlphaLLow	0.35
cutNsigmaElTPC	5
cutQTK0SHigh	0.215
cutAlphaLHigh	0.7
cutNsigmaPiTPC	5
cutQTK0SLow	0.1075

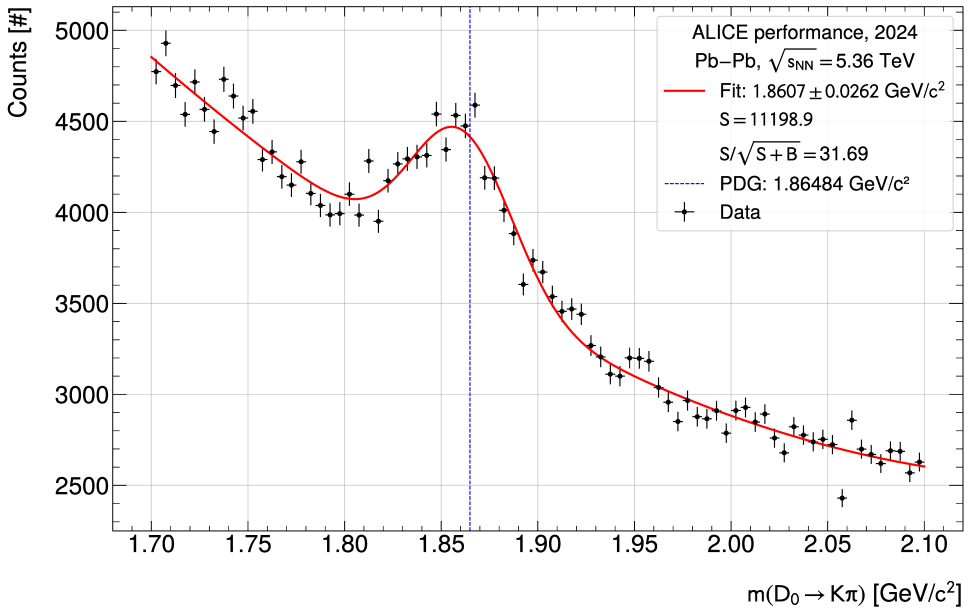
Table C.1.2: Parameter configuration for the O2Physics v0-selector task for the clean sample selection.

O2Physics, v0-selector task, K^0_S and Λ selection	selection value / threshold
cutAlphaGLow	0.4
dcav0dau	0.3
cutQTG2	0.006
cutAlphaALow	-0.7
cutNsigmaPrTPC	5
cutAlphaAHigh	-0.35
cutAlphaG	0.4
maxpsipair	0.65
cutAPK0SHigh	0.8
cutQTL	0.03
produceV0ID	1
v0cospa	0.9997
mincrossedrows	70
cutAlphaGHigh	0.8
cutQTG	0.006
maxchi2tpc	4
dcamin	0
v0Rmin	0
fillhisto	1
cutAPK0SLow	0.185
v0max_mee	0.012
cutAPL2	-0.69
cutAPL1	0.107
dcamax	1e+10
cutAPL3	0.5
v0Rmax	90
cutAlphaLLow	0.35
cutNsigmaElTPC	5
cutQTK0SHigh	0.222
cutAlphaLHigh	0.7
cutNsigmaPiTPC	5
cutAPK0SHighTop	1
cutQTK0SLow	0.1075

Table C.1.3: Parameter configuration for the O2Physics v0-selector task for the K^0_S and Λ invariant mass distributions.

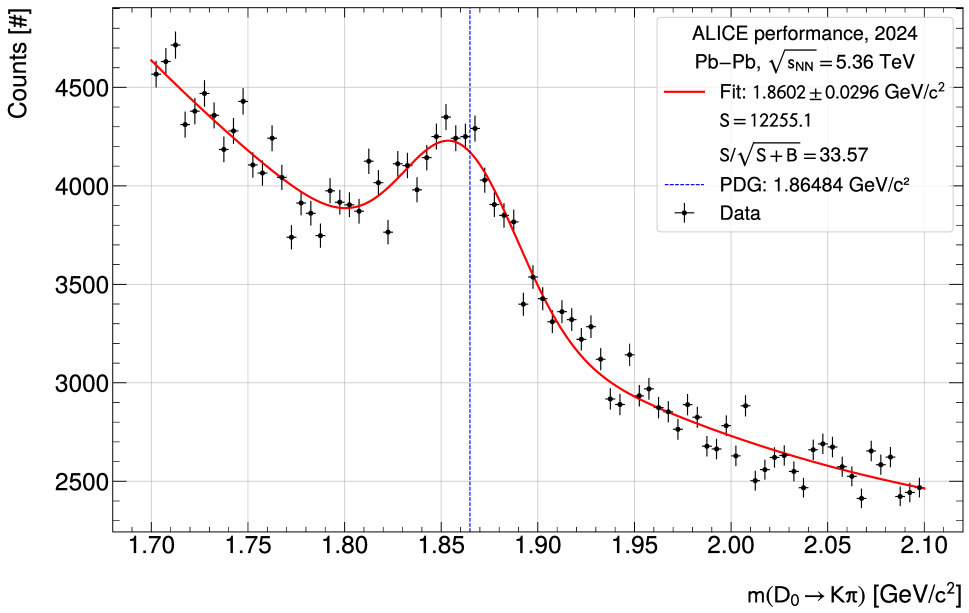
C.2 D^0 analysis

D^0 invariant mass peaks after Optuna optimization on the dataset reconstructed with the regular cluster finder. The resulting BDT and cos(PA) scores are applied identically on both datasets. 1000 steps were used in the Optuna optimization procedure. Weights of $w_1 = 0.3$ and $w_2 = 0.7$ were used.



ALI-PERF-619278

Fig. C.2.19: Optuna-optimized D^0 invariant mass peak with a third order polynomial background for the neural network cluster finder. The threshold setting was set to 0.1 ($\approx$ 18% cluster reduction).



ALI-PERF-619273

Fig. C.2.20: Optuna-optimized D^0 invariant mass peak with a third order polynomial background for the heurisitc cluster finder.

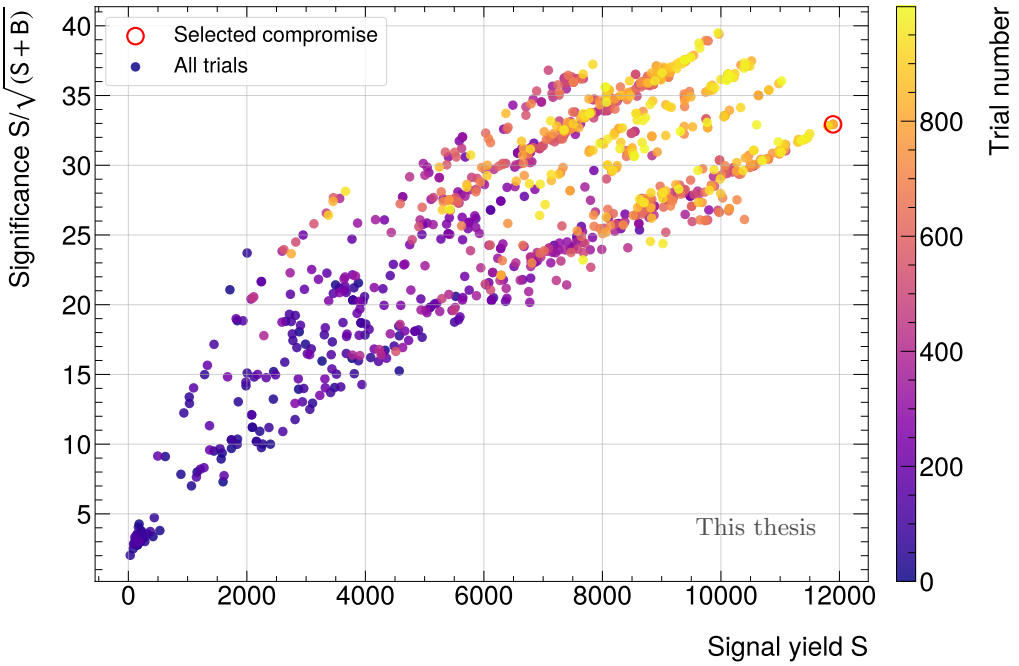


Fig. C.2.21: Score with colorcoded trial number of the Optuna optimzation.

C.3 Online runs

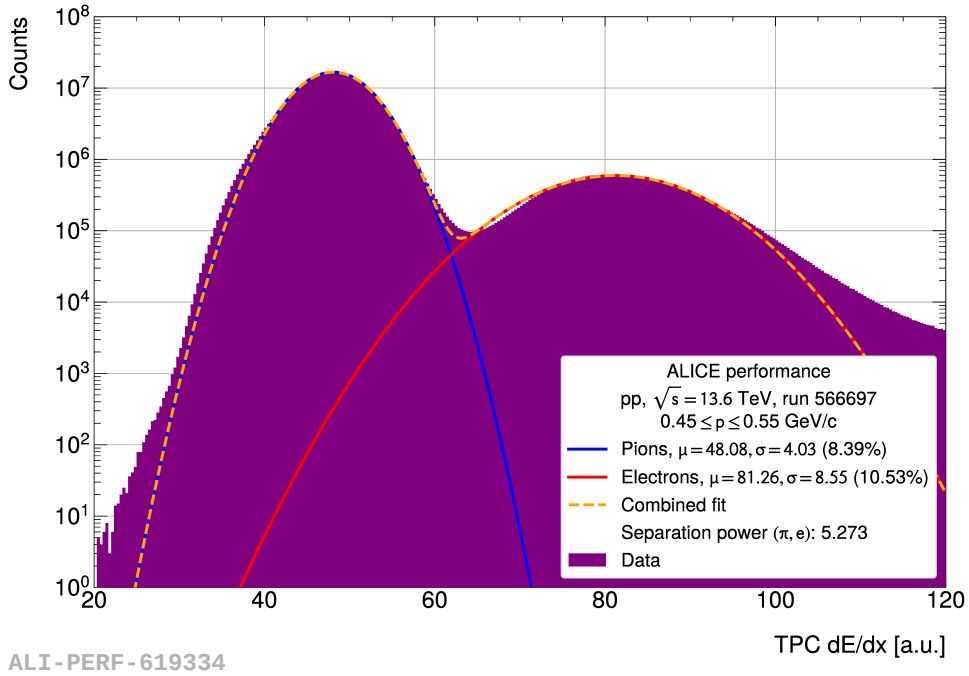


Fig. C.3.22: Electron-pion separation for Run 566697 (neural network classification, 0.05 + heuristic regression algorithm).

D Computing

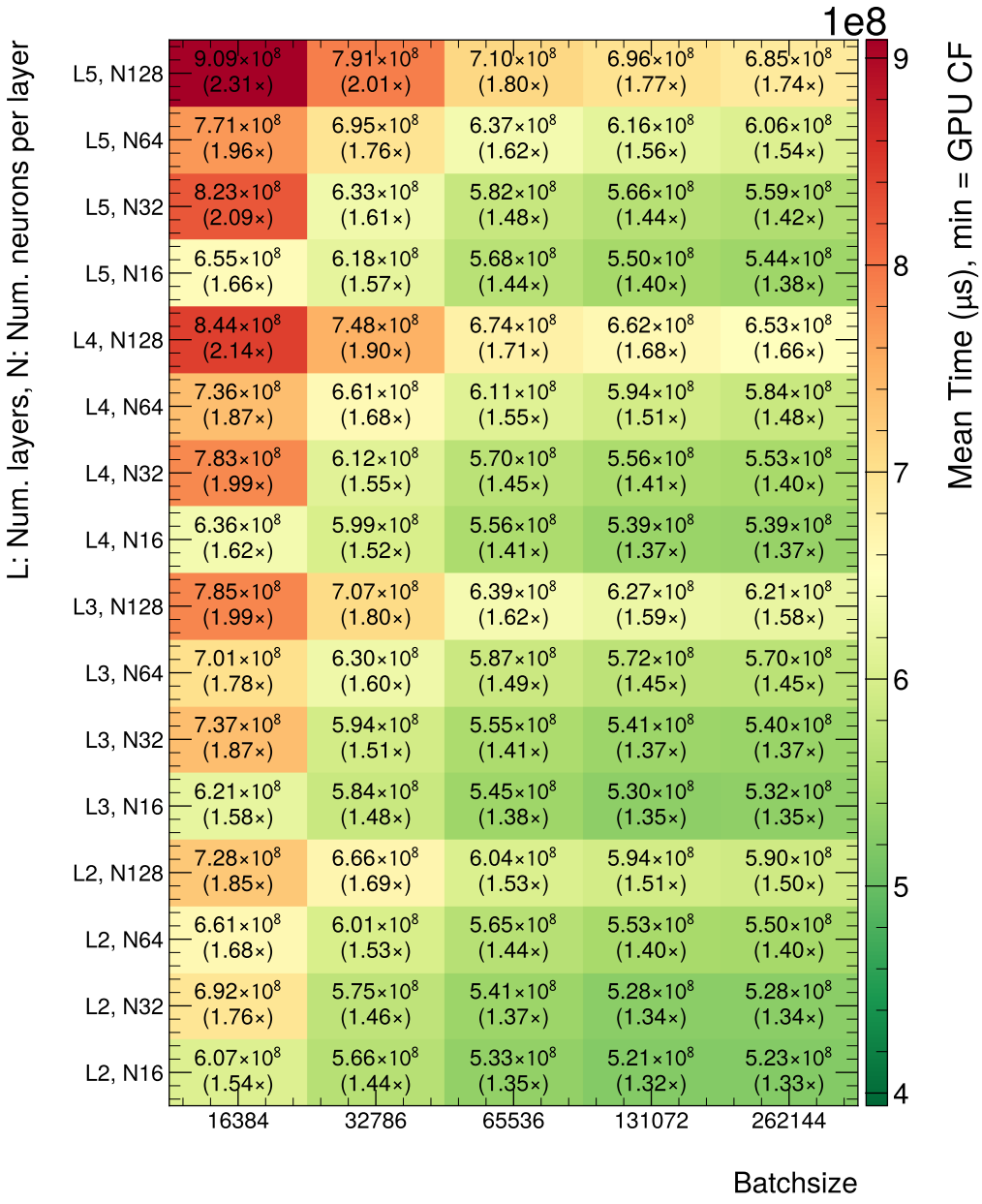


Fig. D.0.23: Total wall time of the reconstruction process for 16 timeframes on one event processing node with 8 MI50 GPUs. 2 timeframes are processed sequentially per GPU. The ratio to the default reconstruction algorithm (GPU CF) is stated as a factor in each cell.

	GPU CF [s]	FC, 16384 [s]	FC, 65536 [s]	FC, 262144 [s]
GPUTPCNNClusterizer_ONNXClassification_0_0	0.0000	1.8266	0.8508	0.6191
GPUTPCNNClusterizer_ONNXRegression_1_1	0.0000	2.0742	0.8127	0.6106
GPUTPCNNClusterizer_ONNXRegression2_2_2	0.0000	0.0000	0.0000	0.0000
GPUTPCNNClusterizer_ONNXClassification_0_3	0.0000	1.8881	0.8031	0.5499
GPUTPCNNClusterizer_ONNXRegression_1_4	0.0000	2.0857	0.8101	0.6192
GPUTPCNNClusterizer_ONNXRegression2_2_5	0.0000	0.0000	0.0000	0.0000
GPUTPCNNClusterizer_ONNXClassification_0_6	0.0000	1.6198	0.8057	0.5532
GPUTPCNNClusterizer_ONNXRegression_1_7	0.0000	2.0755	0.8170	0.6149
GPUTPCNNClusterizer_ONNXRegression2_2_8	0.0000	0.0000	0.0000	0.0000
GPUTPCNNClusterizer_ONNXClassification_0_9	0.0000	0.0000	0.0000	0.0000
GPUTPCNNClusterizer_ONNXRegression_1_10	0.0000	0.0000	0.0000	0.0000
GPUTPCNNClusterizer_ONNXRegression2_2_11	0.0000	0.0000	0.0000	0.0000
GPUMemClean16	0.2170	0.2212	0.2218	0.2214
GPUTPCCFDecodeZSDenseLink	1.1980	1.2028	1.1991	1.1988
GPUTPCCFCheckPadBaseline	0.0320	0.0320	0.0321	0.0319
GPUTPCCFPeakFinder	0.2865	0.2870	0.2868	0.2871
GPUTPCCFStreamCompaction_scanStart	0.0482	0.0483	0.0484	0.0484
GPUTPCCFStreamCompaction_scanUp	0.0100	0.0104	0.0104	0.0105
GPUTPCCFStreamCompaction_scanTop	0.0102	0.0105	0.0107	0.0106
GPUTPCCFStreamCompaction_scanDown	0.0093	0.0095	0.0095	0.0095
GPUTPCCFStreamCompaction_compactDigits	0.0758	0.0767	0.0766	0.0767
GPUTPCCFNoiseSuppression_noiseSuppression	0.3295	0.3297	0.3297	0.3301
GPUTPCCFNoiseSuppression_updatePeaks	0.0612	0.0615	0.0611	0.0614
GPUTPCCFDeconvolution	0.4597	0.4608	0.4606	0.4590
GPUTPCNNClusterizerKernels_fillInputNNGPU	0.0000	1.4896	1.1540	1.0767
GPUTPCNNClusterizerKernels_publishDeconvolutionFla	0.0000	0.5349	0.4071	0.4181
GPUTPCNNClusterizerKernels_determineClass1Labels	0.0000	0.2438	0.0671	0.0238
GPUTPCNNClusterizerKernels_publishClass1Regression	0.0000	0.5627	0.2935	0.2547
GPUTPCCFGather	0.0175	0.0149	0.0149	0.0148
GPUTPCCreateOccupancyMap_fill	0.1295	0.1375	0.1314	0.1289
GPUTPCCreateOccupancyMap_fold	0.0002	0.0002	0.0002	0.0002
GPUTPCCreateTrackingData	0.5282	0.4830	0.4837	0.4853
GPUTPCNeighboursFinder	0.3332	0.2379	0.2378	0.2380
GPUTPCNeighboursCleaner	0.0198	0.0180	0.0180	0.0180
GPUTPCStartHitsFinder	0.0066	0.0054	0.0054	0.0054
GPUTPCStartHitsSorter	0.0172	0.0170	0.0171	0.0170
GPUTPCTrackletConstructor	5.2191	5.1478	5.1461	5.1749
GPUTPCTrackletSelector	0.2962	0.2871	0.2870	0.2875
GPUTPCExtrapolationTrackingCopyNumbers	0.0003	0.0003	0.0003	0.0003
GPUTPCExtrapolationTracking	0.0003	0.0003	0.0003	0.0003
GPUTPCGMMergerUnpackSaveNumber	0.0006	0.0006	0.0006	0.0006
GPUTPCGMMergerUnpackResetIds	0.0004	0.0003	0.0004	0.0004
GPUTPCGMMergerSectorRefit	1.3772	1.3274	1.3225	1.3301
GPUTPCGMMergerUnpackGlobal	0.0016	0.0015	0.0015	0.0015
GPUTPCGMMergerClearLinks	0.0001	0.0001	0.0001	0.0001
GPUTPCGMMergerMergeWithinPrepare	0.0036	0.0035	0.0035	0.0035
GPUTPCGMMergerMergeBorders_step0	0.0038	0.0037	0.0037	0.0037
GPUTPCGMMergerMergeBorders_variant	0.0333	0.0321	0.0320	0.0321
GPUTPCGMMergerMergeBorders_step2	1.7599	1.6962	1.6960	1.6959
GPUTPCGMMergerResolve_step0	0.0001	0.0001	0.0001	0.0001
GPUTPCGMMergerResolve_step1	0.0015	0.0015	0.0015	0.0015
GPUTPCGMMergerResolve_step2	0.0049	0.0046	0.0046	0.0046
GPUTPCGMMergerResolve_step3	0.0010	0.0009	0.0009	0.0009
GPUTPCGMMergerResolve_step4	0.1827	0.1767	0.1755	0.1768
GPUTPCGMMergerMergeSectorsPrepare	0.0198	0.0192	0.0192	0.0193
GPUTPCGMMergerLinkExtrapolatedTracks	0.0002	0.0002	0.0002	0.0002
GPUTPCGMMergerCollect	0.3222	0.2808	0.2740	0.2890
GPUTPCGMMergerMergeCE	0.0025	0.0026	0.0024	0.0026
GPUTPCGMMergerSortTracksPrepare	0.0001	0.0001	0.0001	0.0001
GPUTPCGMMergerSortTracks	0.0001	0.0001	0.0001	0.0001
GPUTPCGMMergerPrepareForFit_step0	0.0001	0.0001	0.0001	0.0001
GPUTPCGMMergerSortTracksQPt	0.0222	0.0208	0.0208	0.0210
GPUTPCGMMergerPrepareForFit_step1	0.1176	0.1060	0.1054	0.1060
GPUTPCGMMergerPrepareForFit_step2	0.0488	0.0457	0.0458	0.0458
GPUTPCGMMergerTrackFit	6.9221	5.5868	5.8183	5.7310
GPUTPCGMMergerFollowLoopers	0.1421	0.1406	0.1384	0.1371
GPUTPCGMMergerFinalize_step0	0.0350	0.0338	0.0338	0.0338
GPUTPCGMMergerFinalize_step1	0.0678	0.0657	0.0654	0.0656
GPUTPCGMMergerFinalize_step2	0.0147	0.0127	0.0126	0.0126
GPUTPCGMMergerMergeLoopers_step0	0.0009	0.0008	0.0008	0.0008
GPUTPCGMMergerMergeLoopers_step1	0.0056	0.0052	0.0053	0.0053
GPUTPCGMMergerMergeLoopers_step2	0.1111	0.0944	0.0948	0.0953
GPUTPCGMO2Output_prepare	0.0406	0.0389	0.0387	0.0388
GPUTPCGMO2Output_sort	0.0010	0.0010	0.0010	0.0010
GPUTPCGMO2Output_output	0.2529	0.2494	0.2505	0.2513
GPUTPCCompressionKernels_step0attached	0.8127	0.7794	0.7690	0.7766
GPUTPCCompressionKernels_step1unattached	2.4915	2.1255	2.1491	2.1876
GPUTPCCFClusterizer	0.1844	0.0000	0.0000	0.0000
GPUTPCCompressionGatherKernels_multiBlock	0.0552	0.0531	0.0530	0.0531
Tasks - TPC Sector Tracking	6.9044	6.6740	6.6648	6.6950
DMA to GPU - TPC Sector Tracking	0.0019	0.0021	0.0021	0.0022
DMA to Host - TPC Sector Tracking	0.0023	0.0024	0.0025	0.0027
Tasks - TPC Track Merging and Fit	11.5256	9.9804	10.1971	10.1350
GPU - TPC Track Merging and Fit	0.0034	0.0035	0.0035	0.0035
Host - TPC Track Merging and Fit	0.0702	0.0677	0.0677	0.0679
Tasks - TPC Compression	3.3593	2.9580	2.9710	3.0173
DMA to GPU - TPC Compression	0.0009	0.0009	0.0009	0.0009
DMA to Host - TPC Compression	0.1401	0.1222	0.1222	0.1224
Tasks - TPC Cluster Finding	2.9392	5.5960	4.6833	4.5337
DMA to GPU - TPC Cluster Finding	2.4199	2.6160	2.4203	2.8121
DMA to Host - TPC Cluster Finding	0.1999	0.1910	0.1869	0.1929
Total Kernel Time	24.7286	25.2085	24.5163	24.3811
Total Wall Time	52.3181	127.9063	68.6162	55.2541

Table D.0.4: Individual kernel times as a function of the batchsize chosen for the input to the neural network. A fully connected network with 4 layers, 32 neurons per layer and an input window of (3,9,9) was used. The threshold was set to 0.05, rejecting approximately 10% of raw clusters. The evaluation time is normalized to 1 time-frame of 50 kHz synthetic PbPb data (full system test), the measurement error is $< 2\%$.

	GPU CF [s]	FC, 0.005 [s]	FC, 0.01 [s]	FC, 0.02 [s]	FC, 0.05 [s]	FC, 0.1 [s]
GPUTPCNNClusterizer_ONNXClassification_0_0	0.0000	0.5975	0.5981	0.6077	0.6191	0.6100
GPUTPCNNClusterizer_ONNXRegression_1_1	0.0000	0.6035	0.6066	0.6068	0.6106	0.6089
GPUTPCNNClusterizer_ONNXRegression2_2_2	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
GPUTPCNNClusterizer_ONNXClassification_0_3	0.0000	0.5221	0.5274	0.5321	0.5499	0.5352
GPUTPCNNClusterizer_ONNXRegression_1_4	0.0000	0.6111	0.6109	0.6100	0.6192	0.6147
GPUTPCNNClusterizer_ONNXRegression2_2_5	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
GPUTPCNNClusterizer_ONNXClassification_0_6	0.0000	0.5159	0.5279	0.5336	0.5532	0.5253
GPUTPCNNClusterizer_ONNXRegression_1_7	0.0000	0.6095	0.6073	0.6088	0.6149	0.6116
GPUTPCNNClusterizer_ONNXRegression2_2_8	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
GPUTPCNNClusterizer_ONNXClassification_0_9	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
GPUTPCNNClusterizer_ONNXRegression_1_10	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
GPUTPCNNClusterizer_ONNXRegression2_2_11	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
GPUMemClean16	0.2170	0.2202	0.2203	0.2200	0.2214	0.2201
GPUTPCCFDecodeZSDenseLink	1.1980	1.1979	1.1997	1.1987	1.1988	1.1982
GPUTPCCFCheckPadBaseline	0.0320	0.0323	0.0322	0.0322	0.0319	0.0321
GPUTPCCFPeakFinder	0.2865	0.2873	0.2875	0.2871	0.2871	0.2868
GPUTPCCFStreamCompaction_scanStart	0.0482	0.0483	0.0484	0.0484	0.0484	0.0484
GPUTPCCFStreamCompaction_scanUp	0.0100	0.0103	0.0104	0.0103	0.0105	0.0104
GPUTPCCFStreamCompaction_scanTop	0.0102	0.0104	0.0106	0.0104	0.0106	0.0106
GPUTPCCFStreamCompaction_scanDown	0.0093	0.0094	0.0095	0.0094	0.0095	0.0095
GPUTPCCFStreamCompaction_compactDigits	0.0758	0.0765	0.0764	0.0766	0.0767	0.0765
GPUTPCCFNoiseSuppression_noiseSuppression	0.3295	0.3299	0.3300	0.3298	0.3301	0.3299
GPUTPCCFNoiseSuppression_updatePeaks	0.0612	0.0612	0.0614	0.0607	0.0614	0.0614
GPUTPCCFDeconvolution	0.4597	0.4588	0.4593	0.4594	0.4590	0.4595
GPUTPCNNClusterizerKernels_fillInputNNGPU	0.0000	1.0759	1.0764	1.0767	1.0767	1.0767
GPUTPCNNClusterizerKernels_publishDeconvolutionFla	0.0000	0.4176	0.4182	0.4180	0.4181	0.4182
GPUTPCNNClusterizerKernels_determineClass1Labels	0.0000	0.0237	0.0237	0.0236	0.0238	0.0240
GPUTPCNNClusterizerKernels_publishClass1Regression	0.0000	0.2663	0.2647	0.2607	0.2547	0.2494
GPUTPCCFGather	0.0175	0.0173	0.0168	0.0161	0.0148	0.0136
GPUTPCCreateOccupancyMap_fill	0.1295	0.1420	0.1417	0.1391	0.1289	0.1224
GPUTPCCreateOccupancyMap_fold	0.0002	0.0002	0.0002	0.0002	0.0002	0.0002
GPUTPCCreateTrackingData	0.5282	0.5319	0.5206	0.5086	0.4853	0.4539
GPUTPCNeighboursFinder	0.3332	0.3221	0.3062	0.2812	0.2380	0.1990
GPUTPCNeighboursCleaner	0.0198	0.0199	0.0196	0.0191	0.0180	0.0168
GPUTPCStartHitsFinder	0.0066	0.0064	0.0062	0.0059	0.0054	0.0050
GPUTPCStartHitsSorter	0.0172	0.0173	0.0173	0.0172	0.0170	0.0166
GPUTPCTrackletConstructor	5.2191	5.2661	5.2516	5.2575	5.1749	4.9995
GPUTPCTrackletSelector	0.2962	0.2991	0.2965	0.2932	0.2875	0.2807
GPUTPCExtrapolationTrackingCopyNumbers	0.0003	0.0003	0.0003	0.0003	0.0003	0.0003
GPUTPCExtrapolationTracking	0.0003	0.0003	0.0003	0.0003	0.0003	0.0003
GPUTPCGMMergerUnpackSaveNumber	0.0006	0.0006	0.0006	0.0006	0.0006	0.0006
GPUTPCGMMergerUnpackResetIds	0.0004	0.0004	0.0004	0.0004	0.0004	0.0004
GPUTPCGMMergerSectorRefit	1.3772	1.3778	1.3667	1.3503	1.3301	1.2975
GPUTPCGMMergerUnpackGlobal	0.0016	0.0016	0.0016	0.0016	0.0015	0.0014
GPUTPCGMMergerClearLinks	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
GPUTPCGMMergerMergeWithinPrepare	0.0036	0.0036	0.0036	0.0036	0.0035	0.0035
GPUTPCGMMergerMergeBorders_step0	0.0038	0.0038	0.0038	0.0037	0.0037	0.0037
GPUTPCGMMergerMergeBorders_variant	0.0333	0.0333	0.0331	0.0329	0.0321	0.0315
GPUTPCGMMergerMergeBorders_step2	1.7599	1.7593	1.7511	1.7365	1.6959	1.6457
GPUTPCGMMergerResolve_step0	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
GPUTPCGMMergerResolve_step1	0.0015	0.0015	0.0015	0.0015	0.0015	0.0014
GPUTPCGMMergerResolve_step2	0.0049	0.0049	0.0048	0.0047	0.0046	0.0045
GPUTPCGMMergerResolve_step3	0.0010	0.0010	0.0009	0.0009	0.0009	0.0009
GPUTPCGMMergerResolve_step4	0.1827	0.1837	0.1818	0.1812	0.1768	0.1685
GPUTPCGMMergerMergeSectorsPrepare	0.0198	0.0199	0.0199	0.0196	0.0193	0.0186
GPUTPCGMMergerLinkExtrapolatedTracks	0.0002	0.0002	0.0002	0.0002	0.0002	0.0001
GPUTPCGMMergerCollect	0.3222	0.3280	0.3533	0.3107	0.2890	0.2550
GPUTPCGMMergerMergeCE	0.0025	0.0023	0.0024	0.0025	0.0026	0.0025
GPUTPCGMMergerSortTracksPrepare	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
GPUTPCGMMergerSortTracks	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
GPUTPCGMMergerPrepareForFit_step0	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
GPUTPCGMMergerSortTracksQPt	0.0222	0.0220	0.0218	0.0217	0.0210	0.0204
GPUTPCGMMergerPrepareForFit_step1	0.1176	0.1168	0.1155	0.1111	0.1060	0.1030
GPUTPCGMMergerPrepareForFit_step2	0.0488	0.0487	0.0482	0.0475	0.0458	0.0438
GPUTPCGMMergerTrackFit	6.9221	6.8350	7.1927	6.3052	5.7310	5.1766
GPUTPCGMMergerFollowLoopers	0.1421	0.1580	0.1392	0.1368	0.1371	0.1320
GPUTPCGMMergerFinalize_step0	0.0350	0.0350	0.0349	0.0346	0.0338	0.0328
GPUTPCGMMergerFinalize_step1	0.0678	0.0676	0.0675	0.0671	0.0656	0.0637
GPUTPCGMMergerFinalize_step2	0.0147	0.0144	0.0140	0.0137	0.0126	0.0117
GPUTPCGMMergerMergeLoopers_step0	0.0009	0.0008	0.0009	0.0008	0.0008	0.0008
GPUTPCGMMergerMergeLoopers_step1	0.0056	0.0056	0.0056	0.0055	0.0053	0.0050
GPUTPCGMMergerMergeLoopers_step2	0.1111	0.1051	0.1038	0.1009	0.0953	0.0840
GPUTPCGMO2Output_prepare	0.0406	0.0407	0.0406	0.0404	0.0388	0.0375
GPUTPCGMO2Output_sort	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010
GPUTPCGMO2Output_output	0.2529	0.2576	0.2528	0.2534	0.2513	0.2430
GPUTPCCompressionKernels_step0attached	0.8127	0.8133	0.8074	0.7987	0.7766	0.7543
GPUTPCCompressionKernels_step1unattached	2.4915	2.5316	2.4791	2.3962	2.1876	1.9882
GPUTPCCFClusterizer	0.1844	0.0000	0.0000	0.0000	0.0000	0.0000
GPUTPCCompressionGatherKernels_multiBlock	0.0552	0.0551	0.0547	0.0543	0.0531	0.0518
Tasks - TPC Sector Tracking	6.9044	6.9632	6.9154	6.8726	6.6950	6.4179
DMA to GPU - TPC Sector Tracking	0.0019	0.0021	0.0021	0.0020	0.0022	0.0021
DMA to Host - TPC Sector Tracking	0.0023	0.0026	0.0025	0.0022	0.0027	0.0024
Tasks - TPC Track Merging and Fit	11.5256	11.4582	11.7923	10.8183	10.1350	9.4170
GPU - TPC Track Merging and Fit	0.0034	0.0035	0.0035	0.0035	0.0035	0.0034
Host - TPC Track Merging and Fit	0.0702	0.0702	0.0699	0.0694	0.0679	0.0657
Tasks - TPC Compression	3.3593	3.3999	3.3411	3.2492	3.0173	2.7943
DMA to GPU - TPC Compression	0.0009	0.0009	0.0009	0.0009	0.0009	0.0009
DMA to Host - TPC Compression	0.1401	0.1390	0.1361	0.1315	0.1224	0.1139
Tasks - TPC Cluster Finding	2.9392	4.5431	4.5454	4.5382	4.5337	4.5255
DMA to GPU - TPC Cluster Finding	2.4199	2.2264	2.2145	2.3510	2.8121	2.3924
DMA to Host - TPC Cluster Finding	0.1999	0.1956	0.1935	0.1912	0.1929	0.1741
Total Kernel Time	24.7286	26.3644	26.5941	25.4783	24.3811	23.1547
Total Wall Time	52.3181	56.3228	56.4593	55.5616	55.2541	53.3955

Table D.0.5: Individual kernel times as a function of the cutoff threshold chosen for the input to the neural network. A fully connected network with 4 layers, 32 neurons per layer and an input window of (3,9,9) was used. The batchsize was set to 262144, the value chosen for deployment. The evaluation time is normalized to 1 timeframe of 50 kHz synthetic PbPb data (full system test), the measurement error is $< 2\%$.

E Reproducibility

Below are the versions of the different software packages, necessary to reproduce the results presented in this thesis.

Monte Carlo studies in chapter 4

Repository	Link, branch, commit hash
alidist	https://github.com/ChSonnabend/alidist.git master 45d4b57c7d3bbda103d6ee7e8b240913c9801885
O2	https://github.com/ChSonnabend/AliceO2.git working_chi2woFlags_newTracking dc145017e3f14a394a1528a9078e004d2da5a3e1
O2DPG	https://github.com/ChSonnabend/O2DPG working a90f34d0397534c3b85f72210625404355c5a654

Real data studies in chapter 5

Repository	Link and commit hash
alidist	https://github.com/ChSonnabend/alidist.git master 45d4b57c7d3bbda103d6ee7e8b240913c9801885
O2	https://github.com/ChSonnabend/AliceO2.git dev 7277814d0a9822e342462787a4d5a088144391a3
O2Physics	https://github.com/ChSonnabend/O2Physics.git devel 5f5d34d6bfb15060feb453e249ea5019a417ebf0

Performance tests in the online farm in chapter 6 (official tags)

- O2PDPSuite/epn-20250901-DDv1.6.9-QCv1.181.0-flp-suite-v1.67.0-1
- O2DPG tag: epn-20250901

Commissioning runs (03.10.2025) (official tags)

- O2PDPSuite/epn-20250925.1-DDv1.6.9-QCv1.183.0-flp-suite-v1.69.0-1
- O2DPG tag: epn-20250925.1

Publications

In addition to the references cited in the bibliography of this thesis, several publications related to the work presented here have been published by the author or are currently in preparation. The publications listed below have a direct connection to the research conducted in this thesis and partially overlap with the cited literature.

2025

Neural network clusterization for the ALICE TPC online computing
C. Sonnabend for the ALICE collaboration
CHEP2025, EPJ Web Conf., DOI: 10.1051/epjconf/202533701017

Machine learning techniques for PID calibration in the ALICE TPC
C. Sonnabend for the ALICE collaboration
LHCP2025, PoS, DOI: 10.22323/1.499.0128

2024

Machine learning in online/offline calibration and reconstruction at the LHC
C. Sonnabend for the ALICE, ATLAS, CMS and LHCb collaborations
LHCP2024, PoS, DOI: 10.22323/1.478.0151

2023

Particle identification in ALICE and LHCb
C. Sonnabend for the ALICE collaboration
LHCP2023, PoS, DOI: 10.22323/1.450.0051

In preparation

Particle identification with neural networks in the ALICE TPC in Run 3

B Bibliography

[1] ATLAS collaboration, "A neural network clustering algorithm for the ATLAS silicon pixel detector," *Journal of Instrumentation*, vol. 9, no. 09, P09009, Sep. 2014. https://doi.org/10.1088/1748-0221/9/09/P09009.

[2] ALICE collaboration, "The ALICE experiment: a journey through QCD," *European Physical Journal C*, vol. 84, no. 8, 2024, ISSN: 1434-6044. https://doi.org/10.1140/epjc/s10052-024-12935-y.

[3] Particle Data Group, "Review of particle physics, 2024-2025," vol. 110, 2024. https://doi.org/10.1103/PhysRevD.110.030001.

[4] R. Hagedorn, "Statistical thermodynamics of strong interactions at high-energies," *Nuovo Cim. Suppl.*, vol. 3, pp. 147–186, 1965.

[5] B. Szabolcs, et al., "Transition temperature and the equation of state from lattice qcd, wuppertal-budapest results," 2011. https://arxiv.org/abs/1109.5032.

[6] T. Bhattacharya, et al., "QCD Phase Transition with Chiral Quarks and Physical Quark Masses," *Phys. Rev. Lett.*, vol. 113, p. 082 001, 2014. https://doi.org/10.1103/PhysRevLett.113.082001. https://arxiv.org/abs/1402.5175.

[7] Larmor, J., "LXIII. On the theory of the magnetic influence on spectra and on the radiation from moving ions," Dec. 1897. https://doi.org/10.1080/14786449708621095.

[8] S. Navas *et al.*, "Review of particle physics," *Phys. Rev. D*, vol. 110, no. 3, p. 030 001, 2024. https://doi.org/10.1103/PhysRevD.110.030001.

[9] ALICE collaboration, "Technical Design Report for the Upgrade of the ALICE Inner Tracking System," 2014. https://doi.org/10.1088/0954-3899/41/8/087002.

[10] I. Ravasenga, "Commissioning and Performance of the New ALICE Inner Tracking System in the First Phase of LHC Run 3," 2023. https://doi.org/10.48550/arXiv.2302.00432.

[11] ALICE collaboration, "ALICE Time Projection Chamber," ALICE Time Projection Chamber : Technical Design Report, 2000. https://cds.cern.ch/record/451098.

[12] ALICE Collaboration, "Upgrade of the ALICE Time Projection Chamber," 2013. https://cds.cern.ch/record/1622286.

[13] M. Kleiner, "Drift-Field Distortions and Specific Energy Loss Calibration of the ALICE TPC in LHC Run 3," 2024. https://doi.org/10.21248/gups.88271.

[14] H. Bethe, "Zur Theorie des Durchgangs schneller Korpuskularstrahlen durch Materie," *Annalen der Physik*, vol. 397, no. 3, pp. 325–400, 1930. https://doi.org/https://doi.org/10.1002/andp.19303970303.

[15] F. Bloch, "Zur Bremsung Rasch Bewegter Teilchen beim Durchgang durch Materie," *Annalen der Physik*, vol. 408, pp. 285–320, 1933. https://doi.org/10.1002/andp.19334080303.

[16] L. D. Landau, "On the Energy Loss of Fast Particles by Ionisation," *J. Phys. (USSR)*, vol. 8, D. ter Haar, Ed., 1944. https://doi.org/10.1016/b978-0-08-010586-4.50061-4.

[17] ALICE Collaboration, "Time resolution of the ALICE Time-Of-Flight detector with the first Run 3 pp collisions at $\sqrt{s}$ = 13.6 TeV," 2025. https://arxiv.org/abs/2511.10311.

[18] ALICE Collaboration, "The ALICE Transition Radiation Detector: construction, operation, and performance," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 881, pp. 88–127, 2018, ISSN: 0168-9002. https://doi.org/10.1016/j.nima.2017.09.028.

[19] "Space-point calibration of the ALICE TPC with track residuals," *EPJ Web of Conferences*, vol. 245, 2020. https://doi.org/10.1051/epjconf/202024501003.

[20] ALICE collaboration, "Technical Design Report for the Upgrade of the Online-Offline Computing System," 2015. https://cds.cern.ch/record/2011297.

[21] J. Adolfsson, et al., "SAMPA Chip: the New 32 Channels ASIC for the ALICE TPC and MCH Upgrades," *JINST*, 2017. https://doi.org/10.1088/1748-0221/12/04/C04008.

[22] J. von Neumann, "Theory of self-reproducing automata," A. W. Burks, Ed., 1966.

[23] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering*, 1960. https://doi.org/10.1115/1.3662552.

[24] R. Frühwirth, "Application of Kalman Filtering to Track and Vertex Fitting," *Nuclear Instruments and Methods in Physics Research Section A*, pp. 444–450, 1987. https://doi.org/10.1016/0168-9002(87)90887-4.

[25] D. Rohr, "Improvements of the ALICE GPU TPC tracking and GPU framework for online and offline processing of Run 3 Pb-Pb data," 2026. https://arxiv.org/abs/2601.20569.

[26] S. Gorbunov, et al., "ALICE HLT High Speed Tracking on GPU," *IEEE Transactions on Nuclear Science*, vol. 58, no. 4, pp. 1845–1851, 2011. https://doi.org/10.1109/TNS.2011.2157702.

[27] D. Rohr, "On development, feasibility, and limits of highly efficient CPU and GPU programs in several fields," 2014.

[28] D. Rohr, et al., "GPU-accelerated track reconstruction in the ALICE High Level Trigger," *Journal of Physics: Conference Series*, vol. 898, no. 3, p. 032 030, Oct. 2017. https://doi.org/10.1088/1742-6596/898/3/032030.

[29] M. Lettrich, "Fast Entropy Coding for ALICE Run 3," *PoS*, vol. ICHEP2020, 2021. https://doi.org/10.22323/1.390.0913. https://arxiv.org/abs/2102.09649.

[30] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958. https://doi.org/10.1037/h0042519.

[31] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, 1943. https://doi.org/10.1007/BF02478259.

[32] D. E. Rumelhart, et al., "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986. https://doi.org/10.1038/323533a0.

[33] Y. N. Dauphin, et al., "Identifying and Attacking the Saddle Point Problem in High-Dimensional Non-Convex Optimization," *NIPS*, vol. 27, Jun. 2014.

[34] L. Bottou, et al., "Optimization Methods for Large-Scale Machine Learning," *SIAM Review*, vol. 60, no. 2, pp. 223–311, 2018. https://doi.org/10.1137/16M1080173.

[35] H. Robbins and S. Monro, "A Stochastic Approximation Method," *Annals of Mathematical Statistics*, vol. 22, no. 3, pp. 400–407, 1951.

[36] R. Ge, et al., "No Spurious Local Minima in Nonconvex Low Rank Problems: A Unified Geometric Analysis," *Journal of Machine Learning Research*, vol. 18, no. 1, pp. 1–32, 2017.

[37] J. B. Diederik P. Kingma, "Adam: A Method for Stochastic Optimization," 2014. https://arxiv.org/abs/1412.6980.

[38] Y. Lecun, et al., "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. https://doi.org/10.1109/5.726791.

[39] A. Krizhevsky, et al., "ImageNet Classification with Deep Convolutional Neural Networks," vol. 25, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., 2012. https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[40] J. Kaplan, et al., "Scaling Laws for Neural Language Models," *CoRR*, 2020. https://arxiv.org/abs/2001.08361.

[41] B. P. Welford, "Note on a Method for Calculating Corrected Sums of Squares and Products," *Technometrics*, vol. 4, no. 3, pp. 419–420, 1962. https://doi.org/10.1080/00401706.1962.10490022.

[42] D. H. D. West, "Updating mean and variance estimates: an improved method," *Commun. ACM*, vol. 22, no. 9, pp. 532–535, Sep. 1979, ISSN: 0001-0782. https://doi.org/10.1145/359146.359153.

[43] P. Villalobos, et al., "Machine Learning Model Sizes and the Parameter Gap," 2022. https://arxiv.org/abs/2207.02852.

[44] J.A. Hanley, "The Meaning and Use of the Area Under a Receiver Operating Characteristic (ROC) Curve," *Radiology*, vol. 143, pp. 29–36, 1982. https://doi.org/10.1148/radiology.143.1.7063747.

[45] C. Sonnabend, "Neural network regression for particle identification with the ALICE TPC detector in Run 3," 2023. https://repository.cern/records/dyj3q-4kd28.

[46] C. Sonnabend, "Machine learning techniques for PID calibration in the ALICE TPC," LHCP conference, 2025. https://doi.org/10.22323/1.499.0128.

[47] R. W. Assmann, R. Johnson, and Z. Feng, "Calibration of the ALEPH dE/dx," 1994. https://cds.cern.ch/record/806004.

[48] J. Podolanski and R. Armenteros, "III. Analysis of V-events," *Phil. Mag.*, vol. 45, no. 360, pp. 13–30, 1954. https://doi.org/10.1080/14786440108520416.

[49] M. Slupecki, "Fast Interaction Trigger for ALICE upgrade," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 1039, p. 167 021, 2022, ISSN: 0168-9002. https://doi.org/https://doi.org/10.1016/j.nima.2022.167021.

[50] R.T. Solangel, "Status of the Fast Interaction Trigger detector for the ALICE upgrade," *PoS*, vol. EPS-HEP2021, 2022. https://doi.org/10.22323/1.398.0795.

[51] C. Sonnabend, "Particle identification in ALICE and LHCb," *PoS*, vol. LHCP2023, p. 051, 2024. https://doi.org/10.22323/1.450.0051.

[52] C. Sonnabend, "Machine learning in online/offline calibration and reconstruction at the LHC," *PoS*, vol. LHCP2024, p. 151, 2025. https://doi.org/10.22323/1.478.0151.

[53] G. Aarts, et al., "Heavy-flavor production and medium properties in high-energy nuclear collisions - What next?" *Eur. Phys. J. A*, vol. 53, no. 5, p. 93, 2017. https://doi.org/10.1140/epja/i2017-12282-9. https://arxiv.org/abs/1612.08032.

[54] T. Akiba, et al., "Optuna: A Next-generation Hyperparameter Optimization Framework," 2019.

[55] ONNX Runtime developers, *ONNX Runtime*, https://onnxruntime.ai/, Version: 1.21.0, 2021.

[56] B. Jacob, et al., "Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2704–2713. https://doi.org/10.1109/CVPR.2018.00286.

[57] H. Li, et al., "Visualizing the Loss Landscape of Neural Nets," pp. 6391–6401, 2018. https://doi.org/10.3929/ethz-b-000461393.

C Acknowledgment

The last three years have shaped me more than I could have ever imagined, as a person, as a scientist, as a friend. I am truly thankful for the opportunity I was given to conduct my research at CERN and get to know the many great people I met along the way. In the end, the research is exciting but what lasts are the memories and stories.

I would like to express my sincere gratitude to my supervisors David Rohr and Silvia Masciocchi for their untiring support and guidance to make this project a success. Davids endless patience with my sometimes silly questions and ideas is astonishing. I couldn't be more grateful to work under his supervision, one of the most kind and brilliant humans I had the pleasure to meet along my journey. Aside from the academic aspect, he is a great skiing teacher and will now be personally responsible for the money I invest into wintersports and mountaineering gear. To the many adventures that are about to come!
This gratitude extends to the entire PDP corridor: Sandro, Marco, Gabriele, Peter, Lubos, Giulio, Anton, Costin, Sergio, Marta, Max, Felix, Ernie, Ruben, Latchezar, Sean, Oliver and Chiara. With our coffee debates in the morning, at the cafeteria in building 13 or Costin's office, about life the universe and everything. I always feel at home in this corridor. And it extends to the EPN team, whenever I !unintentionally! crash one (or multiple) nodes again: Lubos, Giada and Federico, I am truly sorry and thankful at the same time.

I would also like to thank and mention the Gentner programme which enabled me to perform this research at CERN in the scope of the doctoral student programme. This work has been sponsored by the Wolfgang Gentner Programme of the German Federal Ministry of Education and Research (grant no. 13E18CHA).

I want to thank all the precious people that I had the pleasure of meeting at CERN and otherwise, too many to fit in these pages with all the great stories attached. Some people who have grown especially close and with whom I share great memories: Nico, Florian, Toni and Felix. Always down for adventure, between lasertag and O'Brasseur, you truly make life on the French countryside fun.

To all the people whom I unfortunately meet only infrequently but who nevertheless made a lasting impact on me with support and friendship: Nicolo, the rawest combination of brilliance

and kindness and one of the best ski instructors there is, and Basti, the aspiring ironman world champion (one day), who is responsible for the fact that my investments only flow into road bikes and race registration fees.

To the poker round: Bogdan, Pascal, Jonathan, Cas and Kai. Gentlemen, just to make it clear, the belt is mine!

To Kai, with whom I've become great friends over the years. An outstanding and brilliant physicist, a true friend who kept believing in me when times got tough and doubt took over.

And finally to my family and close friends who shape every step of the way, the rock on which I stand.

You all live in my heart and mind.

A final remark

Doing a PhD is a mix between skill and luck. The amount of hours needed to conduct such a work are not to be underestimated and are on occasion unavoidable, even with great efficiency. The hard work is necessary, without cutting corners, a lesson I have learned in many ways. In contrast to the figures of the TPC charge distributions per row (see e.g. figure (2.1.2)) shown in this thesis, figure (ACK.1) shows the Github commit activity over the last three years (darker color means more commits). This excludes activities on Gitlab or other platforms, but is a nice representation of the work that went into producing the results you have seen.

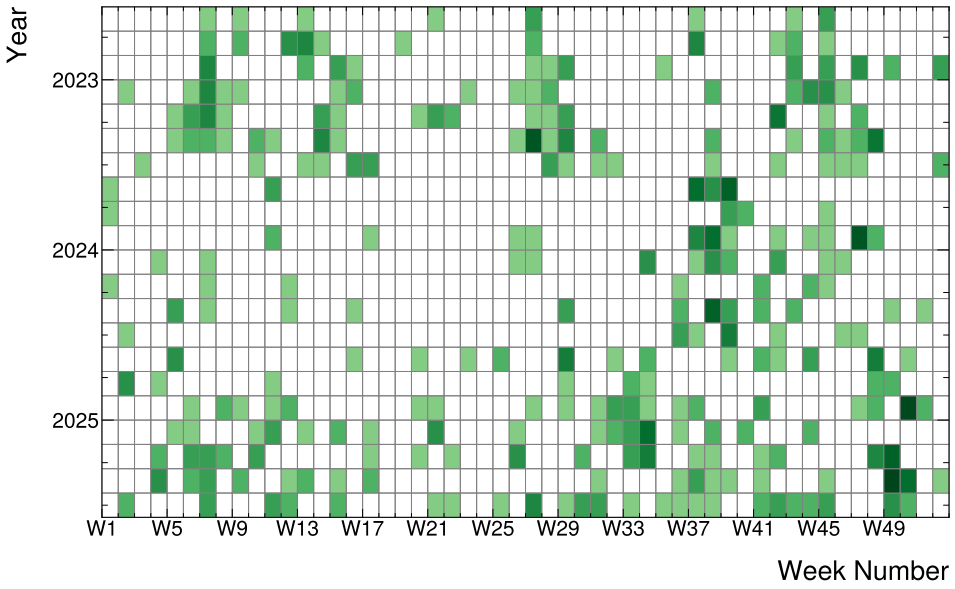


Fig. ACK.1: Github commit history over the last three years. Its been a lot of work.

Don’t ask me to find clusters here.

Usage of Large Language Models and AI assistance

Throughout the work conducted in this thesis the Large Language Model (LLM), ChatGPT (versions 5.0, 5.1, 5.2), released by OpenAI, was used. As this thesis deals with the topic of neural networks, this technology is not alien to me. I rather embrace its benefits to speed up work in several areas, concretely for code generation. An example is the generation of simple ROOT or python scripts to read and write data from and to files or bash functions to monitor large sets of slurm jobs. While all of this is well documented, handwriting the code even with perfect prior knowledge of the API is often a long and error-prone task without any learning process. Another example is the aid in implementation of multi-GPU training of PyTorch models in my (self-written) neural network class. This is a conceptually easy but time consuming task with poor documentation for trouble shooting, perfectly suited for a LLM. However, in many cases the generated output was flawed or required several iterations and manual rewriting.

While the main use-case of LLMs is the generation of text, at no point was an LLM used to generate large sets of words or entire sentences for this thesis. Questions like "Do you think this chapter is coherent" are however a good indicator for potential logical flaws and help me to see where an uninitiated reader might struggle. The intellectual content and ideas pursued in this thesis remain my own, with careful review of any incorporated output generated by the LLM. Nevertheless it is a useful tool that has significantly increased my efficiency.