Developing Machine Learning Models for Proton Computed Tomography and LHCb Particle Tracking

UNIVERSITY OF LIVERPOOL

Thesis submitted in accordance with the requirements of the University of Liverpool for the degree of Doctor in Philosophy by

Thomas Ackernley

Department of Physics

Oliver Lodge Laboratory

University of Liverpool

September 2025

Abstract

This thesis describes novel, proof-of-concept machine learning models for particle tracking in fundamental research and medicine. Proton computed tomography is a medical imaging technology with the potential to improve on current medical proton therapy treatment planning, but hampered by the computationally costly need to reconstruct individual proton tracks. With this in mind, we developed the Proton Path Neural Network, a neural network model capable of matching, and in some situations exceeding, the performance of the standard reconstruction method, with a significantly shorter execution time. Building on this experience, we turned to pattern recognition within track reconstruction at the LHCb experiment, one of the four major detector experiments located at CERN's flagship LHC particle accelerator. Focusing on reconstruction within the VELO tracking subdetector, we developed a graph neural network approach with the capacity to draw inferences from all measurements made by the subdetector for a given event, which exhibited promising performance over existing trials.

Acknowledgements

I would like to thank everyone whose knowledge, guidance and particularly support has made this thesis possible. Thank you to my supervisors for your mentorship and expertise throughout; to Themis Bowcock for his enthusiasm and support; to Kurt Rinnert for his help getting started with LHCb tracking; and to Marco Cristoforetti and his group for helping me settle in and navigate living abroad. I am eternally grateful to Tara Shears, David Hutchcroft, Stephen Farry and everyone in the LHCb Liverpool Group for all your help and making me feel at home. Many thanks also to Neil McCauley and my examiners Joe Price and Rebecca Chislett for your support in making it through the final stretch. I would like to extend thanks to my peers and friends for making my time so enjoyable, and somehow keeping me sane. Thank you also to my family, and to William and Hope for their help with proof-reading and prodding to keep me on track towards the end. And I would like to express my deepest thanks to my Mum, for getting me through the highs and lows and your continued unconditional support, without which I would never have got here.

And finally, thank you to everyone I inevitably forgot and will kick myself for having done so.

I would like to acknowledge the support provided by the University of Liverpool; Liverpool John Moores University; the Fondazione Bruno Kessler (FBK); CERN and LHCb Collaboration; the LIV.DAT centre for doctoral training; the Science and Technology Facilities Council (STFC); and the Barkla High Performance Computing facilities at the University of Liverpool.

Declaration

I hereby confirm this work is my own, except where other works are referenced or otherwise indicated. This work has not previously been submitted to any institute, including this one.

Thomas Ackernley

Contents

Part I

Introduction

Chapter 1

Introduction

We have always sought to understand the world around us. From geometry and the motion of planets, to quantum mechanics and beyond, humankind has built up a remarkable picture of the workings of reality. Yet that picture is still incomplete.

Using complex particle accelerators to recreate conditions at the Big Bang, scientists work to probe the fundamental building blocks of matter and forces; seeking answers to big questions on a scale too small to see. Here, particle tracking detectors seek to record the flight of individual particles, reconstructing particle tracks from their measurements to understand what unfolded at a subatomic scale. Even though large scale experiments may be the first that come to mind, particle tracking is not only the domain of pure research. Among other uses, particle tracking is key to many forms of medical imaging, where information deduced from particles and their paths is used to reconstruct the anatomy of the patient they traverse.

The last few years have seen a veritable explosion in machine learning; algorithms that can learn from data, approximating complex processes without needing them first to be formally defined, or even understood. Given the vast and complex quantities of data modern particle detectors can produce, machine learning techniques have emerged as an invaluable tool for fundamental research.

Across this thesis, I will explore two investigations into harnessing cutting-edge deep learning methods for particle tracking tasks, predominantly carried out between 2018 and 2022. This work was conducted under a scheme to spend time between pure research and industry applications; as part of a data science program funded by the Science and Technology Facilities Council.

Time at the Fondazione Bruno Kessler in Trento, Italy, as part of the LIV.DAT doctoral training centre program, led to investigations into machine learning for proton computed tomography, or pCT; a medical imaging technology using protons

to map a patient's anatomy. Using the same particle for treatment and planning, pCT has the potential to improve on current proton therapy treatment planning, but is hampered by the computationally costly need to reconstruct individual proton tracks[1,2]. From this emerged the Proton Path Neural Network, a proof-of-concept model capable of matching and in some situations exceeding the performance of the standard reconstruction method, with a significantly shorter execution time.

Subsequently brought to publication in a peer-reviewed journal[3], this work laid the foundation for tackling particle tracking challenges at the LHCb experiment. With a core program of Charge-Parity violation and rare beauty and charm hadron decays, LHCb is the Large Hadron Collider's dedicated flavour experiment and one of the accelerator's four large detectors[4,5,6]. As the tracking subdetector sitting immediately around the collision point itself, track reconstruction within the LHCb VELO is a vital component in understanding collision events at the detector. With this in mind, we investigated the potential of a machine learning model for pattern recognition with the capacity to draw inferences from all measurements made by the subdetector for a given event. This culminated in the development of an overarching framework and demonstration of a working graph neural network approach that successfully grouped measured particle positions into tracks, and showing promise over existing trials.

1.1 Structure

Owing to the varied nature of the work presented, this thesis has been structured into several parts as follows.

- Part I provides an introduction to material common throughout. Chapter 2 discusses particle accelerators, detectors and tracking, and Chapter 3 provides a brief introduction to machine learning and neural networks.

- Part II focuses on track prediction for proton computed tomography. An introduction to proton computed tomography is given in Chapter 4, and aims and method are described in Chapter 5. Results are presented in Chapter 6, with concluding discussion in Chapter 7.

- Part III then turns to track reconstruction for the LHCb VELO. Chapter 8 provides an introduction to graphs and graph neural networks, while Chapters 9, 10 and 11 discuss the LHC, LHCb and the VELO respectively. Aims

and Method are described in Chapter 12, and results presented in Chapter 13. Discussion of this work is given in Chapter 14.

- Finally, Part IV draws the previous two parts together, with concluding remarks from across the whole work in Chapter 15, followed by figure and table lists, and bibliography. Appendices are included in Part V.

1.2 Naming Conventions

Given the varied domains touched upon in this work, there are some terms which, depending on the field, often appear with different meanings. In an effort to avoid confusion we will be using the following conventions;

- A *graph* denotes the mathematical structure, described further in Section 8. Figures containing a graphical representation of data are referred to as *plots*. Neural networks and diagrams depicting them are referred to as *networks*.

- The abbreviation *MLP* denotes the Most Likely Path formalism, as described in Section 4.3.3, while a Multi Layer Perception will instead be referred to as a *perception*, or by referring to the wider concept of a *feed-forward neural network*, or *FNN*.

- A *tensor* is used here to refer to an n-dimensional array, as opposed to the rigorous mathematical structure.

Chapter 2

Working with Particles

2.1 Particle Detectors

Subatomic particles operate on a scale that our own senses are simply not precise enough to discern, let alone sufficient to make precise measurements with. Therefore we rely on particle detectors of one form or another to sense for us, leveraging particle interactions with matter to produce discernable signals for us to interpret[7]. Though the complex machines used in high energy physics may be the first to come to mind, a particle detector can be as simple as a photographic plate; passing photons interacting with crystals to produce blackening visible to the eye[8]. Depending on its intended purpose, a detector might look to measure some specific parameter, such as the quantity of certain particles to pass through an area, or it might endeavour to provide a comprehensive picture of everything that takes place within a set scope. In the latter case, detectors are often composed of a myriad of component detectors, designed so that particles ideally undergo interactions across multiple sensor elements in order to provide a wide range of different measurements[7]. Many research detectors, such as the 4 large experiment detectors at the Large Hadron Collider, follow this approach.

While fundamental scientific research has often driven the development of detectors and detection methods, modern particle detectors can be found in a diverse range of applications[7]. In medicine, many diagnostic imaging techniques, such as x-ray, CT and PET scanning, use particle detectors to make the precise radiation measurements from which medical images are constructed[9]. Equally, various forms of particle detector are a common sight for those working with ionising radiation, with Geiger counters offering a means to measure current radiation levels, and personal dosemeters widely employed for monitoring individual exposure[10]. Further

afield, examination of carbon-14 levels allows archaeologists to estimate the age of organic materials through radiocarbon dating[11], while geologists probe subsurface materials through neutron borehole logging[12].

2.2 Particle Tracking

One component in understanding events unfolding on the subatomic scale is knowing the position and momentum of the particles involved, parameters referred to as a particle's kinematic properties[13]. Responsibility for determining these properties is the domain of tracking; the combination of systems and analysis processes responsible for recovering the path, or track, taken by one or more particles through space. By applying a known magnetic field to at least a portion of the region covered by a tracking detector, or tracker, it is then possible to deduce the momentum of charged particles through their deflection[7]. In the context of high energy particle physics experiments, particularly looking at a wide range of particles, detector systems are often divided between systems for tracking, and those related to particle identification, which seek predominately to make measurements with which to determine the identity and nature of the particles detected.

Detector technology has come a long way since H. Becquerel's 1896 discovery of radioactivity using a photographic plate[7]. In the past, tracking detectors such as bubble and early spark and streamer chambers produced photographic readouts for researchers to manually inspect and measure, modern detectors typically capture a particles trajectory as a series of electronic signals, such as through fixed planes of sensors that register a particles location as it passes to produce a set of spatial coordinate measurements[14]. Current experimental tracking detector designs employ various common approaches to performing measurements; though regardless of the approach used, tracking detectors aim to disturb a particles flight as little as possible, and therefore generally present the minimum amount of material to a traversing particle that they can[13]. Equally, any measurements made must still be analysed and interpreted in order to reconstruct the tracks of the observed particles.

2.2.1 Gaseous Tracking Detectors

Gaseous tracking detectors such as a multiwire proportional chamber, or MWPC, operate around a large, gas filled volume. In a typical MWPC configuration, sheets of anode wires are sandwiched between cathode planes, with a voltage applied between

them[7,15]. As a charged particle passes through the detector, the gas is locally ionised along its trajectory. Electrons and ions drift towards anodes and cathodes respectively, where a current signal is induced. Given a sufficiently high voltage, the electrons produce an ionisation cascade on approach, resulting in a charge cloud around the anode, amplifying the signal measured. Stacking layers allows a particles position to be determined from the wires it passed close by, building up a series of measurements[7,15]. Drift chambers build the same principal by measuring the drift time of electrons to the anode wires, allowing for the original ionisation to be further localised to a position in between wires, providing higher resolution. While similar configurations to a MWPC can be used, drift chambers are usually formed from cell like structures of a single anode wire surrounded by cathode wires, beams or shells, with such cells arranged in a regular pattern to covering the desired detector volume[7,15]. Taking the concept of using drift time measurements even further, the time projection chambers consist of a large gas volume with an applied electric field terminating at one end, with a planar arrangement of alternating anode and cathode wires. This eschews the usual use of multiple layers of sensors to produce a sequence of coordinates. Charged particles produce corkscrew trajectories within the volume, electrons from ionisation drifting to the end cap to form a projection of the particles path, with displacement from the end cap determined using drift time measurements[15].

Developed to meet the increasing occupancy and resolution demands of experimental physics, micro-strip gas chambers use narrow strips engraved onto an insulating support in place of wires[7,15]. While this approach can achieve a spacing and strip thickness on the micrometer scale, far finer than that obtainable with wires, it is sensitive to discharges, and so prone to damage, at the voltages required for working with minimally ionizing particles[15,16]. Gas electron multiplier designs alleviate this issue through the introduction of a pre-amplifier; one or more thin polymer foil sheets, clad with metal on both sides, and perforated with holes on the micrometer scale. These are placed in between an anode strip covered surface and a parallel cathode plane, or equally between the wires of a MWPC. The electric field is 'squeezed' within the foil gaps, allowing for a lower voltage to be used while retaining the field necessary for the signal amplifying ionisation cascade within the holes, bringing the process away from the anodes themselves[15,17].

Gaseous tracking detectors pose various mechanical challenges, such as containment of gas, ensuring charged wires remain taught and degradation due to prolonged radiation exposure; and have begun to fall out of favour as dedicated tracking systems

for experimental high energy physics detectors[7,14]. Nevertheless, gaseous tracking detectors can still be found at the forefront of research, such as in the ALICE[18,19] and g-2[20] experiments; and are frequently employed for position sensitive detection in muon systems, which typically require coverage of large areas, such as those in ATLAS[21], CMS[22] and LHCb[23].

2.2.2 Solid State Tracking Detectors

Turning now to solid state designs, semiconductor detectors employ thin layers of a chosen semiconductor, exploiting that as a charged particle passes through the material, electron-hole pairs are generated. The resultant charges are collected and amplified, producing a readable signal. Strip detectors use layers segmented into parallel strips, whereas pixel detectors instead use sheets of discrete tiles. While strip detectors only measure one spacial coordinate, they are far easier to readout as the number of strips scales linearly with size, and as each strip reaches the edge of the detection area there is a convenient place to connect readout chips. Silicon is by far the most common medium employed, though there is exploration of other semiconductor materials for small scale trackers[7]. While silicon detectors come with various technical challenges and a significant cost, such detectors offer high spatial resolution are widely used in cutting edge experiments, such as ATLAS, CMS and LHCb[7].

Monolithic pixel detectors are a variant of silicon pixel detectors. As both the active sensor and readout chip in a silicon pixel detector are composed of silicon, the concept is to fashion the two components as a single piece. Though presenting significant technical challenges, with advances in fabrication methods such detectors have begun to see practical use, including the current and future ALICE inner tracking system[7,24].

Originally conceived as a form of memory device[25], Charge-Coupled Devices, or CCDs, have been employed as optical sensors for decades[7]. Charges are generated in semiconductor capacitors, but then transferred around the sensor to readout nodes. Capable of high spacial resolution, but with slower readout times and a sensitivity to radiation, CCDs have many successes in astrophysics. In high energy experiments however, they have largely been supplanted by the aforementioned forms of silicon semiconductor detector[7,25].

Unlike clear optical fibres, scintillating fibres consist of a scintillator doped core, surrounded by cladding of a lower refractive index[26]. As particles traverse the fibre, energy deposited in the core causes excitation and the emission of light. The cladding

traps some of this light by total refraction, guiding it to the fibres end and capture by photodetectors, such as silicon photomultipliers[26,27]. Scintillating fibres provide a relatively low cost and low material approach, though for high energy physics radiation damage can pose a challenge, and has recently been employed for post magnet tracking in the LHCb detector[27,28].

2.2.3 Track Reconstruction

Even neglecting the consequences of the uncertainty principal, practical constraints preclude us from being able to continuously measure everything we may want at all times. Equally, we cannot always directly measure those properties we wish to know, and therefore must determine what has taken place and the nature of those particles involved from the measurements we do make. Ultimately, a detector's various readout elements produce a variety of signals in response to one or more particles passage through the detector, and those signals need to be processed and interpreted[23]. Track reconstruction is the process of interpreting those measurements made by a tracking detector in order to recover the paths taken by particles, and is typically divided into three tasks; clustering, pattern recognition and fitting[13,29].

As particles traverse the tracking subsystems of a detector, they interact with sensitive elements, activating them to produce signals announcing their presence at a corresponding location; forming a series of spacial measurements, or hits *[13,29]. Depending on the form and granularity of a given detector, a particle may interact with multiple neighbouring elements, producing a representative region of signals. In such cases, clustering is performed, identifying and grouping readings into clusters corresponding to a single particle's passage, each interpreted as a single hit[29].

Then comes pattern recognition, or track finding, which consists of identifying those hits believed to correspond to the passage of the same particle, grouping them to form potential track candidates[13]. Where individual particles are sufficiently separated in time this process is relatively trivial. But where a detector will witness the passage of multiple particles in an indistinguishable time frame, such as the dearth of particles emerging from a single high energy collision, it becomes a complex classification task; particularly if dealing with curved tracks like those produced in magnetic fields. Various common pattern recognition techniques have been developed over the years, and can generally be distinguished as global methods, which adapt

*Though a hit may be used to refer to the signal of each activated sensitive element, we will use it to denote the input data used by the pattern recognition stage, after any interpreting and clustering has been performed.

and solve an equivalent formulation of the problem in order to address the task simultaneously as a whole, or local methods, which iteratively construct tracks by examining groups of hits at a time[13,30].

Conformal mapping, a form of spacial transformation that preserves the angle between any two lines, can be employed as a global method when working with curved tracks. This exploits that circular tracks passing through or close to the origincan in this way be mapped into a coordinate system where they form straight lines or parabola. In the new coordinate system the azimuthal angle of hits forming such a track will be close to one another. Thus tracks can be found by collecting angular components into a histogram, with a peak indicating the corresponding hits line up to form a potential track[13,31]. The Hough transform offers a similar approach when working with straight tracks that do not necessarily pass close to the origin. This instead uses that a point in the $x - y$ plane can, through the straight line equation $y = cx + d$, be mapped to a line in the $c - d$ plane, $d = -cx + y$; and exploits that points which lie along the same straight line in the x-y space map to lines in c-d that cross at a particular point[13,32]. Along the same lines, transforming drift chamber measurements into Legrande space leads tracks to appear as the intersection of the now sine curve representations[13,33]. With analogies to the Hough transform, the artificial retina algorithm builds heat maps and uses clustering to identify tracks in an approach inspired by straight line recognition within the eye[34,35].

Turning to local approaches, track road methods begin by selecting a set of hits which may have been created by the same particle. A potential trajectory, or road, is interpolated between them, and additional hits are assigned from those lying close to that prediction. The formed track candidates are then compared and assessed to determine which are likely to be tracks, often based on the number of hits successfully found along this trajectory and the quality of subsequent track fitting[13]. Track following, or forwarding, methods take a somewhat similar approach, initially forming seed tracks from hits identified as likely constituting a short track segment. This is often performed by looking at hits across neighbouring sensitive components, and focusing on regions of a detector where it is usually easier to distinguish hits made by the same particle, such as areas furthest from the collision point at a particle accelerator. Each seed track is then extrapolated progressively through the detector, picking up additional hits that lie close to the predicted path as sensitive elements are encountered, continuing until either the end of the detector is reached, or multiple sequential sensors fail to observe hits potentially continuing the track. Similar methods may then be used to determine the likely true tracks among those

formed.[13,29,30]. Though versatile, the iterative nature of such algorithms can present challenges for parallelisation[30].

Moving to track fitting, once track candidates have been formed out of the observed hits a smooth predicted trajectory is determined for each candidate which most accurately describes the path taken by the particle which left the corresponding hits[13,29]. This is commonly performed using a least squares fit approach, with many conventional methods employing variants of a Kalman filter[36,37]. Though Kalman filter techniques concern themselves with estimating variables of dynamic systems as they evolve, they can be applied to tracking by sequentially adding hits to the presented track, recursively building up and refining the predicted path as more hits are included[37]. Assessing the quality of fitted trajectories can be used in turn to evaluate if track candidates may instead represent non existent tracks formed from random combinations or to distinguish among incompatible candidates reusing hits between them[13,23,29], therefore forming a concurrent part of pattern recognition[13].

Depending on a detector's particular context, separate vertex reconstruction may also be performed as part of reconstructing an event; the task of accurately locating one or more common origin points of the various particles tracked[13]. Often the process is similarly divided into vertex finding, determining and grouping those tracks that likely originate from the same vertex, and vertex fitting, accurately locating that vertex in space[13]. Accurate vertex reconstruction is particularly important in heavy flavour physics research, as it allows for reconstructing short lived particles by identifying decay products and the common secondary vertex from which they came[13,36]. Using the displacement of this secondary decay vertex from the original primary collision vertex, it is then possible to determine the particle's decay length. Moreover, constraining tracks to originate at identified vertices can be used to further improve the quality of fitted trajectories[13].

2.2.4 Trigger Systems

Overall, the processing of experiment data can be divided into online processing, that which takes place in real time as data is collected, and offline processing, the separate processing of stored data at its own pace[7,13]. Given the stochastic nature of quantum physics, events produced at research particle accelerators are frequently of limited interest. While it may be attractive simply to store a detector's total output and perform all processing offline, the staggering quantities of events detectors at such experiments usually observe combined with finite storage space mean it is simply not possible[7]. The current LHCb configuration, for example, produces around

4 TB of raw data per second when running at nominal instantaneous luminosity, an infeasible quantity to simply store[23].

It is therefore common to employ a trigger; a system designed to reduce the quantity of events by discarding those that are unlikely to be of interest, usually by looking for a number of generic signatures that don't require significant analysis to identify. Triggers are, by their nature, closely intertwined with a detector's data acquisition, or DAQ, system, and a detector may employ multiple triggers, each reducing down the flow of events to allow more detailed, and so slower, assessment by subsequent triggers[7]. Given the demanding speed requirements online processing typically presents, simplified methods may be used to deduce a rough understanding of collected data, such as for a trigger system, with full quality event reconstruction and analysis later carried out offline. In detector experiments at particle accelerators, some form of simplified track and vertex reconstruction is often performed online as part of trigger systems[7,13].

2.3 Particle Accelerators

Though elementary particles may make up the known universe, we can't rely upon chance to present them as and when we want. Particle accelerators therefore offer a means to generate beams of specific particles on demand, with precise control over energies. From the ubiquitous cathode ray tube's key contribution to the discovery of the electron[38], to the numerous publications of the Large Hadron Collider, particle accelerators have built a widely established track record of groundbreaking discoveries. As will be discussed in Section 9.1, they are an invaluable tool in modern experimental physics, and increasingly sophisticated designs continue to push the boundaries of high-energy physics and discovery[39].

Yet just as with detectors, modern particle accelerators are not only the domain of pure research[40]. For decades the aforementioned cathode ray tube played a well known role in television displays, and accelerators now allow for the non-destructive analysis of artwork and artifacts[41]. Accelerators are widely used in medical settings, from diagnostic imaging, such in X-ray imaging equipment and radioisotope production for PET scans, to directly providing treatment in radiotherapy and the sterilization of medical equipment[42,43]. Turning to industry, accelerators have found various roles in material and earth sciences, such as material testing and semiconductor manufacturing[40,43].

2.3.1 Forms of Particle Accelerator

Particle accelerators take many forms, and can be divided into several types based on how they operate. Fundamentally, no one method can claim to be strictly superior, with each potentially more suitable for different applications. Owing to their distinct silhouettes and obvious differences, accelerators are broadly divided into linear and circular designs, although circular accelerators encompass a range of different forms[40].

Linear Accelerators, or Linacs, utilise oscillating electric fields to propel charged particles in straight line trajectories. With the capacity for precise control over beam energy, Linacs are a common sight in medical applications. However, the significant size needed in order to attain higher energies means that in modern research they are usually found as injectors, pre-accelerating particles for entry into larger, circular-style accelerators[44].

Synchrotrons accelerate discrete particle bunches in a circular path, using magnetic fields to guide particles and radio-frequency (RF) cavities, chambers with a strong directed electromagnetic field, to provide acceleration. With a closed loop path, particles can perform multiple revolutions, accumulating energy each time they pass the RF cavities. In order to keep particles on course, the magnetic field must be varied to compensate as their velocities increase, and thus cannot provide a continuous stream of particles[40]. Not only an essential tool in high-energy physics and the basis of modern high energy experiment accelerators such as the Large Hadron Collider[39], synchrotrons are also the archetypal method of generating synchrotron radiation. This electromagnetic radiation is emitted as relativistic charged particles are forced to turn, and supports material science and biological research[45].

Cyclotrons instead accelerate a continuous stream of particles in a spiral trajectory within a circular, disk shaped cavity. A constant magnetic field bends particles' trajectories, while an alternating electric field provides acceleration. Compact and cost-effective, cyclotrons are widely used in medical and industrial applications[40,46].

Synchrocyclotrons extend the principal of cyclotrons, adjusting the frequency of the electric field to account for relativistic effects as velocity increases[40]. This allows particles to attain higher energies, but at the cost of comparative reduced beam intensity.

Isochronous cyclotrons go further, increasing the magnetic field to match their energy as particles move outward along their spiral path. This increasing magnetic field compensates for relativistic effects and acts to maintain synchronization between the particles and the accelerating electric field[40]. The result is an accelerator

particularly suited for producing high-intensity particle beams.

Chapter 3

Machine Learning and Neural Networks

3.1 The Concept of Machine Learning

In general terms, the field of machine learning concerns itself with algorithms that learn from data or experience. Rather than being determined directly, tunable aspects of models are optimised algorithmically through exposure to examples. Though the terms are frequently used interchangeably, machine learning is a subset of the wider field of Artificial Intelligence, the study of replicating human intelligence and thinking; in this case, emulating the ability to learn and improve through experience. In many ways, machine learning is about the process of calibrating algorithms, rather than the nature of a final algorithm itself. Equally, machine learning is as much a field of statistical analysis, seeking to extract meaning from data[47,48].

To illustrate the concept, let us imagine we have a function we wish to model. One direction would be to generate the Fourier series or Taylor polynomial of the function. In either case, we have a set form, or model, for the representation, and procedurally derive the corresponding coefficients for our particular function through application of a formula. If we were instead to take a machine learning approach, rather than looking to the function itself, we would work with examples of the function, applying it to a range of inputs. From this data, we might apply a suitable model to the same examples, tuning our coefficient equivalents to recreate the function's behaviour. In this way, machine learning methods can often be seen as a form of trial and error, adjusting bit by bit to better produce the results we desire.

Consider for a moment electron drift in a wire under a small electric field. If at a given moment we were to examine a single electron, each would have a seeming

random velocity. Yet if we were to continue to select electrons, and record and average their velocities, over time the drift velocity will emerge from among the randomness. Calibrating, or training, of a machine learning model operates on a similar principal. Each individual example a model is optimised towards may be seemingly random and contradictory, but the aim is that as it is exposed to more and more examples, it converges towards representing any patterns present in the overall body of data.

This leads neatly to a key strength of machine learning; as algorithms are calibrated from data, we don't need to be able to express, or even know, the formal underlying rules behind a process in order to approximate it. Machine learning algorithms are therefore ideally suited to tasks that as humans we find intuitive, but which are extremely hard to explain formally, such as recognising letters and numbers from their shapes or making conversation[47,48]. This adaptability and lessening of the need for prior knowledge has made Machine learning a cornerstone of many modern applications, ranging from natural language processing to autonomous vehicles. On the reverse side, great care must therefore be taken to ensure that any data used is representative of that which we want a model to learn, and be mindful of other, unintended patterns that may be present, or even introducing them ourselves with how the data is presented. As with other forms of statistical analysis, large datasets are therefore desirable as they are less likely skewed by random chance.

Given the scope and variety encompassed by machine learning, particularly as a field straddling statistical analysis and computing domains, exact definitions vary. Though we have endeavoured to remain general, this chapter is intended as a rough illustration of some general ideas in order to understand the work, and its context, discussed across this thesis.

3.2 Neural Networks

3.2.1 Deep Learning and Neural Networks

Though biological interpretations have largely fallen by the wayside, artificial neural networks, or more commonly just neural networks, are loosely inspired by the connection structure of the brain[47].

At its core, a neural network consists of a number of units known as neurons. Each neuron itself only performs a simple computation; it receives a series of input variables, performs a linear combination using a weightings map, and passes it

through a non-linear activation function to produce a single output variable. Typically, neurons are organised into sets known as layers, with results from neurons in one layer usually passed to neurons in another, though more complex arrangements are possible. Neuron weights are the tunable parameters of a neural network, and are calibrated through backpropagation; calculating the gradients of network parameters using the chain rule to jointly adjust all weightings maps throughout the network at once, moving the model towards producing a desired output for a given input each time. While neurons are individually simple, the result is greater than the sum of its parts. Through alternating linear and non-linear operations, neural networks have proven capable of modelling complex non-linear processes[47,48].

Deep learning concerns itself with neural network models with many layers of neurons, which are optimised jointly from data as opposed to individually calibrating each layer to perform a specific task. This enables an algorithm to learn its own intermediate knowledge, building up complicated concepts out of simpler ones of its own determining[47,49]. However, their abstract nature makes understanding the meaning behind how and why a model reaches its results challenging, and explainable AI with deep neural networks (DNN) remains a significant field of study[50]. Though we may know the exact calculations a model performs, interpreting what those calculations represent is often elusive.

While DNN are capable of efficiently approximating complex tasks, they are computationally intensive, particularity during training; and significant efforts have been invested into operating deep learning models on specialised hardware platforms. Being extremely efficient for floating-point matrix-based calculations, and coupled with high throughput and memory band widths, GPU's have become the go to general purpose platform for deep learning. Dedicated ASIC chips can provide superior performance with less energy use, but there are significant costs and drawbacks to developing and fabricating dedicated chips for specific purposes. Field Programmable Gate Arrays, or FPGAs, offer a programmable architecture alternative to ASICs, bringing reconfigurability at the loss of some performance[51].

3.2.2 A Basic Neural Network

With the variety in modern neural network models, it is difficult to be entirely general. Therefore we will introduce what is frequently described as the standard or 'vanilla' deep neural network model; the feed forward, fully connected neural network[48]. In a feed forward neural network, or FFNN, layers are arranged in a clear sequence structure, with neurons in one layer passing their outputs onto those

in the next. In this way, neurons are only connected to those on neighbouring layers, and there are no cycles or skips in the structure. fully connected neural network, or FCNN, utilises only fully connected, or dense, layers, where every neuron receives the output of all neurons in the preceding layer as inputs. A well proven design, networks of this form are effective for tasks like tabular data classification and regression, and are often used as component sections within more complex architectures [49].

3.2.3 Anatomy of a Neural Network

To begin, let k denote a neuron with a set of m input values x_i, for $1 \in [1, m]$. The output y_k of neuron k is then given by

$$y_k = \sigma\left(b_k + \sum_{i}^{i=1} w_{k_i} x_i\right) \tag{3.2.1}$$

where w_{k_i} and b_k are the set of weights and the bias term respectively, and $\sigma()$ the chosen activation function, if there is one. The bias term b_k acts as an offset constant, and by instead defining this as w_{k_0} and fixing $x_0 = 1$, the above can be simplified to

$$y_k = \sigma\left(\sum_{i}^{i=0} w_{k_i} x_i\right) . \tag{3.2.2}$$

Depending on definition, the activation function is frequently considered as a separate layer to the neurons, operating on all outputs at once. Given each neuron in a fully connected layer receives the same inputs, this separation allows the layer to be performed as a matrix operation;

$$\mathbf{y} = \sigma(\mathbf{z}), \quad \mathbf{z} = \mathbf{W} \cdot \mathbf{x} \tag{3.2.3}$$

where,

$$\mathbf{y} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_k \\ \vdots \\ y_N \end{pmatrix}, \quad \mathbf{z} = \begin{pmatrix} z_0 \\ z_1 \\ \vdots \\ z_k \\ \vdots \\ z_N \end{pmatrix}, \quad \mathbf{W} = \begin{pmatrix} w_{0_0} & w_{0_1} & \dots & w_{0_i} & \dots & w_{0_M} \\ w_{1_0} & w_{1_1} & \dots & w_{1_i} & \dots & w_{1_M} \\ \vdots & \vdots & \ddots & \vdots & & \vdots \\ w_{k_0} & w_{k_1} & \dots & w_{k_i} & \dots & w_{k_M} \\ \vdots & \vdots & & \vdots & \ddots & \vdots \\ w_{N_0} & w_{N_1} & \dots & w_{N_i} & \dots & w_{N_M} \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} 1 \\ x_1 \\ \vdots \\ x_i \\ \vdots \\ x_M \end{pmatrix} \tag{3.2.4}$$

for a fully connected layer of N neurons with M input values, and where z_k represent the intermediate results of the neurons.

From here it is straightforward to generalise a whole feed forward fully connected neural network. Let C be a network of L layers, with initial inputs $\mathbf{x}$ and corresponding output $C(\mathbf{x}) = \mathbf{o}$. Let σ^l and $\mathbf{W}^l$ be as above, with the introduction of l to indicate layer $l \in [0, ..., L]$. Then for a given layer l receiving inputs $\mathbf{y}^l$, the output $\mathbf{y}^{l+1}$ is given by,

$$\mathbf{y}^{l+1} = \sigma^l\left(\mathbf{z}^l\right), \quad \mathbf{z}^l = \mathbf{W}^l \cdot \mathbf{y}^l \tag{3.2.5}$$

where $\mathbf{y^0} = \mathbf{x}$ and $\mathbf{y^{L+1}} = \mathbf{o}$; and with this we can finally express C recursively as,

$$C(\mathbf{x}) = \sigma^L(\mathbf{W^L} \cdot \sigma^{L-1}(\mathbf{W^{L-1}} \cdot ... \cdot \sigma^l(\mathbf{W^l} \cdot ... \cdot \sigma^0(\mathbf{W^0} \cdot \mathbf{x})...)...)) \tag{3.2.6}$$

for $l \in [0, ..., L]$ [47,48,49].

The choice of activation function depends on a range of factors, such as the application a network is designed for and the position of a corresponding neuron within the network. Though it must be non-linear and differentiable, there are a variety of potential activation functions available. Several commonly used functions are[47]:

- The Sigmoid Function,

$$\sigma(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{1 + e^x}. \tag{3.2.7}$$

  Mapping any real value into the $(0, 1)$ range, the sigmoid function is a particularly useful activation function for the final layer of binary classification networks. However, it suffers from vanishing gradients, which can slow down learning in deep networks[52].

- The Tanh Function,

$$\sigma(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}\,. \tag{3.2.8}$$

  Similar to the sigmoid function, the tanh function maps inputs into a set range, this time $(-1, 1)$, but similarly suffers with vanishing gradients for extreme input values[52].

- The Rectified Linear Unit, or ReLU,

$$\sigma(x) = \max(0, x)\,. \tag{3.2.9}$$

  Both simple and less susceptible to the vanishing gradient issues, ReLU has risen to be the most commonly used activation function. However, if it reaches a point where it is producing zeros for too many typical inputs, it can become stuck in a 'dead' state[53].

3.2.4 The Training Process

Though exact optimisation algorithms themselves vary, the optimisation process of a neural network model, or training, is built on the concept of gradient descent; an iterative method for finding the minimum of a function by adjusting against its gradient.

At its core, training revolves around evaluating a model in its current state for a given set of inputs, and adjusting the neuron weights to produce something closer to a desired output, known as the target. However, the overall algorithm containing the neural network structure, the nature of the desired target, and overall training procedures can vary greatly, obfuscating this core loop. While the process is here envisioned sequentially, with the power of modern computing it is usually performed on batches of input-and-target examples at a time.

Loss Functions

In order to optimise objectively, it is first necessary to define how a models performance should be quantified. This role is fulfilled by the loss function, sometimes known as the error function, which provides a numerical value characterising the difference between a model's output and the target. The specific choice of loss function depends on the form of task being tackled; and common choices include the Mean Square Error for regression tasks, and variation of the Cross-Entropy Loss for

classification[54]. Referring back to Equation 3.2.6, let $E(\mathbf{o}, \mathbf{t})$ be the loss function, where $\mathbf{t}$ is the target corresponding to inputs $\mathbf{x}$, and as before $\mathbf{o}$ the output of our network on the same. Formally, the objective of training is therefore to adjust the neuron weightings $\mathbf{W}^l$ in order to minimise the loss function $E(\mathbf{o}, \mathbf{t})$.

Backpropagation

Before we can update the neuron weightings, we need to know how adjusting them impacts the loss function. This is achieved through backpropagation; an efficient method of calculating the loss function gradients with respect to the neuron weights by traversing the network in reverse order, from the output to the input layer, using the chain rule and storing intermediate variables as we go[47].

Referring back to Equation 3.2.5 for a layer l, let H be a function of $\mathbf{z}^l$, $H = H(\mathbf{z}^l)$. Then the gradient of H can be expressed as *.

$$\frac{\partial H}{\partial \mathbf{W}^l} = \frac{\partial H}{\partial \mathbf{z}^l} \cdot \frac{\partial \mathbf{z}^l}{\partial \mathbf{W}^l} \tag{3.2.10}$$

$$= \frac{\partial H}{\partial \mathbf{z}^l} \cdot \frac{\partial \left(\mathbf{W}^l \cdot \mathbf{y}^l\right)}{\partial \mathbf{W}^l} \,. \tag{3.2.11}$$

As $\mathbf{y}^l$ is the input to layer l, it is independent of $\mathbf{W}^l$, so this becomes

$$\frac{\partial H}{\partial \mathbf{W}^l} = \left(\frac{\partial H}{\partial \mathbf{z}^l}\right)^T \mathbf{y}^l \,. \tag{3.2.12}$$

Now let J be a function of $\mathbf{y}^{l+1}$, $J = J(\mathbf{y}^{l+1})$. Then the gradient can be expanded as

$$\frac{\partial J}{\partial \mathbf{z}^l} = \frac{\partial J}{\partial \mathbf{y}^{l+1}} \cdot \frac{\partial \mathbf{y}^{l+1}}{\partial \mathbf{z}^l} \tag{3.2.13}$$

$$= \frac{\partial J}{\partial \mathbf{y}^{l+1}} \cdot \frac{\partial \left(\sigma^l\left(\mathbf{z}^l\right)\right)}{\partial \mathbf{z}^l} \mathbf{y}^l \,. \tag{3.2.14}$$

Expressing the right hand term using the derivative of the activation function, ϕ'^l, and element-wise multiplication, which we will denoted with $\odot$, this becomes

$$\frac{\partial J}{\partial \mathbf{z}^l} = \frac{\partial J}{\partial \mathbf{y}^{l+1}} \odot \phi'^l(\mathbf{z}^l)\mathbf{y}^l \,. \tag{3.2.15}$$

*Given the complexities of calculus involving tensor style objects, throughout this chapter assume any necessary operations such as transposition and swapping input positions are implied.

Returning to Equation 3.2.12, we can now expand it further,

$$\frac{\partial H}{\partial \mathbf{W}^l} = \left(\frac{\partial H}{\partial \mathbf{z}^l}\right)^T \mathbf{y}^l \tag{3.2.16}$$

$$= \left(\frac{\partial H}{\partial \mathbf{y}^{l+1}} \odot \phi'^l(\mathbf{z}^l)\right)^T \mathbf{y}^l \,. \tag{3.2.17}$$

In addition, looking at the gradient with respect to the layers input,

$$\frac{\partial H}{\partial \mathbf{y}^l} = \frac{\partial H}{\partial \mathbf{z}^l} \cdot \frac{\partial \mathbf{z}^l}{\partial \mathbf{y}^l} \tag{3.2.18}$$

$$= \frac{\partial H}{\partial \mathbf{z}^l} \cdot \frac{\partial \left(\mathbf{W}^l \cdot \mathbf{y}^l\right)}{\partial \mathbf{y}^l} \tag{3.2.19}$$

$$= \frac{\partial H}{\partial \mathbf{z}^l} \mathbf{W}^l \,. \tag{3.2.20}$$

Now let us return to our loss function E. First a forward pass is performed to calculate the models current output $\mathbf{o} = C(\mathbf{x})$, for our given set of inputs $\mathbf{x}$; retaining relevant intermediate variables as we go for use on the way back. Then, let us begin with the final layer, $l = L$. Remembering that $E = E(\mathbf{o}, \mathbf{t}) = E(\mathbf{y}^{L+1}, \mathbf{t})$, and as the target $\mathbf{t}$ is independent of all $\mathbf{W}^l$, we can use Equation 3.2.16 to say

$$\frac{\partial E}{\partial \mathbf{W}^L} = \left(\frac{\partial E}{\partial \mathbf{y}^{L+1}} \odot \phi'^l(\mathbf{W}^L \cdot \mathbf{y}^L)\right)^T \cdot \mathbf{y}^L \tag{3.2.21}$$

where $\frac{\partial E}{\partial \mathbf{y}^{L+1}}$ depends on the specific loss function used. Progressing to $l = L-1$ and the gradient with respect to $\mathbf{W}^{L-1}$,

$$\frac{\partial E}{\partial \mathbf{W}^{L-1}} = \left(\frac{\partial E}{\partial \mathbf{y}^L} \odot \phi'^l(\mathbf{W}^{L-1} \cdot \mathbf{y}^{L-1})\right) \cdot \mathbf{y}^{L-1} \,. \tag{3.2.22}$$

As E depends on $\mathbf{y}^{L+1}$, we can further expand the differential term,

$$\frac{\partial E}{\partial \mathbf{y}^L} = \frac{\partial E}{\partial \mathbf{z}^L} \mathbf{W}^L \tag{3.2.23}$$

which in turn can be expanded similarly to before using Equation 3.2.15,

$$\frac{\partial E}{\partial \mathbf{y}^L} = \left(\frac{\partial E}{\partial \mathbf{y}^{L+1}} \odot \phi'^L(\mathbf{z}^L)\right)^T \mathbf{W}^L \tag{3.2.24}$$

$$= \left(\frac{\partial E}{\partial \mathbf{y}^{L+1}} \odot \phi'^L(\mathbf{W}^L \cdot \mathbf{y}^L)\right)^T \mathbf{W}^L \,. \tag{3.2.25}$$

Given the prevalence of similar terms, let us define δ^l such that

$$\delta^l = \frac{\partial E}{\partial \mathbf{y}^{l+1}} \odot \phi'^l(\mathbf{W}^l \cdot \mathbf{y}^l)\,. \tag{3.2.26}$$

Using this we can then simplify our calculation for the gradient with respect to $\mathbf{W}^L$ (Equation 3.2.21) as

$$\frac{\partial E}{\partial \mathbf{W}^L} = \left(\delta^L\right)^T \cdot \mathbf{y}^L \tag{3.2.27}$$

and for $\mathbf{W}^{L-1}$ as

$$\frac{\partial E}{\partial \mathbf{W}^{L-1}} = \left(\delta^{L-1}\right)^T \cdot \mathbf{y}^{L-1} \tag{3.2.28}$$

where δ^{L-1} relates back to δ^L,

$$\delta^{L-1} = \left(\left(\delta^L\right)^T \mathbf{W}^L\right) \odot \phi'^l(\mathbf{W}^L \cdot \mathbf{y}^L)\,. \tag{3.2.29}$$

As we can see, calculating the gradient corresponding to the second to final layer involves repeating calculations performed for the final layer gradient, a pattern continuing down the network. Thus by working backwards through the network, intermediate calculations can be reused from previous gradients, avoiding a duplication of efforts[47,55].

Optimisers

With the results of backpropagation in hand, it is then time to adjust the neural network. Optimisation algorithms themselves, commonly known as optimisers, describe how updates to weightings are performed, and typically feature a learning rate hyperparameter to control the relative size of adjustments. For the work in this thesis, the Adam algorithm[56] was used throughout. Combining techniques from a range of different learning algorithms, the Adam algorithm has a proven record as a robust algorithm for training deep neural networks[47].

3.2.5 Forms of Neural Network Model

While we have so far discussed feed forward and fully connected neural networks, they are far from the only architectures available. Given the versatile nature of their core building blocks, various forms of neural network have been developed, enabling models that can successfully tackle a variety of challenges.

Recurrent neural networks, or RNN, operate principally on sequential data, maintaining information form previous inputs though cyclic neuron connections. This

makes them effective for tasks requiring progression or sequences as input, such as time series prediction and natural language processing. However, they can struggle with retaining long-term information, an issue Long Short-Term Memory, or LSTM, networks seek to address [57].

Convolutional neural networks, or CNN, use convolutional layers to operate on data with a grid-like topology. Learning spatial features, in convolutional layers neurons take the form of a filter or kernel, which is convoluted across its input object to produce feature maps, and are frequently combined with pooling layers for down-sampling. Many CNN models transition to a series of fully connected layers, flattening the feature map, enabling them to perform tasks such as image classification. Those that retain their structure throughout, known as fully convolutional neural networks, can operate on variable sized inputs, and are employed for tasks such as image generation or segmentation [47,49,58].

Deep Reinforcement Learning algorithms, such as Deep Q-Learning and Actor Critic, utilise neural network models which interact with an environment and learn from their past actions, effectively generating their own examples to work from. Optimising towards maximising a reward signal received from said environment, models are designed to carry out decisions sequentially, acting based on potential future gains rather than just immediate rewards [47,59].

Graph neural networks, or GNN, operate on relational, network-like structures, and are discussed in detail in Chapter 8.

3.3 Machine Learning in High Energy Physics

As the awarding of the 2024 Nobel Prize in Physics attests [60], machine learning and neural networks have emerged as key tools in modern fundamental physics research. Experimental high energy physics is no exception, and machine learning based methods are now commonplace throughout cutting edge experiments and analyses.

Since 2010, and as part of an online trigger reconstruction sequence from 2015, the LHCb experiment has employed neural network based classifiers to screen for fake charged particle tracks [5,61,62,63]; and by 2011, a machine learning based method was serving as the main trigger selections for beauty physics [64]. In neutrino experiments, both the NOvA [62,65,66] and DUNE [36,67] experiments utilise CNN to categorise neutrino interactions within the detector volume, and similar CNN approaches have been explored for calorimetry [68,69,70,71,72]. RNN have demonstrated success at beauty

jet identification[73,74] and recently ATLAS and CMS have been exploring GNN and attention based transformers for the task[75,76,77].

In analyses, Boosted Decision Trees have largely replaced traditional cut based methods for signal selection and background discrimination[36]. Over a decade ago, the CMS experiment employed Boosted Decision Trees to identify and categorize potential diphoton Higgs decay particles to make their first solo observation of the Higgs[78], while ATLAS's analysis leveraged neural networks[79]. The first evidence of beauty to muon anti-muon decays, carried out by CMS and LHCb, utilised Boosted Decision Trees[80], as did the first solo observation by LHCb[62,81], and the first observation of pentaquarks[82,83]

Given the vast quantities of simulated data produced by researchers, there is growing interest in machine learning for fast and ultra-fast simulation; reducing computing needs through approximated simulation methods such as resampling methods or parametrisation of detector response. The LHCb experiment has implemented a framework that includes machine learning based fast-simulation as part of its simulation package[23,84,85]. Meanwhile the ATLAS collaboration has explored the use of variational autoencoders and generative adversarial networks for simulating particle showers within electromagnetic calorimeters[86], and CMS has investigated sample re-weighting to generate samples for variant models without needing to re-simulate detector responses[77,87,88].

Part II

Proton Computed Tomography

Chapter 4

Hadron Therapy and Proton Computed Tomography

The work presented here in Part II has previously been published as

> T. Ackernley, G. Casse, and M. Cristoforetti. Proton path reconstruction for proton computed tomography using neural networks. *Physics in Medicine & Biology, 66(7):075015, apr 2021.* doi: 10.1088/1361-6560/abf00f. [3]
>
>

The following chapters therefore include material reproduced, paraphrased and expanded on from said publication, including text and figures. Accompanying declarations regarding the paper can be found in Appendix A. The full accepted manuscript can be found at https://arxiv.org/abs/2010.00427, and for completeness has been reproduced in Appendix D.

4.1 Radiotherapy

Before the century is out, cancer is predicted to surpass heart disease as the leading cause of premature death throughout much of the world [89,90]. It is therefore little wonder that cancer therapy has a history, and present, intertwined with cutting edge scientific and technological development.

The use of radiation for cancer therapy has come a long way since the first experimental treatments of 1896 [91,92]. Now, radiation therapy, or radiotherapy, is one of

the principal treatment modalities, alongside chemotherapy and surgical resection, employed by medical professionals[93]; with over a fifth of cancer patients in England[94], and up to half in the USA[95] receiving radiation therapy in some form. Most commonly this takes the form of external beam therapy, in which a beam of ionising radiation is used to attack a targeted tumour[42,93]. Typically a megavoltage photon beam, or occasionally an electron beam, generated by linear particle accelerator is used[42,96].

4.1.1 Biological Impact

On passage through biological tissue, ionising radiation imparts energy as it slows down, damaging the DNA within cells; with sufficient damage leading to cell death[42,93]. While this is the motivation as far as cancerous tissue is concerned, radiation does not discriminate between healthy and tumour cells. A photon beam imparts energy throughout its whole path, though this does not take place uniformly[42]. Different particles have their own characteristic dosage distribution patterns, several of which are illustrated in Figure 4.1.

Not all tissues respond the same to this radiation, and sensitivity largely derives from the speed at which cells divide and repair themselves, along with the specific radiation and dosage level used[42,93]. Some tissues are referred to as late reacting, able to an extent to repair themselves to compensate for damage at low dosages; making them less susceptible to damage when the same combined dose is suitably spread out. Exploiting that many forms of cancer are conversely fast to divide and poor at repair, modern treatment regimes are commonly broken into several smaller instalments, called fractions, delivered in successive instalments over several weeks[42,93,95]. Unfortunately, some cancers are late reacting, limiting effectiveness in such cases[42].

4.1.2 Treatment Planning

Avoiding collateral damage to healthy tissues is the principal limiting factor in radiotherapy, and many developments revolve around this balancing act[91,92,93]. Imaging plays a crucial role in modern high precision radiotherapy, with advanced conformal techniques naturally requiring an intimate knowledge of patient and tumour in order to fine-tune delivery[97]. For this reason, many modern clinical linear accelerators now possess integrated x-ray CT (computed tomography) imaging or radiography capabilities[95,97]. Heralded as a major step forwards, the introduction of

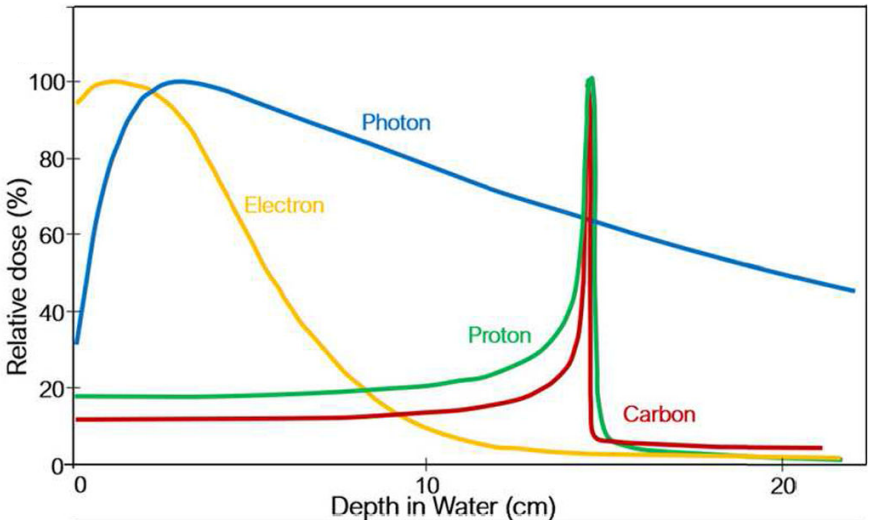


Figure 4.1: A representation of the relative dose depth distributions in water for various particle delivery methods used, or proposed for use, in clinical radiotherapy treatment. Based on particle beams of photons at $\sim$6 MV, electrons at $\sim$18 MeV, protons at $\sim$145 MeV, and carbon ions at $\sim$300 MeV/u. Reproduced from [100].

intensity-modulated radiation therapy in the 1990s allows tailoring beam to closely match complex targets[95,98]. Volumetric methods such as stereotactic radiotherapy and its variants leverage beams from different angles, overlapping at the target, to reduce exposure of any one area of healthy tissue[95,99], while recent advancements in 4D imaging and high precision regimes can even account for the motion of internal organs[91,95].

4.2 Hadron Therapy

4.2.1 Proton Therapy

Recent decades have seen considerable interest in the potential of hadron based therapy. In particular, it would be hard to review developments in cancer care without mention of the enormous growth of proton therapy. Though it has seen increasing adoption in the last few years, the case for proton therapy was proposed as far back as 1945[101], with the first patients treated in the 1950s[102] and clinical facilities since 1990[92].

As illustrated in Figure 4.1, photons impart the largest dose close to the surface, with a gradual fall off as they travel deeper. Electrons have similar behaviour, but

with a sharper fall off making them preferable for tumours within a few centimetres of the surface[42]. In contrast, protons impart more energy at lower speeds, and lose velocity as they travel within matter. This results in focusing the largest dose deposition into a narrow peak, known as the Bragg peak, near the end of their path. Exploiting this behaviour, protons can be used to target a specific region with comparatively little or no exposure for other tissues. Proton therapy is therefore ideally suited for tumours near sensitive organs or for young patients, where sparing other tissues is paramount[92,93,98,103]. Overall approaches such as stereotactic radiotherapy[104] and image guided therapy[105] are equally applicable; though the differences in dose deposition and scattering render many specific techniques and algorithms incompatible[96,105].

4.2.2 Practice

Modern proton therapy is not, however, suitable or advantageous in all cases, and not a straight replacement for regular photon radiotherapy[96,98]. Clinical proton beam production is a costly enterprise requiring expensive specialist equipment, making facilities a considerable investment and effective treatment costs high. Proton therapy is therefore typically used sparingly, reserved for when it is proven advantageous compared to other available approaches. However, high costs and the limitations of laboratory apparatus also leads to limited clinical assessments with which to make an informed decision on if and when proton therapy is worthwhile[91,96,103].

Clinical proton beam facilities typically employ a common particle accelerator to provide particles to several treatment rooms[1,96]. Many such accelerators take the form of a synchrotron, while others use isochronous cyclotrons, and are commonly capable of achieving maximum energies around 230-250 MeV[106]. More recently, single-room systems have been demonstrated utilising compact synchrocyclotrons[107].

4.2.3 Other Heavy Ions

Exploration into the potential of heavy ions has not been limited to protons, though none have seen the same level of adoption so far. Carbon ions have shown significant promise, potentially more so, for their similar despoitatiation properties as protons, but are handicapped by the exceptional equipment costs involved[91,95]. In a different direction, the use of neutrons for radioresistant cancers has been investigated, but suffers from unresolved issues involving damage to other tissues[91].

4.2.4 Treatment Planning

Just as with photon radiotherapy, treatment planning is crucial. Unlocking the full potential of protons' dosimetry properties relies on being able to align the Bragg peak with a high degree of precision; all the more so given proton therapy is frequently employed when vulnerable tissue is nearby. Accurately calibrating proton ranges relies on a detailed knowledge of the relative stopping power, or RSP, of the materials it will traverse; where the RSP is the ratio of the stopping power of said material to that of water.

Current practice is to construct the RSP map using x-ray CT density measurements, which are often expressed in the form of Hounsfield Units, or HU,

$$\mathrm{HU} = \left(\frac{(\mu_{mat} - \mu_{wat})}{(\mu_{wat})} \right) \times 1000\,, \tag{4.2.1}$$

where μ_{wat} and μ_{mat} are the measured linear attenuation coefficients of water and a given respectively[108]. The linear attenuation coefficient describes the fraction of photons attenuated over a given distance, and is usually expressed in cm^{-1}[109]. Standard measurements are often given per unit density for a given material, such as $\mu_{wat}/\rho = 7.072 \times 10^{-2}\,\mathrm{cm}^2\mathrm{g}^{-1}$ for 1 MeV photons in water[110].

Unfortunately, differences in how x-rays and protons interact with matter introduces complications to this approach. Deriving RSP from Hounsfield Unit measurements leverages that both proton stopping power and photon attenuation coefficients are each approximately, though not exactly, proportional to electron density in order to find the relationship between them[108]. Reliance on these approximate relationships gives rise to conversion errors in the range of 2-5%[97], though recent developments suggest this can be further reduced to 0.1-2.1% through measurements made at duel energies[111].

4.3 Proton Computed Tomography

Envisioned as a method of imaging around the same time as the now ubiquitous x-ray CT[112], proton computed tomography, or pCT*, offers a potential way to circumvent the issue through measuring proton RSP directly, removing conversion uncertainties by using the same particle for both planning and treatment[113]. But while pCT has seen renewed interest with the expansion of proton therapy, protons

*In some literature pCT is used to denote particle computed tomography, encompassing the use of other particles besides protons.

behaviour in matter presents its own challenges.

4.3.1 The Need for Proton Paths

Determining RSP using pCT depends on knowledge of protons trajectories. For a given proton i, the line integral of the RSP is related to the energy loss using

$$\mathrm{WEPL}_i \equiv \int_{\Gamma_i} \mathrm{RSP}(x)\mathrm{d}x \approx \int_{E_i^{out}}^{E_i^{in}} \frac{\mathrm{d}E}{S_{water}(E)} \tag{4.3.1}$$

where $\Gamma_i \subset \mathbb{R}^3$ is the proton path, RSP(x) is the stopping power relative to water at position $x \in R^3$, E_i^{in} and E_i^{out} are the entrance and exit proton energies, and $S_{water}(E)$ is the stopping power of water for energy E. This integral is the Water Equivalent Path Length (WEPL). Starting from this equation, the pCT reconstruction problem can be mapped to that of reconstructing each individual protons path, combined with the calculation of WEPL (through the right side of the equation), to recover the RSP map. The better the determination of the proton trajectories, the better the RSP calculation will be.

4.3.2 Transport Through Matter

In order to achieve spatial resolution comparable to typical medical x-ray CT scanners, proton trajectories need to be measured individually[1,114], and unlike in x-ray CT cannot be adequately approximated using straight paths[106]. Protons experience significant small angle scattering while passing through a medium, resulting in non-trivial curved paths. Most interactions occur from multiple coulomb scattering from medium nuclei, more than 10^6 per cm[115], though strong nuclear interactions also contribute in a significant number of proton trajectories[106,116].

At 200 MeV, nuclear interactions account for less than 1% compared to ionization interactions, and in our scope can largely be treated as a correction to the electromagnetic processes[115]. The resulting net scattering and displacement distributions are approximately Gaussian, with hard scattering interactions contributing non-Gaussian tails[116].

For a given proton travelling through a medium of radiation length X_0, the root-mean-square of the Gaussian core of several displacement distributions, projected into a plane, can be approximately given as

$$\theta_{plane}^{rms} = \theta^0 \ , \quad \psi_{plane}^{rms} = \frac{1}{\sqrt{3}}\theta^0 \ , \quad y_{plane}^{rms} = \frac{x}{\sqrt{3}}\theta^0 \ , \quad s_{plane}^{rms} = \frac{1}{4\sqrt{3}}\theta^0$$

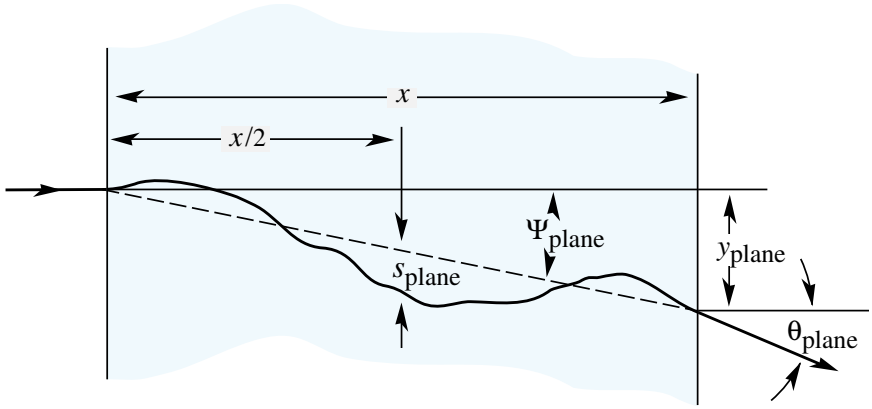


Figure 4.2: An example of a particle trajectory subject to multiple coulomb scattering, with various properties used to describe said trajectories labelled. The particles trajectory is aligned in the plane of the page. Reproduced from [116].

where

$$\theta^0 = \frac{13.6\,\text{MeV}}{\beta cp}\sqrt{\frac{x}{X_0}}\left[1 + 0.38\ln\left(\sqrt{\frac{xz^2}{X_0\beta^2}}\right)\right], \tag{4.3.2}$$

in which x denotes depth travelled parallel to $\hat{x}_{plane}$, and p and βc the particles momentum and speed respectively[116]. θ_{plane} denotes the angular displacement of direction of travel, ψ_{plane} the angular displacement in position, y_{plane} the displacement perpendicular to $\hat{x}_{plane}$, and s_{plane} the displacement from the equivalent straight line path at half way in $\hat{x}_{plane}$; these quantities are illustrated in Figure 4.2.

From this we can see that proton paths are substantially dependent on the material traversed, making them sensitive to anatomical variations within a patient; reinforcing the importance of the RSP map. This has the added effect of making photon safety margin practices unsuitable for working with protons[96]. It also shows that scattering can be reduced by using higher energy protons. However doing so potentially comes at the cost of reduced WEPL resolution, so an appropriate balance needs to be struck[106].

4.3.3 The Most Likely Path Formalism

The need for individual particle tracks excludes direct reuse of many well-developed image reconstruction methods developed in x-ray CT[106,117]. Iterative algebraic methods, such as the algebraic reconstruction technique (ART), have been proposed as plausible pCT image reconstruction methods[106,118], but the computational cost

of these algorithms is considerably high. More efficient techniques are direct reconstruction methods, often following on from x-ray CT methods, whose development is an active area of research, as discussed in [119].

While scattering remains an inherently probabilistic process, precluding the exact prediction of any single track, the Most Likely Path formalism, MLP, is well established as the most statistically precise method to account for MCS processes[120,121,122]. Since its introduction in 1994[123], the MLP formalism as presented in [120] has undergone various refinements for use in different application scenarios[122,124,125,126,127].

In addition to the entry and exit positions of the beam, the MLP algorithm utilises the angle between the direction of travel and the perpendicular to the phantom surface to significantly improve the prediction[123]. Though it has been shown to significantly reduce spatial resolution, excluding entry measurements would simplify the challenge of apparatus design and achieving an adequate resolution may still be possible[1,128].

For a proton beam located at the origin and directed along the z direction, incident into a medium, at any given depth along z a proton's path can be characterised by the two coordinates x and y and the two angles θ and ϕ relative to the z-axis. Proton scattering can be considered independent along the x and y axis and the MLP can be expressed independently for the two 2D parameter vectors $\mathbf{x} = (x, \theta)$ and $\mathbf{y} = (y, \phi)$.

Considering $\mathbf{x}$ for example, from [120] the MLP of protons in a homogeneous medium can be expressed, in a Gaussian approximation of the generalised Fermi-Eyeges theory of Multiple Coulomb Scattering, as

$$\mathbf{x}_{\text{MLP}}(z) = (\Sigma_1^{-1} + R_1^T \Sigma_2^{-1} R_1)^{-1} (\Sigma_1^{-1} R_0 \, \mathbf{x}_{in} + R_1^T \Sigma_2^{-1} \, \mathbf{x}_{out}), \tag{4.3.3}$$

where $\mathbf{x}_{in}$ and $\mathbf{x}_{out}$ are the relevant entry and exit coordinates in the two 2D parameter vectors as mentioned above, R_0 and R_1 are the change of basis for small-angle rotation matrices

$$R_0 = \begin{pmatrix} 1 & z - z_{in} \\ 0 & 1 \end{pmatrix} \quad , \quad R_1 = \begin{pmatrix} 1 & z_{out} - z \\ 0 & 1 \end{pmatrix} \tag{4.3.4}$$

and Σ_1 and Σ_2 are scattering matrices describing the variances and covariances of x

and θ between z_{in} and z, for Σ_1, or z and z_{out}, for Σ_2,

$$\Sigma_1 = \begin{pmatrix} \sigma^2_{x_1} & \sigma^2_{x_1\theta_1} \\ \sigma^2_{x_1\theta_1} & \sigma^2_{\theta_1} \end{pmatrix} \quad , \quad \Sigma_2 = \begin{pmatrix} \sigma^2_{x_2} & \sigma^2_{x_2\theta_2} \\ \sigma^2_{x_2\theta_2} & \sigma^2_{\theta_2} \end{pmatrix} . \tag{4.3.5}$$

The above components, called *scattering moments*, are given for Σ_1 by the integrals

$$\sigma^2_{x_1} = E_0^2 \left(1 + 0.038 \ln \frac{z - z_{in}}{X_0}\right)^2 \int_{z_{in}}^{z} \frac{(z-u)^2}{\beta^2(u)p^2(u)} \frac{\mathrm{d}u}{X_0} \tag{4.3.6}$$

$$\sigma^2_{\theta_1} = E_0^2 \left(1 + 0.038 \ln \frac{z - z_{in}}{X_0}\right)^2 \int_{z_{in}}^{z} \frac{1}{\beta^2(u)p^2(u)} \frac{\mathrm{d}u}{X_0} \tag{4.3.7}$$

$$\sigma^2_{x_1\theta_1} = E_0^2 \left(1 + 0.038 \ln \frac{z - z_{in}}{X_0}\right)^2 \int_{z_{in}}^{z} \frac{(z-u)}{\beta^2(u)p^2(u)} \frac{\mathrm{d}u}{X_0} , \tag{4.3.8}$$

where u is z between z_{in} and the the fixed value of z for which we are calculating $\mathbf{x}$. The equivalent scattering moments for Σ_2 are found by replacing z_{in} with z and z with z_{out} in the equations above. $\mathbf{y}_{\mathrm{MLP}}(z)$ follows identically, with variables corresponding to $\mathbf{x}_{in}$ and $\mathbf{x}_{out}$ replaced by $\mathbf{y}_{in}$ and $\mathbf{y}_{out}$ as necessary.

However, nuclear interactions still have a non negligible role in a significant number of proton trajectories[106]. Recommended practice is therefore to reduce the events influenced by nuclear interactions or large angle MCS through a 3σ cut on both the difference in energy and the difference in the direction of travel angle between entry and exit[120]. Unfortunately, this results in a reduction of the protons available for the pCT image reconstruction and in an increase of the time needed to compute the relative stopping power map for proton therapy treatment planning.

4.3.4 Computational Cost

The need for measurement and prediction of individual particle tracks brings with it a considerable computational burden, one which was beyond the scope of viable computing technology only a few decades ago[1,2]. Exploring ways to minimise this issue is an area of ongoing research. In [129] and [130], a computer optimised implementation for use on a GPU is outlined, achieving a notable speed up at high proton density. Others have taken the approach of devising alternative, more computationally efficient, approximations in place of the MLP. The use of cubic splines in [118], further developed in [122], has been shown to be adequate for pre treatment verification purposes[131]. In [126], polynomial approximations lead to a reduction in the number of floating point operations in calculations, while [132] fitted cubic Bézier

curve tracks with a Molière maximum likelihood method, demonstrating advantages particularly for longer proton paths.

Machine learning has shown great potential for estimating complex processes efficiently, and the implementation discussed in Chapters 5 to 7 has as already indicated been published as [3]. Subsequently, in [133] a wider study by a different group into machine learning for pCT proton trajectories has since been published. Examining both feed forward neural networks and a boosted decision tree method in a more realistic setup, their research similarly showed benefits to computational time and accuracy. Further discussion is given in Section 7.1.

Beyond direct prediction of proton paths, machine learning has been applied to pCT in other ways. [134] introduces a range verification measure to improve pCT quality control, utilising a neural network to predict Bragg peak depth. Others have looked to convolutional neural network models to improve the quality of CT images more directly. [135] looked to identify incorrectly reconstructed and secondary production tracks, while [136] used a Bayesian CNN for image correction with accompanying uncertainty predictions. An alternative approach in [137] introduced a machine learning-driven image denoising method designed around preserving WEPL values that may be distorted during traditional image correction.

4.3.5 Equipment

Equipment and running costs represent a significant hurdle to the clinical implementation of pCT. In order to share an already necessary particle accelerator, pCT is generally envisioned as an integrated part of proton therapy installations. Unlike for therapy itself, scanning requires protons to completely pass through a target and out the other side, and therefore higher energies for the same regions of interest are required[1]. Clinical accelerators for therapy use can typically achieve energies in the 230 to 250 MeV range[1]. This does introduce some limitations; it is typically adequate for the head or chest region with a suitably positioned patient, but not, for example, for scanning adult hips at all angles[1].

With the significant cost of beam time, necessitating access for both planning and therapy itself is a difficult proposition. The ideal achievement would be for on-site treatment planning immediately before treatment, with the patient remaining in the room throughout[1]. This would not only reduce overall proton therapy costs, but removing the need for a separate treatment planning session also improves the patient experience[1]. While a naturally attractive proposition, and potentially in the realm of possibility[1], a currently more realistic scenario maybe optimising calibration of

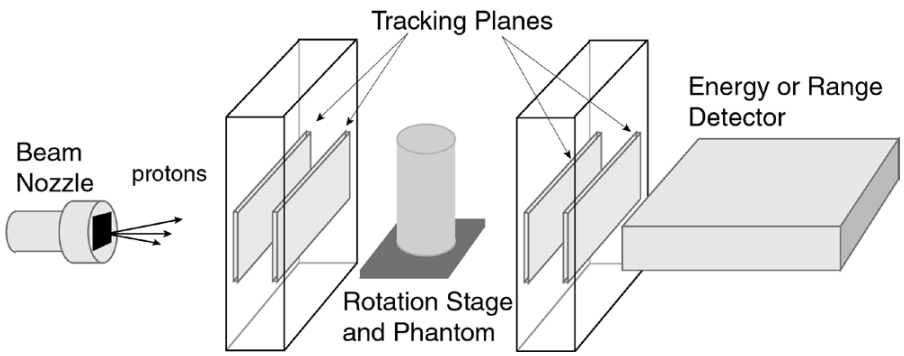


Figure 4.3: An illustration outlining the design of a typical pCT scanner. Reproduced from [106].

an existing treatment plan for a patient on the day[1,138], or reserving its use for only special cases[1].

As it stands, various pCT systems have been proposed, and several prototype pCT systems now exist. However, most current prototypes have been built as research instruments, and are inappropriate for clinical use in their current form[1]. A review of current pCT detector design and prototypes can be found in [1].

At its core, a pCT scanner is a particle detector. It is no surprise therefore that many of the technologies employed are those developed and used in elementary particle physics research detectors[1], even directly utilising components developed for LHC experiments[139].

An outline of a typical pCT scanner proposal can be found in Figure 4.3[106]. To build up a full image, the centrally-placed target, or the apparatus as a whole, must be rotated in order to perform scans from different angles[106]. Tracking detectors located before and after the target measure entry and exit position, and in order to ascertain particles' direction of travel, double tracking planes are required[1]. Designs commonly employ silicon-strip detectors for this purpose[140,141], while scintillating fibre technology[142] and monolithic active pixel sensors have also been used[139], and micro pattern gas detectors may be suitable[1].

As we saw in Equation 4.3.1, calculating the WEPL relies on the energy loss in transit. As knowledge of the accelerator beam production can be used to deduce the incoming kinetic energy, energy measurement is only necessary after the target. Some designs achieve this using calorimeters[140,142], which stop the particle and produce a signal related to the energy imparted. Others take a different approach, measuring the range penetrated in some medium to infer the WEPL[141], while using time of flight measurements has also been proposed[143].

Chapter 5

Aims and Method

5.1 Aims

Neural networks are capable of approximating complex tasks in a computationally efficient manner, aligning with the desire for a faster reconstruction of proton tracks for pCT seen from existing studies[118,122,126]. Our aim was therefore, in the context of pCT, to develop a proof-of-concept approach for the estimation of the proton paths based on Machine Learning, through utilisation of a Deep Neural Network, in order to explore the potential of such an approach to match the performance of MLP formalism in a shorter execution time. This model became the Proton Path Neural Network, or PPNN.

Equally, neural networks are not bound to following only predetermined physical models. Predicting proton paths is limited by the statistical nature of scattering, and as discussed the MLP formalism represents the most accurate estimate for Multiple Coulomb Scattering. A neural network however has the potential for increased accuracy from a capacity to account for contributions beyond Multiple Coulomb Scattering, and the better the determination of the proton trajectories, the better the RSP calculation will be. Additionally, removing tracks with a heavy non Multiple Coulomb Scattering influence is the motivator for applying 3σ data cuts, thus reducing the need for these cuts may allow for more usable data.

5.2 Monte Carlo Simulation

Both model training and analysis were conducted using Monte Carlo simulation data generated using GATE v9.0[144], a framework built upon the widely used Geant4 10.6 Monte Carlo simulation toolkit[145]. Simulations incorporating only

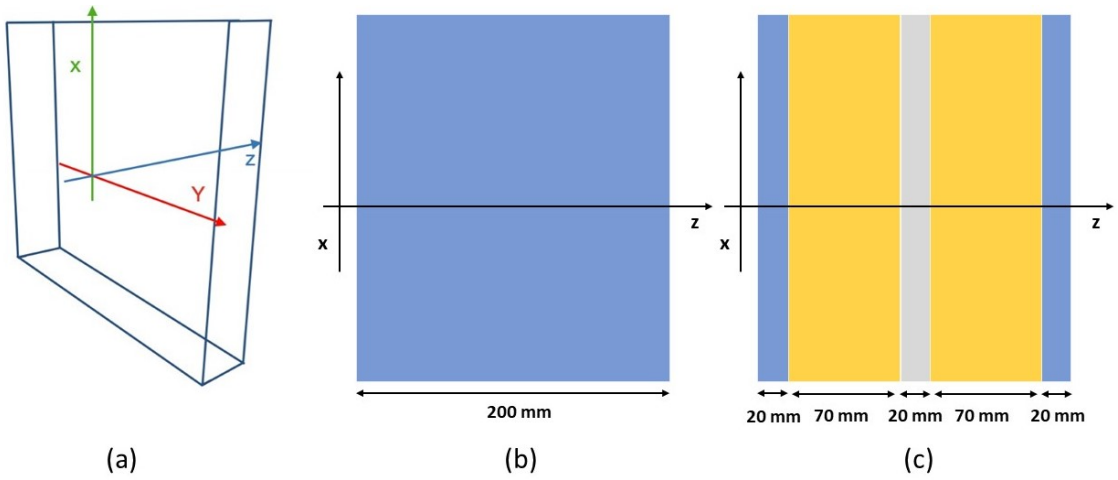


Figure 5.1: Illustration of the Monte Carlo geometry used in this study. 3D representation of the phantom space (a) and 2D projection on the x-z plane for the water (b) and inhomogeneous phantom (c). Trajectories are only scored and monitored within the phantom volume itself. Note that for convenience we redefine our coordinate axis such that the initial point of each trajectory is located at the origin. Reproduced from [3].

electromagnetic processes were performed using the *emstandard* physics list. Nuclear interactions, among a full regime of physics processes, were modelled using the $QGSP\_BIC$ physics list. Details can be found in [146] and [147] respectively. In the discussion of the results, the choice of physics environment is indicated for each simulation.

Our principal model consists of a homogeneous sheet of water centred on the origin of a standard x-y-z coordinate system with a side length of 20 cm in the z-axis direction and arbitrarily large extents in x and y. Phantoms of this kind are widely used as a baseline in existing literature, such as in [120], [126] and [127]. Monoenergetic protons are simulated through the phantom, originating at the central point of the phantom's $z = -10$ cm face, such that their initial direction of travel are orientated inwards and perpendicular to the face and parallel to the positive z-axis direction. For convenience in the following we redefine our coordinate axis such that the initial point of any trajectory is located at the origin, with particles initialised at a depth of 0 cm and extending in range to a depth of 20 cm. This arrangement is illustrated in Figure 5.1(a) and 5.1(b). While the situation as we are modelling it is not a perfect recreation of the real world environment, it is sufficient for assessing the capability of a neural network at path prediction in general terms.

The procedure as stated was repeated using an inhomogeneous phantom comprising 2 cm of water, 7 cm of skull, 2 cm of cortical-bone, 7 cm of skull, and 2 cm of

water. For the purposes of this simulation, cortical-bone was defined using material data found in [148]. This composition was chosen based on a similar inhomogeneous phantom used in [127]. While it is rare that such a large concentration of high density material would be encountered in a clinical setting[127], by using a more extreme inhomogeneous phantom any performance impact will be more distinctive in comparison.

5.3 Datasets

Datasets were produced using the aforementioned Monte Carlo simulations as described in Section 5.2. Each dataset consisted of generating an initial 10^6 simulated events, however only trajectories which traversed the full phantom depth were retained, reducing the number of events ultimately used. Initial datasets for the homogenous phantom were generated using a beam energy of 200MeV, and in order to examine the impact of scattering due to nuclear interactions, separate datasets were produced using the *emstandard* or $QGSP\_BIC$ physics lists.

Due to the increased stopping power of the inhomogeneous phantom, to ensure that a large fraction of impinging protons successfully traverse that phantom's full length, datasets for the inhomogeneous phantoms were generated using a beam energy of 230 MeV. Additional datasets for the homogeneous phantom were also generated at 230 MeV. All 230 MeV simulations were carried out under the $QGSP\_BIC$ physics list. For the inhomogeneous phantom at 230 MeV, this typically led to data sets in excess of $700,000$ events. Principally, examination of behaviour with the homogeneous phantom uses 200 MeV proton datasets, with the homogenous phantom 230 MeV protons datasets used for comparison with the inhomogeneous phantom.

Each proton trajectory was quantified as a series of spatial coordinates evenly distributed at 0.1 cm intervals, including both phantom faces. This granularity was chosen to match that used in [122], [126] and [127]. A total of 201 coordinate points represent a complete path through the phantom, consisting of 603 variables. The angle on entry and exit to the z-axis direction in both x and y planes was also recorded.

5.4 The Proton Path Neural Network

The Proton Path Neural Network, or PPNN, is a fully connected neural network based model designed to predict a proton trajectory in the form of a series of spacial

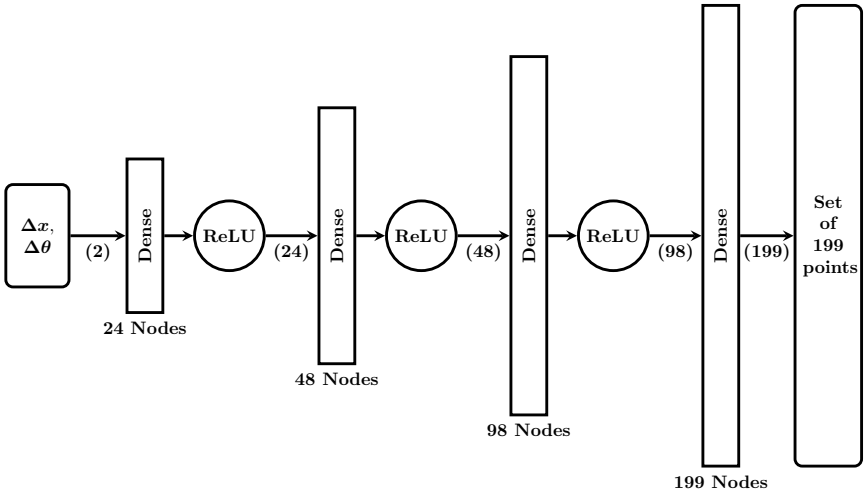


Figure 5.2: PPNN architecture. The Proton Path Neural Network PPNN consists of four fully connected layers with 24, 48, 96, 199 nodes and a Relu activation function after each of the first three layers. The current number of variables present at various points is additionally indicated in brackets. A separate instance of same architecture was used for the **x** and **y** planes. Reproduced from [3].

points at 0.1 cm intervals, matching the representation described in Section 5.3, using variables similar to those employed by MLP calculations. As the z depth coordinates are therefore a fixed set of values shared by all trajectories, for predicting a track only the x and y variables need be considered. Similarly, the initial and final points of each trajectory are known for each track and so likewise neglected. Thus a track prediction consists of two sets of 199 points each, for a total of 398 variables per track.

As with the MLP, trajectories along the x and y directions are reconstructed independently by separate instances of the same network. The input features of the network are quantities which can be recorded by a modern pCT scanning apparatus; $\Delta x = (x_{out} - x_{in})$ and $\Delta\theta = (\theta_{out} - \theta_{in})$ in the x direction and equivalently $\Delta y = (y_{out} - y_{in})$, $\Delta\phi = (\phi_{out} - \phi_{in})$ along y. This data is passed through 4 fully connected (or dense) layers of 24, 48, 96 and 199 nodes respectively. As activation functions, we employed a Rectified Linear Unit (ReLU) after each of the first 3 layers. A representation of the network architecture is presented in Figure 5.2. PPNN is written in python using the PyTorch (pytorch.org) framework.

Training and validation of the model was performed using a dataset of more than $1,600,000$ trajectories ($1,400,000$ for the inhomogeneous phantom), $800,000$ ($700,000$) along each direction, as described in Section 5.3. Two separate instances

of the network were trained using datasets corresponding to different simulations; the first used 200 MeV protons and the homogeneous phantom, the second 230 MeV protons and the inhomogeneous phantom. Both employed the full *QGSP_BIC* physics list. Unless otherwise stated, the relevant model was used when reconstructing datasets generated with the corresponding proton energy.

80% of the tracks are used for the training and the remaining 20% reserved for validation. Optimisation of the network weights is performed using the Adam algorithm[56] with a learning rate fixed at 10^{-5}. For the loss, the Mean Squared Error (MSE) is used,

$$\text{MSE} = \frac{1}{M}\sum_{m}^{M}\frac{1}{N}\sum_{n}^{N}(u_{mn} - \hat{u}_{mn})^2, \tag{5.4.1}$$

where M is the number of samples, $N = 199$ is the number of points in each proton path, u the predicted path and $\hat{u}$ the true trajectory. At a batch size of 32 samples per batch, one epoch (one cycle through the full training dataset) running on Tesla K80 GPU requires approximately 80 seconds on a Standard NC6 Microsoft Azure machine.

The loss history of the model trained on 200 MeV protons, on the homogeneous phantom, with the *QGSP_BIC* physics list, can be seen in Figure 5.3, in which after around 400 epochs the loss flattens both for the training and validation datasets with the ratio between the two histories almost constant; suggesting that the network is not overfitting to the examples present in the training dataset. Ultimately this model was trained for 1000 epochs.

5.5 Measurements and Analysis

5.5.1 Most Likely Path Implementation

For comparisons to the MLP formalism, we implemented the highly optimized version of MLP presented in [130], in which 90% of the MLP is precalculated and the number of operations required is minimised. We ported the code in python using the vectorisation capabilities of the NumPy (numpy.org) library to parallelize the execution on the number of protons. Assuming a homogeneous phantom composed of water, we use $X_0 = 36.1\,\text{cm}$ for the radiation length of the material and $E_0 = 13.6$ MeV. The momentum velocity ratio $1/\beta^2(u)p^2(u)$ is approximated with a fifth-order polynomial following [120]. This quantity is specific to the proton energy used; implementation for other energies requires its recalculation for accurate performance.

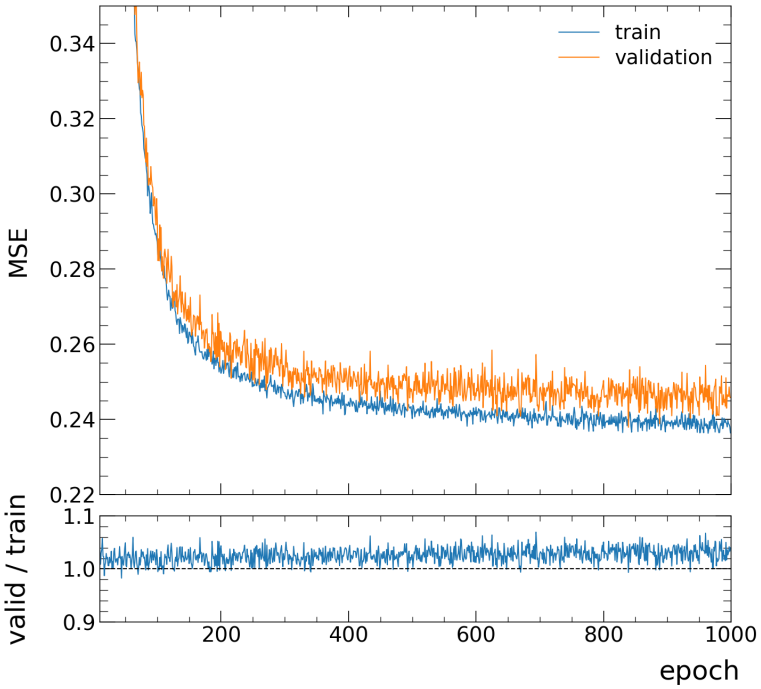


Figure 5.3: Loss history during network training at each epoch, for both the training and validation. This corresponds to model trained using 200 MeV protons, with the homogeneous phantom, using the $QGSP\_BIC$ physics list. Reproduced from [3].

For protons at 200 MeV, the coefficient values given in [120] were used. For protons at 230 MeV, the coefficient values were calculated following the method outlined in [120] for the 200 MeV values. Monoenergetic protons initially at the required energy were incident on a simulated 20 cm deep water sample. The fifth-order polynomial was fitted to distribution of the mean value of $1/\beta^2(u)p^2(u)$ recorded at 5 mm intervals throughout. As with the PPNN model, unless otherwise stated the corresponding momentum velocity coefficients were used when reconstructing each dataset.

5.5.2 Root Mean Squared Error

"The (Square) Root of the Mean Squared Error, or RMSE, is commonly adopted in literature evaluating the performance of the MLP reconstruction procedure, and was adopted here as the principal means to compare performance between PPNN

and the MLP formalism. The RMSE is calculated as

$$\text{RMSE} = \sqrt{\text{MSE}} \tag{5.5.1}$$

where the MSE is as given in Equation 5.4.1. To avoid overfitting, all analysis procedures were carried out on datasets generated independently from those used in the PPNN training and validation process, or the MLP momentum velocity ratio calibrations.

5.5.3 Execution Time

To measure execution time, the two algorithms were run on all trajectories of a test dataset. This procedure was repeated 10 times, with unique batch combinations in each instance, to produce a mean value. Both PPNN and the MLP formalism were executed on the CPU of a Standard NC6 Microsoft Azure machine.

Chapter 6

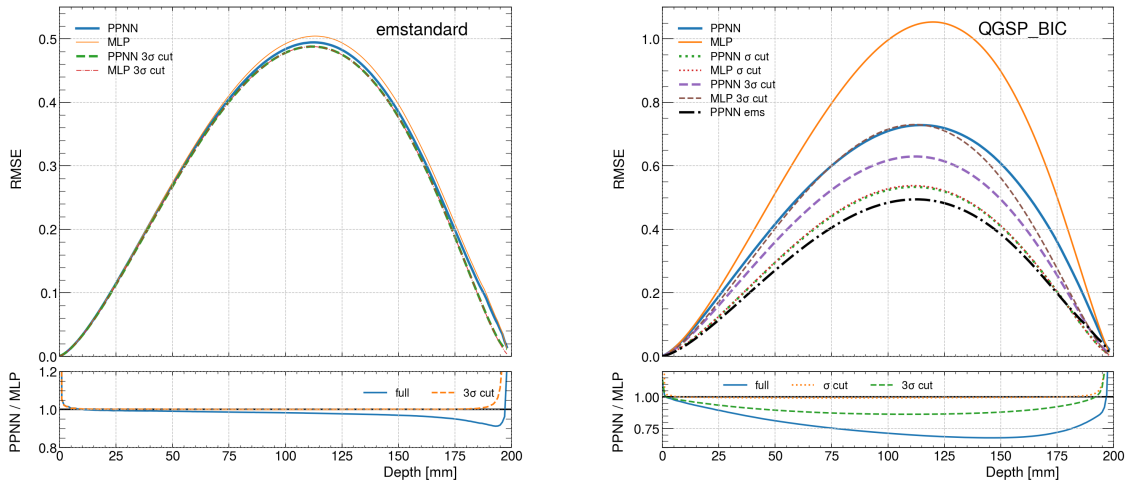
Results and Analysis

6.1 Homogenous Phantom

6.1.1 Root Mean Squared Error

For the homogeneous phantom, the RMSE for estimates of the paths, using PPNN or MLP, with the *emstandard* and 200 MeV protons dataset are shown in Figure 6.1(a). Even without the 3σ cuts suggested in [120] we can see that the difference between the two predictions is quite small. This difference disappears (the two lines corresponding to the MLP and PPNN case are barely distinguishable) upon applying said 3σ cut to the angles and energy; under which here only $\sim 1\%$ of the paths are omitted. This result clearly shows that the PPNN prediction is fully consistent with the MLP approach, indicating that the approximations inherent to the method are valid. This is crucial because anything different would represent a serious flaw in the PPNN reconstruction method.

Moreover, the difference in the PPNN prediction error with or without the cut is practically negligible, suggesting that our method can be applied to reconstruct trajectories where processes other than MCS are present. This is more evident in Figure 6.1(b) where the RMSE is evaluated for the $QGSP\_BIC$ dataset. When nuclear interactions are included the error significantly increases, but to a far lesser extent for PPNN than for MLP. Only with a 1σ cut do the performances of the two methods become comparable. Unfortunately, such a large cut entails the loss of $\sim 24\%$ of the tracks. Comparing the full interaction dataset result with that of the pure electromagnetic result, we see that with the typical 3σ cut applied to both cases the RMSE of PPNN is about 26% larger for the full interaction than for the pure MCS dataset. Though not shown, with a 2σ cut the discrepancy in performance



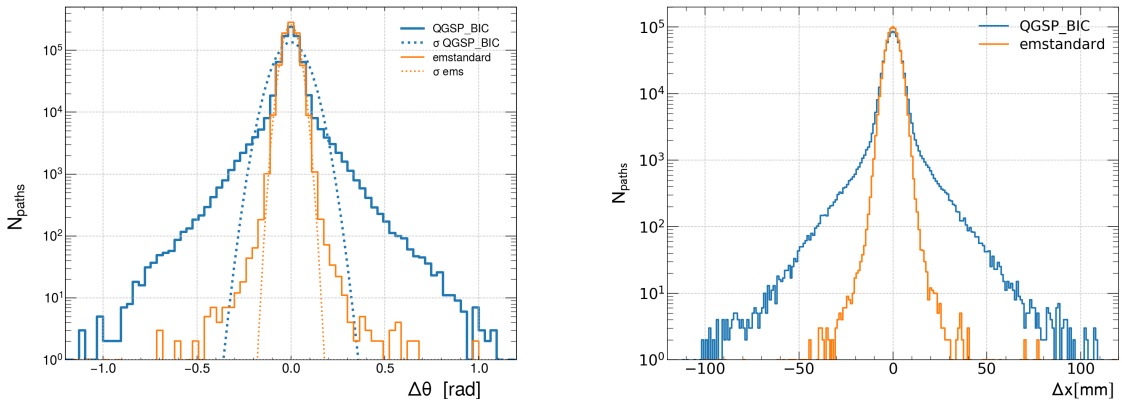
(**a**) (**b**)

Figure 6.1: Root Mean Squared Error obtained with MLP and PPNN using the (a) *emstandard* and (b) *QGSP_BIC* datasets. Solid lines are the performance on the full dataset while dotted and dashed incorporate 1σ, and 3σ cuts, performed on the energy and difference in the direction of travel angle between entering and exiting the phantom, respectively. The dashed-dotted line in (b) is the same solid PPNN result in (a) added here to have a clear picture of the increasing of the errors when including nuclear interactions. Reproduced from [3].

decreases to around 20%, which corresponds to a fraction of discarded tracks of $\sim 8\%$ from the *QGSP_BIC* dataset.

6.1.2 Relationship Between Error and Deviation

To understand the origin of this difference in performance between the two methods, Figure 6.2(a) illustrates the distribution of $\Delta\theta = (\theta_{out} - \theta_{in})$ for both *QSPG_BIC* and *emsstandard* datasets. The σ cut is applied assuming a Gaussian distribution of the signal, but from the figure a difference between the two distributions clearly emerges. For the full physics simulation the Gaussian approximation, as employed in the MLP, clearly fails to describe the distribution away from the centre. While the cuts based on a Gaussian fit are acceptable in the *emstandard* case, they exhibit a large discrepancy with data when the full range of physics processes are included. In Figure 6.2(b) we see a similar result for the distribution of lateral displacement $\Delta x = (x_{out} - x_{in})$, with the Gaussian shape of the *emstandard* distribution supplanted by an exponential decrease in the *QSPG_BIC* distribution.

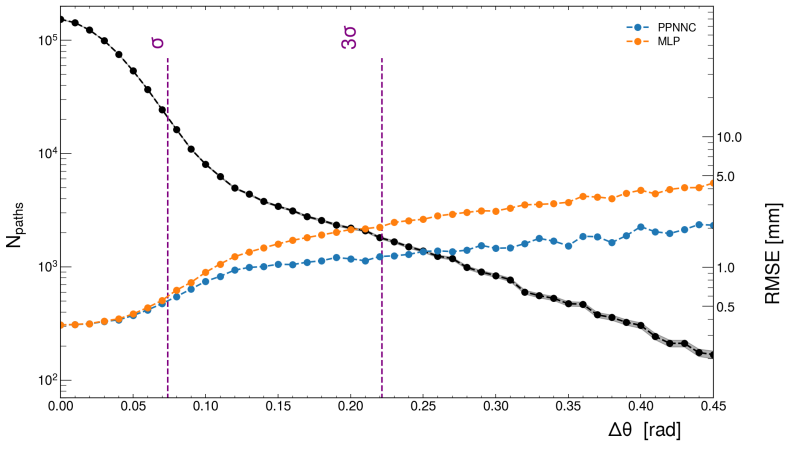


(**a**) (**b**)

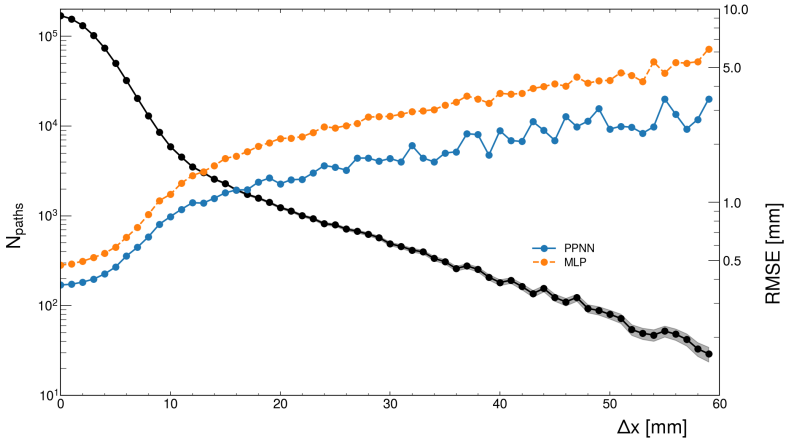
Figure 6.2: (a) Distribution of $\Delta\theta = (\theta_{out} - \theta_{in})$ angle for the two test datasets (solid lines) overlaid with the associated Gaussian using the σ values obtained from a fit of the *emstandard* data and the $QSPG\_BIC$ data (dotted lines). (b) Distribution of $\Delta x = (x_{out} - x_{in})$. In both plots it is evident that an exponential rather than a Gaussian decay provides a better fit with respect to the number of paths for the $QSPG\_BIC$ dataset. Reproduced from [3].

Given the limits of the MLP formulation shown in Figure 6.2, its worth considering how the error increases as a function of the two variables $\Delta\theta$ and Δx. As the PPNN approach has the same performance as MLP in the context of pure electromagnetic interaction, where MLP is designed to work, we will focus on the the $QSPG\_BIC$ physics dataset, which provides a more realistic representation of clinical pCT scenario.

This is presented in Figure 6.3. Here the proton paths are collected into bins of 0.1 rad and 1 mm for $\Delta\theta$ and Δx respectively, with the RMSE computed in the corresponding direction. The figure compares the error (right axis) and the number of trajectories (left axis) to show the differences in performance. Note the logarithmic scale on both right and left y axis. From Figure 6.3(a) we see that, as expected from the RMSE plot, the two lines for PPNN and MLP begin to separate at around 1σ cut at $\Delta\theta \simeq 0.075$ rad. For 35% of the tracks $\Delta\theta$ is larger than 0.075, implying that the PPNN method improves on the MLP reconstruction for an important fraction of proton paths. Notice that the same analysis must be done for the ϕ angle which would remove an analogous number of paths, resulting in a final cut of almost 50% of the tracks in order to recover the same performance. Figure 6.3(b) shows the reconstructed paths distribution broken down in terms of final displacement, Δx. Again the performance of PPNN is consistently better across the full span of the



(**a**)



(**b**)

Figure 6.3: (a) RMSE (right vertical axis, coloured lines) and number of paths (left vertical axis, black lines) as a function of $\Delta\theta$ for PPNN and MLP evaluated on the *QSPG_BIC* dataset. The shaded black area represent the statistical error. Vertical lines refer to the position of the 1 and 3 σ cut. (b) Same as (a) but as a function of Δx. The difference in performance between the two methods emerges immediately. Reproduced from [3]. Note the right vertical axis of Figure 6.3 should read 'RMSE [rad]'

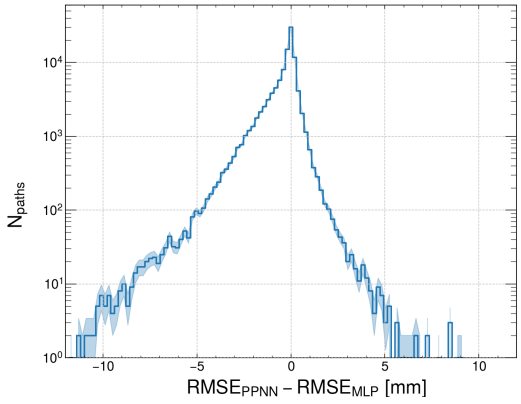


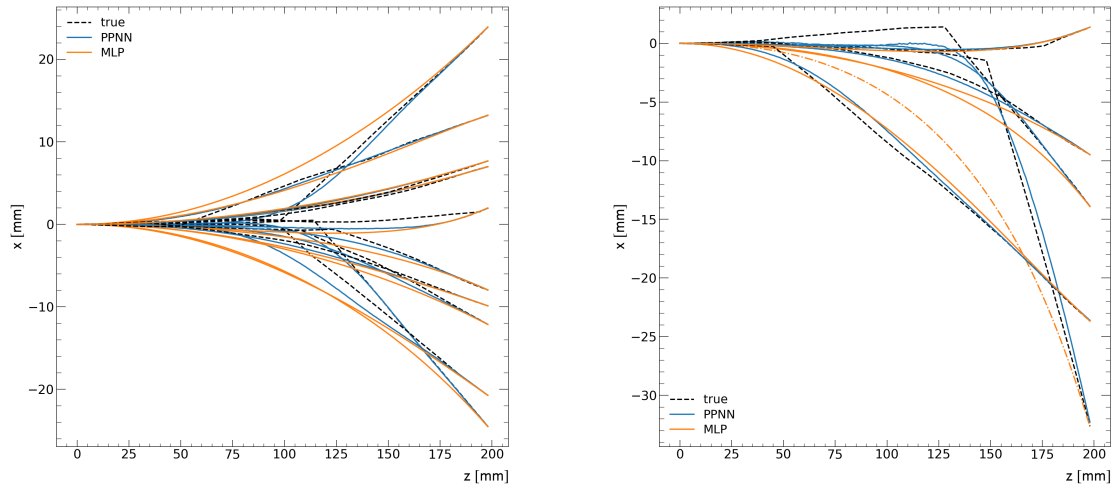
Figure 6.4: Distribution of the difference between the RMSE of PPNN and MLP for the $QSPG\_BIC$ dataset. The shaded area correspond to the statistical error. Reproduced from [3].

plot, with trajectories at large angle deviations resolved with improved precision.

6.1.3 Tracks with Differing Performance

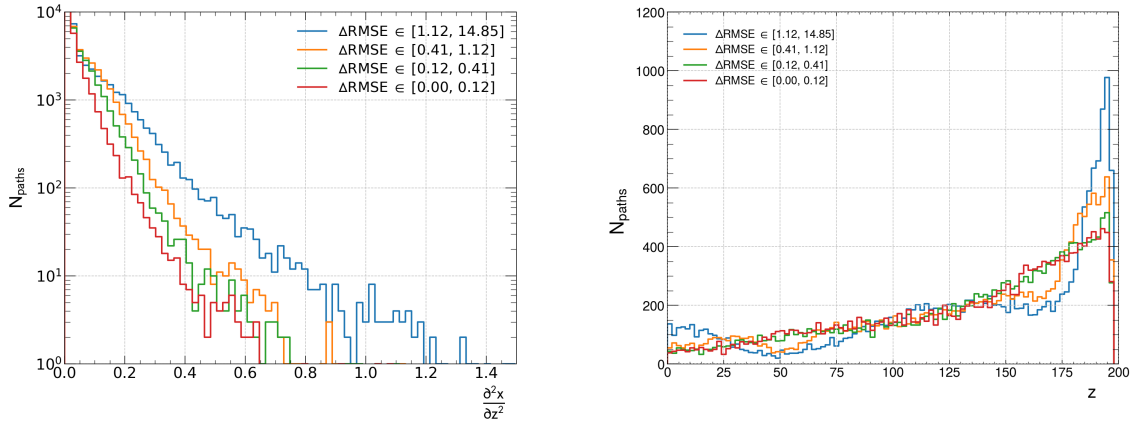
Further insight can be gained by considering tracks where the two models performance substantially differed. In order to focus on tracks with the largest difference, we will consider only tracks outside the 1σ cut in θ. Figure 6.4 presents the distributions of the difference between the RMSE for PPNN and MLP for tracks outside the aforementioned cut. Negative values of the difference correspond to tracks in which PPNN had the smallest error, while the positive side of the axis corresponds to the inverse. In the first instance we can see that the profile is exponential, while in the second the decay is noticeably steeper; confirming that PPNN has better performance at large deviations of the angle θ.

Focusing in on only the behaviour when PPNN outperforms MLP, let us consider only the set of events on the negative side of histogram. Dividing these events into 10 quantiles by $\Delta RMSE$, a selection of randomly chosen tracks, one from each quantile, are shown in Figure 6.5(a). As expected, for larger deviations from straight paths PPNN can better follow the simulated curve in the majority of such cases, growing more notable for larger $\Delta RMSE$. For Figure 6.5(b) the same dataset was divided into quartiles, with the last bin, containing tracks with the largest error difference, further divided into two subgroups. As with Figure 6.5(a), a random track was chosen from each of the five groups. Both figures further support that PPNN improved performance is due at-least in part to a better capability to reproduce the particle path in the presence of nuclear interaction, which causes greater changes in



(a) (b)

Figure 6.5: Examples of tracks for which the PPNN outperform MLP. (a) Tracks are selected at random from inside each of 10 quantiles, using the data of Figure 6.4. (b) Same as (a), but in which tracks are extracted from quartile groups; with the last quartile, which corresponds to tracks with the largest discrepancies between the two methods, divided into two. Reproduced from [3].



(a) (b)

Figure 6.6: (a) Distributions of the second derivative of the tracks in the x direction with respect to the z coordinate. Lines indicate the four quartiles of the distribution of $\Delta RMSE < 0$. (b) Distribution of the position along the z axis for the maximum of the second derivative for each path. Reproduced from [3].

the direction of the track.

To further analyse this characteristic, Figure 6.6(a) shows the distribution of the second derivative of the x component of the tracks, with respect to the z direction, again for track in which PPNN outperforms MLP, broken down into quartiles by $\Delta RMSE$. Large values of this quantity are connected with significant direction change, such as those observed in Figure 6.5. The four lines correspond to the four quartiles of the blue histogram in Figure 6.4, as introduced in Figure 6.5(b). Where PPNN exhibits the better performance, we see that the difference between the tracks reconstructed with PPNN and MLP grows with increasing values of $\frac{\partial^2 x}{\partial z^2}$: the more a trajectory differs from pure MCS scattering, the more the PPNN improves over MLP.

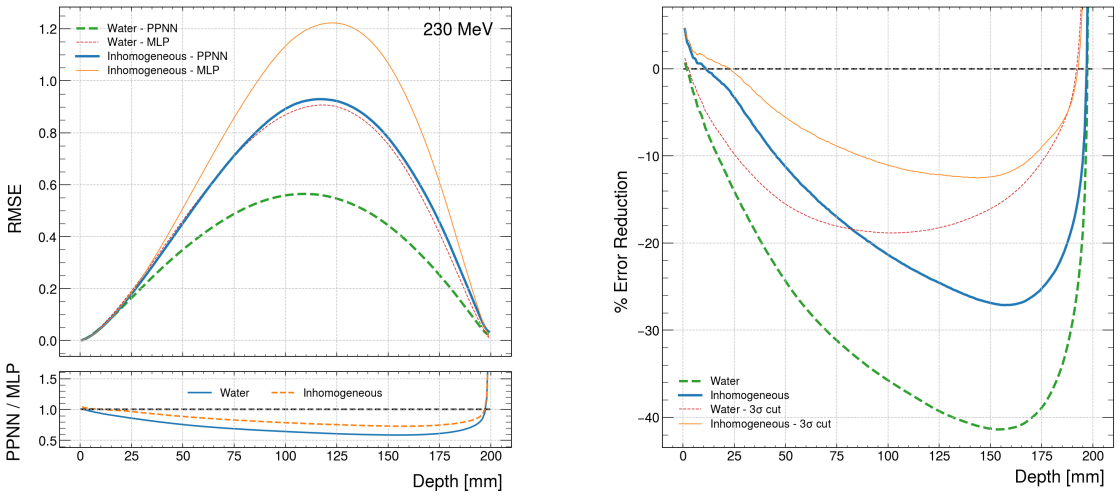
Figure 6.6(b) shows the distribution of $\max(\frac{\partial^2 x}{\partial z^2})$ as a function of z, broken down into the same quartiles by $\Delta RMSE$ as before. The distribution for the last quartile, corresponding to the largest discrepancies between the two methods, has a notably different behaviour compared to the other three lines. It exhibits significantly more events occurring at small and large z values. An example of these events can be seen in Figure 6.5(b) where we have a strong deflection at $z \approx 190$ mm. Here we can see that MLP struggles to reproduce this event while the neural network provides a superior result.

6.2 Inhomogeneous Phantom

In order to consider the performance on the inhomogeneous phantom, as explained in Section 5.3, simulations using 230 MeV protons were used, along with the corresponding models. The RMSE error for both the homogeneous and inhomogeneous phantoms, using either PPNN or MLP, is shown in Figure 6.7(a). This comparison is without cuts and using the $QGSP\_BIC$ physics environment. For the water phantom both PPNN and MLP behave similarly to the corresponding 200 MeV case. This is an important check that the higher energy implementations of the two methods are functioning correctly, particularly as this PPNN instance was trained with the inhomogeneous phantom.

Focusing on the reconstruction error for the inhomogeneous case, we can similarly observe that with PPNN the error is consistently reduced. Interestingly the error on the new phantom using PPNN is comparable with that obtained with MLP in the pure water simulation.

The improvement obtained with PPNN is more pronounced when examining the



(a) (b)

Figure 6.7: (a) Root Mean Squared Error obtained with MLP and PPNN on a water and an inhomogeneous slab phantom irradiated at 230 MeV. (b) Percentage reduction in RMSE with respect to depth by PPNN over MLP. All studies were performed with the *QGSP_BIC* physics environment. Reproduced from [3].

percentage reduction of RMSE by PPNN over MLP, as shown in Figure 6.7(b). A reduction in the error of the order of 25% can be seen around 150 mm, while on average the improvement is in excess of 10% over MLP across a significant portion of the depth. Introducing the familiar 3σ cuts decreases the error reduction in both the water and inhomogeneous cases, along with the difference in improvement between them.

6.3 Execution Time

Using the methodology outlined in Section 5.5, from 10 runs we obtain an an almost constant execution time of 0.47 ± 0.01 sec for PPNN and 7.11 ± 0.08 sec for MLP. Within the validity of this test, the PPNN method is sixteen times faster than the optimised MLP.

Chapter 7

Discussion

Although MLP represents a powerful method of estimating proton path in pCT applications, it suffers from certain limitations. The approach is designed specifically to account only for effects on the proton path connected with MCS and energy loss. This is reflected by the strategy of discarding proton trajectories with large deviation from straight paths to reduce the error. Moreover, simulation in a realistic scenario of high fluence (hundreds of millions of protons) and small spacing for the MLP (fraction of millimetre) can require more than one hour; time mostly spent reconstructing the proton paths[119].

The results presented suggest a machine learning approach such as PPNN has the potential to relieve these two problems to some degree. Figure 6.1 and Figure 6.3 indicate that by using PPNN a good approximation of the path can be obtained for a much larger number of protons than using MLP. This is important because in principle fewer protons are needed to reach the same reconstruction quality, lowering both the dose and the computation time.

The ability of the network to reconstruct tracks outside the validity of the MLP approach is intrinsically tied to the nature of deep learning. Neural networks learn 'blindly' from examples; parsing though the training dataset, by means of the back-propagation procedure for the minimisation of the loss function, the network adapts its weights to the characteristics of the events it experiences, including those that show large $\Delta\theta$ and/or Δx. While such underlying processes may be challenging to formulate into mathematical models, there are sufficient patterns for the network to refine its prediction processes. Without an assumed structure to reproduce, it is not bound to solely replicating the form of a given physical model. A tentative explanation of what the network learns may be inferred from Figure 6.6 and the analysis of the second derivative of x with respect to z. The network displays significant

improvement over MLP where the second derivative is large, especially near the end of the trajectories."†

Regarding execution speed, it is true that the time spent for reconstruction is only one of the various aspects for evaluating a pCT system for clinical routine. Moreover, our work is relevant only in the context of reconstruction methods based on the evaluation of the proton path. Nevertheless, because these methods are seen as the most promising for applicability in the clinical context and the MLP execution speed is by order of magnitudes the slowest part of the algorithm [119], the substantial improvement shown by PPNN compared with the optimised MLP can be regarded as an important feature.

A notable limitation to our findings is the artificial nature of the situation PPNN was implemented in. Using a fixed incoming angle to the phantom serves adequately for a proof of concept that a neural network based approach could handle the problem, but is not a realistic representation of intended use. For MLP, assuming the incoming protons angle and position significantly reduces eventual spatial resolution [128], so it is likely as impactful for a machine learning equivalent. Though the axis can simply be relocated to account for the incoming spacial location, reorientating to account for angle alters the phantom depth, impacting that a fixed z direction spacing was used. Equally, real scanning targets will be a range of thicknesses, and measurement of a protons position and direction of travel is not made on the surface. Not accounting for the air gap has been demonstrated to have a non-negligible impact on the accuracy of MLP predictions [125]. In existing literature, several comparable simulations, such as the principal simulations in [120] and [127], therefore utilise a fan beam placed back from the phantom surface as a particle source. For our study we decided a simpler model was adequate for examining if, in the first instance, a neural network model had the potential of achieving comparable results to the existing MLP formalism; though any future work should likely employ a more advanced setup.

While in extending the principals here to a more realistic simulation is likely possible, it does mean our results may therefore not be perfectly reflective of network designed for and operating in a real situation. Further, given their example based learning nature, neural networks such as these have inherent limitations stemming from the limits of the training data sets used. Another potential concern is that both the training and evaluation Monte Carlo data was produced using the same simulation, making the process blind to any systematic errors in the simulation.

This aside, PPNN performed admirably with the inhomogeneous phantom. The

phantom considered is certainly extreme; large volumes of a high-density material such as those in the slab phantom will rarely be encountered in clinical practice, and in this sense we do not expect the gain to be so large in a realistic situation. Nevertheless, it is encouraging that notably better results are obtained with PPNN with respect to MLP, with a reduction of the RMSE of the order of 20%. This is a more significant improvement compared to the work presented in [127] with a similar phantom, where the maximum enhancement is about 5% for simulation with the same beam energy.

Just the same, this was carried out using a single fixed material composition on which the model had been trained. A realistic target will not be a set composition that separate models can be trained against, and is unlikely to be well known anyway; after all, the proposed primary use of pCT is for mapping relative stopping power. Somewhat counter-intuitively however, several existing studies with other path estimation methods found little or no meaningful impact on resulting RSP accuracy from including prior knowledge of materials within a phantom[125,127,132]. In terms of the MLP formalism, it is posited that while the error will increase, the optimal predicted path is more or less unchanged by the introduction of slab-like material inserts along the beam line, excluding in large angle events[125,149]. Furthermore, in [133] no further accuracy benefit from including material information was shown over including the simpler exit energy as an input variable[133]. However in either instance it is noted that this may not hold for inhomogeneities with finite lateral dimension[133,149].

7.1 Related Study

Shortly after publication of our results[3], a separate group independently published an investigation into machine learning for pCT proton track prediction[133]. Neither was aware anyone else was pursuing a similar line of investigation.

[133] took a different approach to encoding the problem of proton trajectories, and operated under a situation conforming closer to that of a real world implementation. Tracks were parameterised using a separate variable, and models designed to instead predict the particles location for a given value of said variable, which is given as an input variable, as opposed to predicting the location at a set of fixed intervals at the same time. Input variables were given as would be measured by a simulated proposed pCT detector surrounding the phantom, including realistic detector errors, with paths predicted between the plates, whether phantom or not. Additionally, as

mentioned the exit energy was also given. Models were trained and evaluated on combined datasets of multiple different phantoms, with a range of thicknesses and including several realistic voxel phantoms of relevant anatomical sections.

Several different machine learning based models were examined. Similar to our approach, a feed forward neural network considering the problem as separate 2D planes was used, but with an additional accompanying network designed to estimate the error in the predictions of the first. Another used a feed forward neural network instead predicting complete 3D coordinate points, and a model using a XGB gradient boosted decision tree was also considered.

While the path averaging method used for comparison is conceptually different from the MLP formalism, it is similarly suitable in this context, and outside of short path lengths both the 2D approach and boosted decision tree showed superior performance. Further, the accuracy of the error prediction network suggested potential as a filtering method, perhaps as opposed to the 3σ cuts typically used with MLP.

Part III

Graph Neural Network Tracking

Chapter 8

Graph Neural Networks

8.1 Graphs

The graphs of pure mathematics have little to do with bar charts and other graphical means to display information. In their simplest form, graphs are a mathematical structure for representing a number of objects, and encoding the relationships between them[150,151]. From underground transport maps and flow charts, to chemical structures and atomic models, practical examples of graphs are familiar and can be found in a wide range of contexts. But while graphs are typically portrayed with diagrams of dots or circles and connecting lines, the choice of how a graph is depicted visually has no underlying significance. Indeed, graphs are not limited to when a graph structure is readily apparent. Graphs can characterise any system of objects in which relational links can be inferred or assumed, such as social networks[152] or even bouncing balls in a box[153,154].

Relatively speaking, graph theory, the mathematical study of graphs, is a fairly new discipline, with the bulk of developments occurring since 1890[155]. Its beginnings as a formal field of study are usually attributed to Euler's 1736 analysis of the Seven Bridges of Königsberg problem, a local pastime of attempting to devise a route over the cities seven bridges, crossing each only once[156]. Indeed puzzles, games and similar problems have often served as motivators in the development of graph theory[156].

Since the resolution of the four colour conjecture in 1976, graph theory has seen remarkable growth and increasing interplay with other areas of mathematics, as reflected in the amount of literature available, and now plays a remarkable role in applied mathematics and computer science[151].

8.1.1 Fundamentals of Graph Theory

While the core concepts of graph theory are well established, individual authors have their own preferences for exact definitions, terminology and notation. Further, graph theory as a rigorous mathematical discipline and the study of graphs in computing often differ significantly. This here is intended as an introduction to some of the concepts, not a rigours exploration, in order to be able to describe the concepts in graph neural networks used. What follows largely draws from the notation and definitions in [151], [155], [157], and expanding with ideas from [158].

With that in mind, let a graph G be defined as an ordered pair of sets $G = (V, E)$, where V is the set of vertices or nodes, and E the set of edges, such that the elements of E are two-element subsets of V, $E \subseteq [V]^2$ [155,157] *.

The number of vertices, $N^V = |V|$, is known as the order, and the number of edges, $N^E = |E|$, the size. If we consider $v_i, v_j \in V$ and $e_k \in E$ such that $e_k = v_i, v_j$, then we say that edge e_k joins vertices v_i and v_j. Should there be such an edge, then we say that vertex v_i and v_j are incident with edge e_k, and vice versa, and that v_i and v_j are adjacent, or neighbours, to each other. Furthermore, the degree of a vertex denotes the number of edges incident with it [151,157].

To illustrate, let us introduce the following example. Let G be a graph $G = (V, E)$ such that

$$V = \{v_1, v_2, v_3, v_4\} \tag{8.1.1}$$

$$E = \{e_1, e_2, e_3, e_4\} \tag{8.1.2}$$

where

$$e_1 = \{v_1, v_2\}, \quad e_2 = \{v_1, v_3\}, \quad e_3 = \{v_2, v_3\}, \quad e_4 = \{v_2, v_4\}\,. \tag{8.1.3}$$

One possible diagram representation of G is provided in Figure 8.1(a). From the above definition we can readily see that G has an order of 4 and a size of 4. If we take vertex v_1, then this vertex is incident with edges e_1 and e_2, and has two neighbours, v_2 and v_3. If instead we consider edge e_1, it joins, and is incident with, vertices v_1 and v_2.

*Generally, mathematical graph theory texts refer to vertices, while computer science graph neural network texts refer to nodes. For consistence we will use vertices throughout.

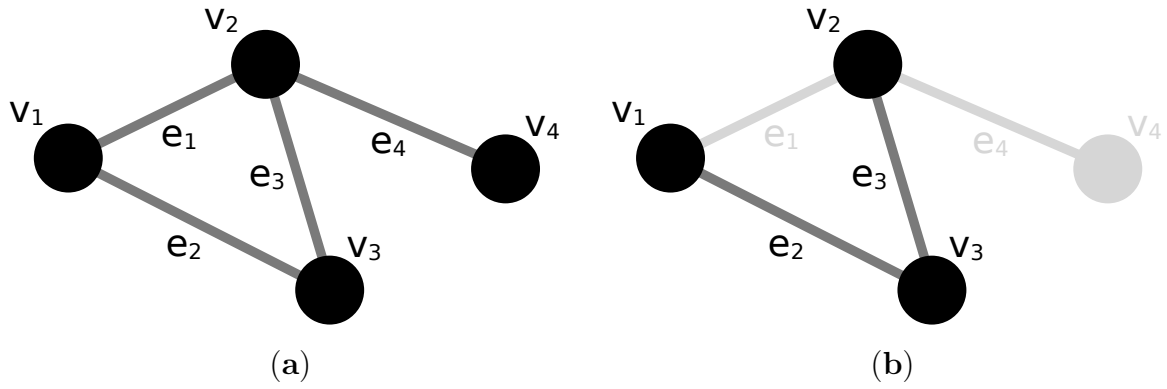


Figure 8.1: (a) A diagrammatic representation of the example graph G. (b) A diagrammatic representation of the example subgraph $H \subseteq G$ (in black), with G superimposed (in grey).

8.1.2 Pseudographs

Strictly speaking sets cannot contain multiples of the same element. With our definition of graphs so far, this precludes multiple edges which join the same pair of vertices, as a single element $\{v_i, v_j\}$ cannot be repeated[155]. Equally edges may not join a vertex to itself, as an edge is itself a set of two vertices, and individual vertices cannot therefore be repeated within an edge[155].

As they will become relevant later on, we therefore relax our definitions to specifically accommodate edges of these kinds. Formally such graphs are known as pseudographs, though we will continue to refer simply to graphs. Multiple edges joining the same pair of vertices are called skein, and such edges are said to be parallel to each other, while edges that join a vertex to itself are known as loops[151,155]. In order for a pair of edges in a directed graph to be parallel, they must also be orientated in the same direction[157].

8.1.3 Subgraphs

In the same way a subset is a portion of a larger parent set, a subgraph is in essence a portion of a larger parent graph. Specifically, a subgraph $G' \subseteq G$ is a graph $G' = (V', E')$ who's vertices and edges are subsets of the vertices and edges of G; $V' \subseteq V$ and $E' \subseteq E$. Furthermore, an induced subgraph is a subgraph that contains all edges $\{v_i, v_j\} \in E$ where $v_i, v_j \in V'$; that is, it contains all edges of G joining the vertices also present in G'. Conversely a spanning subgraph is a subgraph that retains all vertices of G, $V' = V$, but not necessarily any particular edges[157].

With our previous example in mind, let us introduce a graph $H = (W, F)$ such

that

$$W = \{v_1, v_2, v_3\} \tag{8.1.4}$$

$$F = \{e_2, e_3\} \ . \tag{8.1.5}$$

As all vertices $v_1, v_2, v_3 \in V$ and all edges $e_2, e_3 \in E$, H is a subgraph $H \subseteq G$, though it is neither an induced or spanning subgraph. A representation of H is given in Figure 8.1(b), superimposed with the graph G for convenience; however, there is no requirement for a diagram of a subgraph to take the same form as its parent. If we were to modify H to include $e_1 = \{v_1, v_2\} \in F$, then H would contain all edges from G joining vertices in F, which would make H an induced subgraph of G. Equally if $v_4 \in F$, then H would contain all vertices of G, making H a spanning subgraph of G.

8.1.4 Directed Graphs

Up until this point we have considered what are known as undirected graphs, in which edges denote a reciprocal relationship between nodes. In a directed graph however, edges are directional, denoting a one way relationship from an initial vertex to a terminal vertex. Let D be a directed graph $D = (V, E)$, where V is as before, but now let the elements of E be ordered pairs, such that for $v_{r_k}, v_{s_k} \in V$ and $e_k \in E$ the edge $e_k = \{v_{r_k}, v_{s_k}\}$ is a directed edge from an initial, or sender, vertex r_k, to a terminal, or receiver, vertex s_k[157]. We say that e_k joins v_{r_k} to v_{s_k}, but not s_k to r_k[151].

A symmetric directed graph is a directed graph in which for every edge, there is also a corresponding oppositely directed edge, that is $\forall\{v_{r_k}, v_{s_k}\} \in E$, $\exists\{v_{s_k}, v_{r_k}\} \in E$. The directed graph obtained by replacing all edges in a given undirected graph with pairs of oppositely directed edges is known as the associated directed graph[151]. Conversely, the undirected graph obtained by replacing all directed edges between a given two vertices by a undirected single edge is known as the underlying graph[151].

Once again let us introduce an example, reusing the same vertices as G. Let D be a directed graph $D = (V, J)$ such that

$$V = \{v_1, v_2, v_3, v_4\} \tag{8.1.6}$$

$$J = \{e'_1, e'_2, e'_3, e'_4, e'_5\} \tag{8.1.7}$$

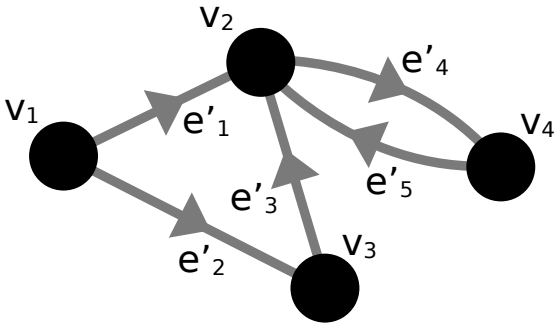


Figure 8.2: A diagrammatic representation of the example directed graph D.

this time where

$$e'_1 = \{v_1, v_2\}, \quad e'_2 = \{v_1, v_3\}, \quad e'_3 = \{v_3, v_2\}, \quad e'_4 = \{v_2, v_4\}, \quad e'_5 = \{v_4, v_2\}\,. \tag{8.1.8}$$

An illustration of D is given in Figure 8.2. As it stands, G is the underlying graph of D, as it does not matter if there are one or more edges in D with a corresponding edge in G, just that there is one joining the same vertices. If we were to introduce additional elements $e'_6, e'_7, e'_8 \in J$ such that $e'_6 = \{v_2, v_1\}$, $e'_7 = \{v_3, v_1\}$, $e'_8 = \{v_2, v_3\}$, then D would then be the associated directed graph of G.

8.1.5 Adjacency Matrix Representations

So far, the approach we have taken for describing graphs has been ill-suited for use in computing[151]. Graph or network embedding concerns representing graphs in a low-dimensional vector form, while retaining their important properties such as structure, enabling regular computing methods[159]*.

One particularly common method for expressing graphs is to encode the presence of edges in a matrix form[151]. Let G be a graph $G = (V, E)$ as usual. The adjacency matrix of the graph G is the $n \times n$ matrix $A_G := \left(a_{v_i,v_j}\right)$, where $n = |V|$, and the elements a_{v_i,v_j} denote the number edges joining the given pair of vertex v_i and v_j [151]**. This representation is applicable to both undirected and directed graphs. All undirected graphs, due to the reciprocal nature of edges, have symmetric adjacency matrixes, while directed graphs will when they are themselves symmetric.

*In mathematical graph theory, graph embedding refers to a different concept, involving the representation of a graph on a surface such that no edges will cross except at the vertices, and other conditions[151].

**For undirected graphs, loops are denoted as 2 edges[151].

To illustrate, for our example graphs G, H and D the corresponding adjacency matrixes A_G, A_H and A_D are given by

$$A_G = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{cccc} v_1 & v_2 & v_3 & v_4 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \end{array} \qquad A_H = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \end{array} \begin{array}{c} \begin{array}{ccc} v_1 & v_2 & v_3 \end{array} \\ \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \end{array} \qquad A_D = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{cccc} v_1 & v_2 & v_3 & v_4 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

Depending on a graph's complexity, it can be more compact to instead use a adjacency list representation. Let a list $N(v_i)$ be the list of adjacent vertices of a vertex v_i. Then the adjacency list L_G for the graph G is then the list of all lists $N(v_i)$ for the vertices v_i of G[151].

8.2 Graph Network Blocks

8.2.1 Weighted Graphs and Attributes

Particularly when modelling practical situations, there are many cases where we might want to characterise the edges of a graph with a value, such as assigning each a cost or importance. In graph theory, a graph with this kind of information is known as a weighted graph. For each edge e_k of a graph G, let there be an associated value $w(e_k)$, called as its weight; then G, together with the weights $w(e_k)$, is a weighted graph[151]. Theses weightings $w(e_k)$ can be considered as a vector indexed by the set of edges e_k. Weighted graphs are frequently used in applied mathematics, and many algorithms exist for solving problems described by weighted graphs.

However, it is possible to go further, assigning multiple properties to edges, and to vertices or even the graph as a whole. Here we are going to do so largely by drawing on the approach of the Graph Network framework, introduced in [158], which takes inspiration from a variety of graph, message passing and non-local neural network approaches. Though designed with graph neural networks specifically in mind, it is an intentionally general approach to incorporating attributes and applying algorithms to graphs[158].

First, let us introduce the concept of the vertex attribute $\mathbf{v}_i$, which represents various chosen properties of a vertex v_i. Similarly, let $\mathbf{e}_k$ be the edge attribute for an edge e_k, and let $\mathbf{g}$ be the global attribute for a graph as a whole[158].

Vertex, edge and global attributes are separate from one another, and while these attributes can be any arbitrary object, they take the same form for all vertices and all edges[158]. For our purposes, and simplicity, from here on we are going to say they are tensors; where we use tensor to refer to an n-dimensional array, rather than the rigorously defined mathematical structure in linear algebra. In this way, multiple weights can be encoded at the same time as the elements of said tensors, allowing for multiple properties, or features, to be described at once.

Let a graph G be now defined as $G = (\mathbf{g}, V, E)$, where $\mathbf{g}$ is the global attribute. Here, V is the set of vertices such that $v_i = \{\mathbf{v}_i\} \in V$, where $\mathbf{v}_i$ is the vertex attribute. Equally, E is the set of edges such that $e_k = \{\mathbf{e}_k, v_{r_k}, v_{s_k}\} \in E$, where $\mathbf{e}_k$ is the edge attribute, and $v_{r_k}, v_{s_k} \in V$[158].

If N^{EA} and N^{VA} are the number of features in edge and vertex attributes respectively, then similar to with a traditional weighted graph theses attributes can be considered, for edges or vertices, as a $N^{EA} \times N^E$ or $N^{VA} \times N^V$ tensor, where N^E and N^V are the number of edges or vertices respectively. Breaking down along the feature indexed dimension retrieves the values for a certain property, or feature, for all edges or vertices, in the same manner as a traditional weighted graph's weighting vector.

8.2.2 Operations

In order to manipulate these attributes, we shall again draw on and generalise ideas in [158], and introduce some operations to formalise interaction with the attributes. Let there be three forms of operation; edge, vertex and global operations, and let each operation consist of mapping a function, ϕ^e, ϕ^v and ϕ^g respectively, which is applied across all edges, all vertices, or once globally to give a new attribute,

$$\mathbf{e'}_k = \phi^e\left(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{g}\right) \tag{8.2.1}$$

$$\mathbf{v'}_i = \phi^v\left(\overline{\mathbf{e}}_i, \mathbf{v}_i, \mathbf{g}\right) \tag{8.2.2}$$

$$\mathbf{g'}_i = \phi^g\left(\overline{\mathbf{e}}, \overline{\mathbf{v}}, \mathbf{g}\right) . \tag{8.2.3}$$

The attributes $\overline{\mathbf{e}}_i$, $\overline{\mathbf{e}}$ and $\overline{\mathbf{v}}$ are the result of aggregate functions, ρ, that reduce any number of attributes from incident edges or vertices into a single representative

attribute,

$$\overline{\mathbf{e}}_i = \rho^{e\to v}(E_i) \tag{8.2.4}$$

$$\overline{\mathbf{e}} = \rho^{e\to g}(E) \tag{8.2.5}$$

$$\overline{\mathbf{v}} = \rho^{v\to u}(V) \tag{8.2.6}$$

where $E_i \subseteq E$ is the subset of all $e_k = \{\mathbf{e}_k, r_k, s_k\} \in E$ such that $r_k = v_i$. These new attributes need not be the same size as the original, and while formally each attribute is replaced, the updated attribute may simply extend to the original, retaining all previous elements and adding new ones. On a practical level, the functions ϕ^e and ϕ^v can be implemented as being applied to the respective $n \times m$ tensor representation of attributes.

8.2.3 Graph Network Blocks

Combining the edge, vertex and global operation, a Graph Network Block[158], or GN Block, represents a full update pass of all attributes, using the operations just introduced. As a flexible framework, a wide range of graph algorithms can be characterised using a sequence of GN Blocks, giving a common approach to describe how disparate algorithms operate. Following from what has been introduced, a GN Block can be summarised as a series six computation steps[158], though it need not have all stages;

1. $\mathbf{e}'_k = \phi^e\left(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{g}\right)$ is performed per edge, replacing each edge attribute $\mathbf{e}_k$ by $\mathbf{e}'_k$.

2. $\overline{\mathbf{e}}'_i = \rho^{e\to v}(E'_i)$ is performed per vertex, aggregating incident edge attributes, to generate a representative attribute $\overline{\mathbf{e}}'_i$ for each vertex.

3. $\mathbf{v}'_i = \phi^v\left(\overline{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{g}\right)$ is performed per vertex, replacing each vertex attribute $\mathbf{v}_i$ by $\mathbf{v}'_i$.

4. $\overline{\mathbf{e}}' = \rho^{e\to g}(E')$ is performed for the graph as a whole, aggregating all edge attributes into a single representative attribute $\overline{\mathbf{e}}'$ for the graph as a whole.

5. $\overline{\mathbf{v}}' = \rho^{v\to u}(V')$ is performed for the graph as a whole, aggregating all vertex attributes into a single representative attribute $\overline{\mathbf{v}}'$ for the graph as a whole.

6. $\mathbf{g}'_i = \phi^g\left(\overline{\mathbf{e}}', \overline{\mathbf{v}}', \mathbf{g}\right)$ is carried out, replacing the global attribute $\mathbf{g}_k$ by $\mathbf{g}'_k$.

In practice, steps 2. and 3. can readily be combined per vertex, performing the edge aggregation then applying the update function for a vertex before progressing to the next.

8.2.4 The Message Passing Paradigm

Introduced in [160], the Message Passing Neural Network generalised and formalised concepts implicit in many GNN[160], and is one of the principal inspirations for the GN Block framework[158]. Since then, the message passing paradigm has emerged as an underlying concept in GNN design[159,161], and part of what makes them effective; indeed, it has been suggested that the principal is inherent in all modern GNN designs in some manner[161]. While envisioned with neural networks in mind, just as with the GN Block framework, the principal is readily generalisable to graph algorithms in general.

The rough concept is that 'messages' are formed for each edge, incorporating attributes from edges themselves and joined vertices. These messages are then passed to the joined vertices (to the terminal vertex in directed graphs), and vertex attributes updated incorporating these messages; thus attributes are influenced by their neighbours. Through repeated application of the algorithm, information effectively flows around the graph, allowing unconnected parts of a graph to be influenced by those distant. The graph's structure governs how information flows, thus incorporating its structure without having to be explicitly encoded into an algorithm.

Referring back to the GN Block framework as given in Section 8.2.3, the first three steps of the framework can be considered as a direct representation of this;

1. The message function ϕ^e form a message for each edge, effectively storing them as attributes on said edges.

2. The reduce function $\rho^{e\to v}$ takes in and combines the incoming messages for each vertex, compiling them into a single aggregate message.

3. The update function ϕ^v uses this aggregate message to update the attributes on each vertex.

Many iterative algorithms fitting the GN Block framework can be considered as using a form of the message passing paradigm, and we will continue to refer to ϕ^e, $\rho^{e\to v}$ and ϕ^v as the message, reduce and update functions.

8.3 Graph Neural Networks

Graph neural networks, or GNN, are a form of deep learning model in the field of geometric deep learning[162], designed to operate on a graph in an end-to-end manner as opposed to translating a task into a non-graph form for operation[159]. Whether the later form of models are considered as GNN or not varies. The use of neural networks with graphs first occurred over two decades ago in [163], with the concept of what we would now refer to as a graph neural network later floated in [164]; and the first true GNN model is often attributed to [165], which extended existing neural network models for use with graphs[150,159,166]. Since then GNN have grown in popularity, and have been successfully applied to various graph analytics tasks, such as vertex (edge, or whole graph) classification, link prediction, and clustering[166].

In many ways, GNN can be viewed as a further generalisation of convolutional neural network principals. Whereas CNN operate on regular euclidean data structures, archetypally images or text, graphs present a more broad structure, and early motivations came from seeking to expand the same underlying principals and operations that make CNN successful to graphs[150,166]. Indeed, a digital image, and other grid-like structures, can be considered as a specific form of graph; each pixel is a vertex, with edges connecting each to its immediate neighbours in the image matrix[150].

Referring back to the GN Block framework, it is straightforward to see how the ideas introduced in Section 8.2 apply to GNN, by introducing neural networks, such as the feed forward neural network, in the role of the message or update functions.

8.3.1 Forms of GNN

Despite the relatively young age of GNN within the study of neural networks, a myriad of models have already emerged. Among them several rough design principals can be identified, though with the interrelated nature of models, many different taxonomies and approaches to categorisation can be taken[166]. More thorough surveys of current graph neural networks methods can be found in reviews such as [166] and [159], or textbooks such as [150].

The model introduced in [165], simply named to as the graph neural network model, is often taken as the 'vanilla' GNN[150]. A feed forward neural network is used as a function to learn to generate a representative state embedding for each vertex, that describes the neighbourhood of that vertex. Exploiting Banach's fixed point theorem, this function is performed repeatedly in order to converge to a final

result. These state embeddings are subsequently used by a separate function to generate an output label for each vertex[150,165,166]. However this model experiences several drawbacks. Iterative application to reach a fixed point is a computationally inefficient approach[159], and the straight repeated application of a FNN in this way is essentially the same as a FNN with more layers. In effect, the FNN is concatenated with copies of itself, but without some of the advantages separate neuron weightings would bring. Additionally, not all information encapsulated by an edge can be modelled effectively, and thus is effectively lost, and issues can emerge if the number of iterations is too large[150,166].

Building on the same iterative approach to converge on a stable representation[159], Graph Recurrent Networks, or GRN, work to overcome these limitations and improve long-term information propagation by looking to recurrent neural networks and LSTM models[150]. Designed for tasks which require outputting sequences of results, the Gated Graph Neural Network and Gated Graph Sequence Neural Network[167] introduce Gate Recurrent Units into the iterative part[150], and the later has been successfully applied to text understanding and program verification tasks[150]. LSTM architectures have been implemented in a similar manner, with [168] and [169] extending Tree-LSTM architectures[170] (LSTM with a tree-like network topology, as opposed to the usual chain-like topology) to graphs[150]. Regardless of form, the repetitive iteration within the models can be seen as a form of message passing, propagating information between vertices[159].

Inspired by GRN[159], Graph Convolutional Networks, or GCN, generalise convolution operations, the heart of convolutional neural networks, to the graph domain. Given the success of CNN at a wide number of tasks, GCN are a popular approach with a vast number of variations[150]. GCN models are often grouped into two forms based on what the convolution operation kernels operate on. Spectral methods, such as the Spectral Network[171] and the archetypally named GCN model[172], operate on the spectral representation of graphs, with convolutions applied to the eigendecomposition of the graph Laplacian. However, as the learned filters then depend on the Laplacian eigenbasis, trained model instances are specific to a certain graph structure[118,166]. Alternatively, spacial methods such as Neural FPS[173] and GraphSAGE[174] define convolutions directly for the graph itself, applying to a vertex and its neighbours. While conceptually more straightforward, spacial methods must contend with defining convolutions that can handle different numbers of neighbours while preserving local invariance[150,166]. In a similar manner to GRN, spacial GCN incorporate the concept of message passing, with the convolution operation acting

to share messages between vertices[159].

While a regular GCN kernel treats all neighbouring vertices equally, Graph Attention Networks, or GAN, such as GAT[175] or GAAN[176] further incorporate an attention mechanism, allowing the designating of different importances to each neighbour[150].

Given the often sparse, irregular and relational nature of data collected at accelerator experiments, graphs offer a potent tool with which such data can be represented. It is therefore little wonder that with the dearth of methods available, there have been recent efforts to explore the potential of GNN for a variety of roles in experimental high energy physics, such as for clustering, particle identification, calibration and simulation[162]. A comprehensive technical review of studies into GNN at the LHC can be found in [162].

8.3.2 The Interaction Network

Designed for reasoning how complex systems interact and their dynamic evolve, the Interaction Network, or IN, introduced in [153] has shown great success at problems such as bouncing balls in a box, N-body problems, and the motion of strings[153].

Consider some dynamic system of objects at a time step t. Let this system be described by a graph $G_t = (\mathbf{g}, V, E)$, where the vertices V correspond to the set of objects, and edges E the relations between objects. The influence of one object on another is not necessarily symmetric, so G is a directed graph, and both objects and their relations are naturally described using various values, which will be encoded as attributes. The set of vertices $v_i = \{\mathbf{v}_i\} \in V$ therefore represents the state of each object, and the edges $e_k = \{\mathbf{e}_k, r_k, s_k\} \in E$ the influence exerted by object r_k on object s_k, or relation. Any external effects on the system as a whole are included as $\mathbf{g}$. Let the IN propagate the system of objects to a time step $t+1$, so that $G_{t+1} = \mathrm{IN}(G_t)$.

A basic IN unit is then defined as[153]

$$\mathrm{IN}(G_t) = \psi^V\left(a\left(V, \mathbf{g}, \psi^E\left(m\left(G_t\right)\right)\right)\right) \tag{8.3.1}$$

$$\begin{aligned} m(G_t) &= B = \{b_k\} & a(V, \mathbf{g}, H) &= C = \{c_i\} \\ f^e(b_k) &= h_k & f^v(c_i) &= p_i \\ \psi^e(B) &= H = \{h_k\} & \psi^v(C) &= P = \{p_i\} \end{aligned}$$

where k indexes the edges, and i the vertices. The IN process can be described

as follows[153];

- The marshalling function m forms interactions terms, b_k, that combine into one place the information on an interaction between objects, therefore the attributes of each edge, sender and receiver vertex; $b_k = \{\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}\} \in B$.

- The relation model ψ^E predicts the effect of each interaction, $h_k \in H$, through applying the function f^e to each b_k.

- The aggregation function a collects up these interaction effects $h_k \in H$ corresponding to a given receiver vertex v_{r_k}, and combines them along with $\mathbf{v}_{r_k} \in v$ and $\mathbf{g}$ to form a single $c_{r_k} \in C$ for that vertex, a representation of the combined interaction effects.

- The object model ψ^v predicts how the combined interaction effects influences each vertex v_i, by applying the function f^v to all c_i, producing a result $p_i \in P$ for each vertex.

For a dynamic system, the result P may represent the new state of the system, G_{t+1}, with $\mathbf{v}_{i,t+1} = p_i$ the new states of the objects within the system. The dynamic system represented by G is therefore evolved in time through iterative application of $\mathrm{IN}(G)$; with each application acting as a message passing. However the IN is more general in application. For example, the model may be expanded with a final aggregation function, and after a single, or perhaps several iterations, said function takes results P, along with $\mathbf{g}$, in order to produce a result characterising the graph as a whole[153].

In the context of a machine learning model, functions f^e and f^v take the form of feed forward neural networks[153]. The same f^E and f^V are applied to every b_k and c_i respectively, allowing the model to apply to graphs with arbitrary numbers of vertices and edges, and irrespective of the order in which functions are performed. To allow for this, the aggregation function a needs to be commutative and associative, and in the example implementation in [153] summation of h_k is used.

The IN served as a forerunner and inspiration to the GN Block framework, which in many ways can be seen to generalise the IN. Translating into the format of the

framework[158]

$$\phi^e\left(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{g}\right) = f^e\left(b_k\right) = f^e\left(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}\right) \tag{8.3.2}$$

$$\phi^v\left(\overline{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{g}\right) = f^v\left(c_i\right) = f^v\left(\overline{\mathbf{e}}'_i\right) \tag{8.3.3}$$

$$\rho^{e\to v}(E_i) = a\left(\mathbf{v}_i, \mathbf{g}, H_i\right) = \{\sum_{k:\, r_k=i}\left(h_k\right)\} = \{\overline{\mathbf{e}}'_i\} \tag{8.3.4}$$

where $H_i \subseteq H$ is all $h_k \in H$ such that $r_k = v_i$. As whole graph attributes $\mathbf{g}$ are not updated, this corresponds to performing stages 1 through 3 of the GN Block framework.

8.3.3 The HEP.TrkX and Exa.TrkX Tracking Model

Exploring a variety of tracking challenges, the pilot HEP.TrkX project[177] proposed the use of GNN for tracking[178], with its ongoing successor the Exa.TrkX project[179] successfully applying particle tracking for ATLAS and CMS style barrel detectors[180]. In a novel approach, the model took inspiration from the Interaction Network introduced in [153]. Though the Interaction Network was designed with a rather different task in mind, the architecture proved remarkably successful for tracking[180,181]; and similar models have since seen success in track finding in the whole ATLAS detector[182], particle flow reconstruction[183], and object reconstruction in liquid argon time projection chambers[184]. Models of this style have been evaluated for use on FPGAs[185], and GNN methods are now under consideration for future ATLAS detector triggering during HL-LHC[186].

The Exa.TrkX model * has since been generalised as a versatile, fully-learned pipeline that can be extended for a variety of tracking problems[162,181,187]. Refering to [181] and [180], the model can be summarised as several stages.

First, data is preprocessed into a suitable format, with each hit within the detector a data-point. The data prepared includes spatial coordinates, pixel cluster shape information and others depending on use case.

As the pipeline is designed to be detector agnostic, the graph representation is not constructed based on a specific detector layout. Instead, in the second stage the Embedding Network, a Multi-Layer Perceptron, takes each data-point and produces a latent space representation. This consists of 6 layers of 512 neurons, with hyperbolic tan activation functions and normalisation. A final layer returns an 8 laten feature

*[181] refers to the model as the TrackML pipeline. However as this is also the name of a specific dataset used across various papers, we will refer to it as the Exa.TrkX model.

representation. From this a graph is formed, each data-point becoming a vertex, and vertices joined to others within a given radius in the latent space representation.

In the third stage another perceptron, the Filter Network, is applied to each edge, taking the two vertices as inputs and returning an edge efficiency score. This consists of 3 layers of 1024 neurons, with a final binary cross-entropy loss function. Applying a cut based on this score greatly reduces the number of edges, saving significantly on computational costs in subsequent stages.

The fourth stage is the graph neural network, using the Interaction Network architecture. 8 iterations are performed following the process as outlined above in Section 8.3.2. Aggregation is performed using summation, and the vertex and edge functions consist of 2 layer perceptrons, of 128 and 64 neurons, with ReLU activation functions. The network also features a form of skip connection, in which after each iteration the output is combined with the values from the previous iteration to form the set of input values for the next. The last layer of the network returns a binary classification score for each edge, scoring the likelihood it corresponds to a segment of a track.

The final stage of the pipeline consists of task-specific processing, based on the desired use and output of the specific model. Notably as performed in [180], for track seeding, after a single edge network perceptron is applied to produce characteristic scores for each edge, the graph may be converted to a triplet graph, where vertices correspond to high score edges in the original graph, connected by an edge if they share a hit. A separate similar GNN applied to this new graph can produce results with a high fraction of tracks matching particles[181].

Chapter 9

CERN and the Standard Model

9.1 The Case for High Energy Particle Physics

9.1.1 The Standard Model

The Standard Model of particle physics is the theoretical model describing all known elementary particles, and their interactions through three of the four known fundamental forces[188]. A product of decades of research and thousands of scientists, it represents our best current understanding of matter[189].

At its heart, the standard model is a compilation of the laws of physics that describe the behaviour of subatomic particles. As a quantum field theory, incorporating the principals of both quantum and relativistic mechanics, physical systems are represented by quantum fields, and particles as the quantised excitations of these fields[190,191]. Elementary particles and their interactions emerge as inherent features of the system itself; deriving, through Noether's theorem, from imposing invariance under local phase transformation[191,192].

The resulting fundamental particles can be differentiated into two families, fermions and bosons. Elementary fermions act as the building blocks of matter, with all stable matter in the universe made from the lightest and most stable among them[189]. With half integer spin, they are subject to the exclusion principal, precluding any two identical fermions from occupying the same state[191]. Fermions are further divided into two types, each of which is composed of six varieties, in three sets of pairs[189]. Quarks interact with the strong force, which binds them together to form composite particles such as protons and neutrons. The binding effect is strong enough that, so far, isolated quarks have never been observed; either hadronizing or decaying on a minuscule timescale[193]. Leptons on the other hand do not interact with the strong

force, and consist of the electron, muon, tau, and a corresponding neutrino flavour for each[189]. While the first three have an electric charge and increasing mass, their partner neutrinos are electrically neutral and with negligible or no mass, making them difficult to detect[193].

In contrast, elementary bosons are the mediators of interactions. Within the standard model, the fundamental forces manifest through the exchange of gauge bosons between matter particles, with each fundamental force possessing its own corresponding bosons'[189,190]. With no electric charge of their own, the carriers of the electromagnetic force, photons, do not interact with one another but consequently exist as free particles. The W and Z particles, carriers of the weak force, poses their own charge equivalents, while gluons, carrier of the strong force, not only poses their own colour charge, but are predicted to form bound states of several gluons joining together; though this has yet to be detected experimentally[190]. The Brout-Englert-Higgs Field, and the corresponding Higgs boson, operate in a different manner, instead imparting mass to those particles with which they interact[190,194].

While the full Standard Model Lagrangian representation stretches into many pages, it can be summarised in a compact form as

$$\begin{aligned}\mathcal{L} = &-\frac{1}{4}F_{\mu\nu}F^{\mu\nu} \\ &+ i\bar{\psi}\not{D}\psi + \text{h.c.} \\ &+ \bar{\psi}_i y_{ij}\psi_j\phi + \text{h.c.} \\ &+ |D_\mu\phi|^2 \\ &- V(\phi)\end{aligned} \tag{9.1.1}$$

where h.c. denotes the hermitian conjugate of the preceding term, and ψ the quantum fields[190]. Through the field strength tensor F, the first line describes how the gauge bosons, aside from the Higgs, manifest and interact with one another. The second line describes, through the covariant derivative $\not{D}$, how the same gauge bosons instead interact with matter particles. The third line describes how the fermions interact with the Brout-Englert-Higgs field, ϕ, with the coupling parameters encoded in the Yukawa matrix, y_{ij}. Of the gauge bosons, only the weak force carriers interact with the Brout-Englert-Higgs field, as described by the fourth line. Finally, the fifth line describes the potential of the Brout-Englert-Higgs field, which unlike the other quantum fields does not have a single minimum at zero, leading to spontaneous symmetry-breaking[190].

The standard model gives rise to the the gauge symmetry group

$$\mathrm{SU}(3) \times \mathrm{SU}(2) \times \mathrm{U}(1) \tag{9.1.2}$$

where SU(n) and U(n) refer to the special unitary and unitary groups of degree n[193]. The SU(3) symmetry corresponds to Quantum Chromodynamics, the component of the standard model concerned with the strong interaction between gluons and quarks[193]. The SU(2) × U(1) symmetry governs interactions of the weak and electromagnetic forces, unified into the electroweak model[193]. As mentioned, the Brout-Englert-Higgs field breaks this symmetry, leading to some particles possessing mass, and others not[190].

9.1.2 Unanswered questions

While the Standard Model has proved resilient under decades of scrutiny, it presents an incomplete picture[190,191].

Though the standard model encompasses quantum theory with special relativity, combining quantum theory with the full theory of general relativity is notoriously difficult, and we have yet to successfully do so in the context of the standard model[189]. Thus, while it describes three of the four known fundamental forces, inclusion of gravity remains illusive[189,190]. Equally there are various specific properties the theory does not predict, such as the entries of the Yukawa matrix[190], and while the Brout-Englert-Higgs Mechanism explains the masses of some particles, there are still open questions such as why neutrinos appear to have mass[193].

On a grander scale, though there is a limited asymmetry between matter and antimatter within the standard model, it does not go far enough to account for the apparent preponderance of matter in the universe, or line up with our current understanding of the Big Bang[189,191]. Nor does it describe dark matter and dark energy, or the alternative modified gravity, implied by the motion of the cosmos[191,195,196]. Thus, while the standard model represents our best understanding of the fundamentals of matter, it does so as part of a bigger picture; one that is still being explored[189,190,191].

9.1.3 The Role of Particle Accelerators and Detectors

The study of fundamental subatomic particles is thus about more than just the manifestation of particles themselves, but offers a means to probe the underlying

physics with which they are inherently intertwined.

As staples of ordinary matter, electrons and protons can be readily found. More exotic particles on the other hand pose more of a challenge. One source of such particles is from cosmic rays; as high energy particles bombard the atmosphere, they can produce showers of secondary particles such as muons and neutrinos. Alternatively, as radioactive nuclei decay they may emit various particles, making nuclear reactor emissions another potential source of material for study[7,191].

However these methods offer limited control over the range of particles produced. Through conservation of energy and the mass-energy equivalence of relativity, heavier particles can be produced through high energy collisions; the intense moment of high energy providing the possibility for such particles to come into being. Going even further, desired particles such as positrons, muons or neutrinos can be siphoned off to produce secondary beams for further experimentation. Offering a means to not only generate collisions, but fine tune energies and other conditions to maximise the likelihood of desired outcomes, particle accelerators have become a mainstay of modern particle physics research[40,191].

Controlling where collisions take place allows advanced particle detectors to be located right by the action, making measurements immediately after an event. Even then, exotic particles often rapidly decay or may be involved in other interactions; and it is up to scientists to deduce what unfolded. Through study of scattering, decays and bound states, we can measure properties and place boundaries on standard model physics, and hunt for signs of what lies beyond[7,191].

9.2 CERN and the Large Hadron Collider

9.2.1 The CERN Accelerator Complex

Born out of a desire for closer collaboration and the increasing costs of cutting edge research facilities, 1953 saw the formal foundation of the Conseil Européen pour la Recherche Nucléaire, or as its more commonly known in English, CERN[197]. In the 70 years since, CERN has grown into a truly international collaboration of scientists, now encompassing over $12,000$ users and 600 institutions from all over the globe[198,199]. As our understanding of fundamental matter pushed deeper than the nucleus, so too did CERN follow into the emerging domain of particle physics; with its principal complex now frequently referred to as the European Laboratory for Particle Physics[200].

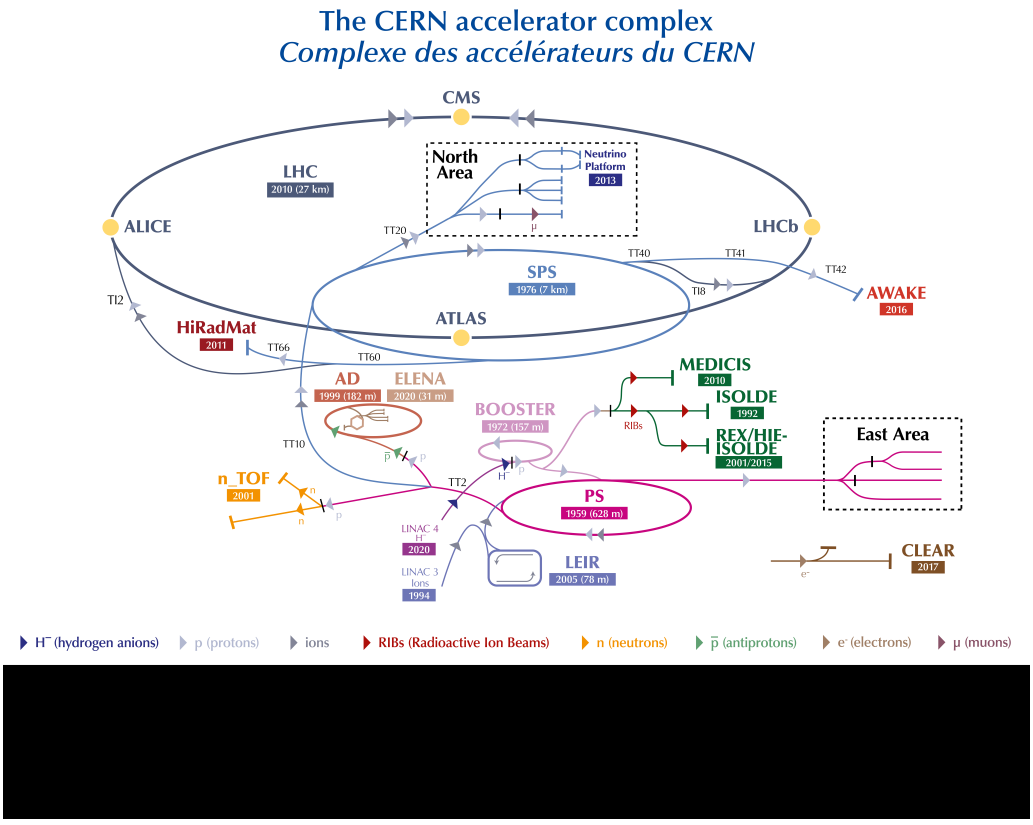


Figure 9.1: Diagrammatic representation of the CERN accelerator complex as of 2022. Reproduction of [201]

Since CERN's first particle accelerator opened its doors in 1957 [202], the CERN accelerator complex has evolved into a record breaking installation hosting a wide range of experiments [203]. Not just a collection of disparate apparatus, as can be seen in Figure 9.1, multiple accelerators build upon one another in succession, incorporating a multitude of experiment end points [204]. Indeed, most CERN accelerators are still in use [202], with previously cutting edge installations serving as pre-accelerators or similar for subsequent generations [203,205].

9.2.2 The Large Hadron Collider

Straddling the France-Swiss border, the Large Hadron Collider, or LHC, stands as the most powerful particle accelerator ever built, and will remain so for at least the next two decades [199]. Replacing the LEP Collider as CERN's flagship accelerator, the LHC is capable of record breaking nominal collision energies of 13 TeV [205], has already seen the publication of a staggering 2852 papers in its first decade of operation [206]. Though principally a proton-proton experiment, heavy ions, particularly

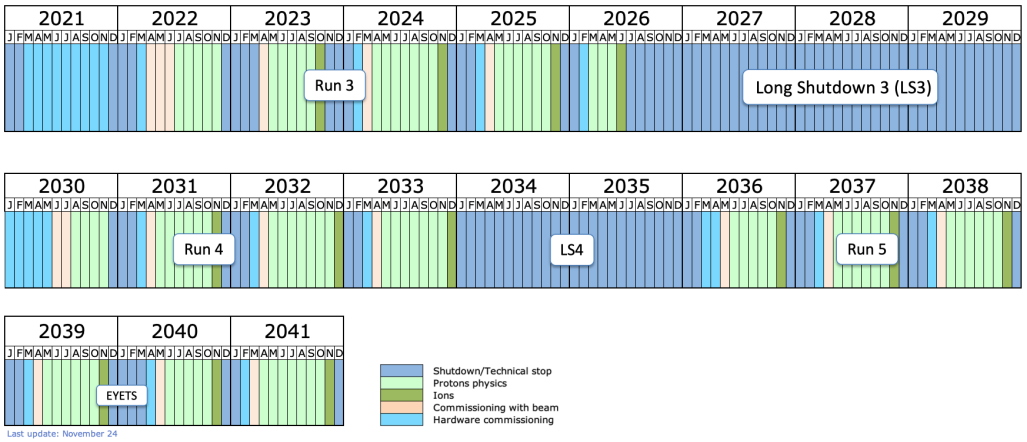


Figure 9.2: Long term schedule for operation of the LHC from 2021, as of September 2024. Reproduction of [207].

lead ions, are also routinely collided for atomic research [205].

Data taking operations are divided into a series of runs separated by long shutdown periods for maintenance and upgrade programs. Runs themselves are punctuated with short yearly winter technical stops, and all shutdowns are succeeded by commissioning periods. The first formal data taking period began with Run 1 in 2011, and stretched into 2013; while Run 2 lasted from 2015 to 2018. Originally scheduled to begin in 2021, Run 3 was postponed to 2022 to account for the challenges brought by the COVD-19 pandemic, and is currently ongoing. A summary of the future LHC operating schedule can be found in 9.2 [207].

Since 2020, Linear Accelerator 4 has taken on the role of producing initial proton beams for the CERN accelerator complex [44,203]. Negative hydrogen ions are accelerated to 160 MeV and injected into the Proton Synchrotron Booster, stripping them of electrons to leave only protons. Here particles are accelerated to 2 GeV, before transfer to the Proton Synchrotron, further accelerating beams up to 450 GeV [203].

Finally, protons are injected into the LHC itself. At almost 27 km in circumference, the Large Hadron Collider is a two-ring synchrotron located between 45 m and 170 m underground, utilising the tunnel complex originally constructed for the LEP Collider [39]. Though often thought of as circular, the collider consists of eight arcs, broken up by eight approximately 528 m long, evenly spaced straight sections. These straight sections, numerated Points 1 through 8, serve as the sites of detector apparatus or other utility purposes [39]. Within the two beam pipes, protons are circulated in opposite directions in a series of discrete bunches, and can be maintained circling for many hours [203]. Protons are kept in their circular path by the magnetic lattice, a collection of thousands of magnets of different varieties and sizes [39,208], including the

1232 superconducting main dipole magnets[39,205]. Acceleration is performed using a 400 MHz superconducting radio-frequency cavity system, featuring 8 cavities for each beam[39,205].

9.2.3 Beam Conditions

Using insertion magnets to squeeze them together, the two beams are brought into collision in four places around the collider's circumference, each host to one of the principal detector experiments[39,203]. Typically in the context of the LHC, an event denotes a single crossing of proton bunches[30]. However for this section an event is used in the more general sense of a particular collision interaction.

The rate at which an accelerator produces collisions is typically characterised by the instantaneous luminosity, $\mathcal{L}$. For a Gaussian beam distribution, such as those at the LHC, and assuming equal size particle bunches, this can be found by

$$\mathcal{L} = \frac{N_b^2 n_b f_{\text{rev}} \gamma}{4\pi \varepsilon_n \beta^*} F \tag{9.2.1}$$

where N_b is the number of particles per bunch, n_b the number of bunches per beam, and f_{rev} the frequency of revolutions made by the beam. γ is the Lorentz factor, ε_n the normalized transverse beam emittance, and β^* the beta function at the collision point, which relates to the bunch dimensions[39,209]. In many detectors, including those at the LHC, beams are not collided straight on, but rather with a small offset angle. This is accounted for by the inclusion of a reduction factor, F, which for small angles can be given as

$$F = \left(\left(\frac{\theta_c \sigma_z}{2\sigma^*} \right)^2 \right)^{-\frac{1}{2}} \tag{9.2.2}$$

where θ_c is the crossing angle, and σ_z and σ^* the root-mean-square bunch length and transverse beam size at the interaction point[39,209]. Given the cross section for a particular interaction, δ_{event}, we can recover the number of events per second by $\mathcal{L}\sigma_{\text{event}}$[39,209]. With around 2800 bunches per beam and 1.2×10^{11} initial protons per bunch, during the 2015 to 2018 window the LHC was producing in the order of a billion collisions per second[205], and achieved a peak instantaneous luminosity of $\mathcal{L} = 2 \times 10^{34}$ cm^{-2}s^{-1}[199].

In practice, over the lifetime of a particular particle batch, or fill, the luminosity will decay, particularly as collisions naturally reduce the number of particles in the beams[39]. Therefore we often instead consider the total quantity of collisions, rather

than focusing on the rate at which they occur. This can be characterised using the integrated luminosity, where over some period T,

$$\mathcal{L}_{\text{int}} = \int_0^T \mathcal{L}(t')\, dt' \,. \tag{9.2.3}$$

Similar to with instantaneous luminosity, given the cross section for a particular event the total number of such events is given by $\mathcal{L}_{\text{int}}\sigma_{\text{event}}$ [209], and between 2015 and 2018, the LHC produced an integrated luminosity of 160 fb^{-1} at its two high luminosity detector experiments [205,210].

9.2.4 Detector Experiments

Though the LHC is best known for the confirmation of the Higgs [211,212,213], that was only one part of a far wider physics program [204,214]. As a whole, the CERN accelerator complex serves a diverse range of experiment installations, with the LHC itself playing host to 9 distinct experiments [204], each designed and operated by its own international collaboration.

Of these, the 4 large experiments sit right at each of the 4 interaction points on the LHC's circumference where particle collisions occur [203]. As a general purpose detector, ATLAS, or A Toroidal LHC Apparatus, studies a broad range of phenomena, including the Higgs, extra dimensions, and potential dark matter candidates. The largest particle detector ever constructed, ATLAS thus aims to make a wide range of measurements with as much coverage around the interaction point as possible [215,216]. Sharing broadly the same scientific goals, CMS, the Compact Muon Solenoid, is in many ways a sister detector to ATLAS. However it approaches its aims independently, with its own separate design and technical solutions. This enables the cross-confirmation of any new discoveries and potential combining of results, such as in the first observations of the Higgs [204,211,216,217].

LHCb on the other hand is a specialist detector, with an initial focus on the behaviour of heavy-flavour quark particles in order to probe charge-parity violation. Providing high precision measurements of low deflection particles, LHCb has broadened into a general purpose forward acceptance detector [5,6,218]. ALICE, A Large Ion Collider Experiment, places its focus on the LHC's alternative ion collision program. Focusing on the physics of the strong interaction, ALICE explores the behaviour of quark-gluon plasma; a state of matter believed to have existed shortly after the Big Bang, and only possible at extreme values of energy density and temperature. [219,220].

In addition, 5 smaller experiments utilise LHC collisions in different ways. Both

TOTEM and LHCf examine those particles thrown forward by collisions, with very low deflection from the beam line[204]. Spread at almost half a kilometre around CMS, TOTEM makes precise measurements of such particles in order to investigate elastic scattering and diffractive processes, offering insight into proton behaviour and structure[221,222]. Exploiting similarities to extremely high-energy cosmic-rays, LHCf sits either side of ATLAS, providing measurements to refine hadron interaction models[223].

Dedicated to hunt for messengers of new physics potentially missed by the more general detectors, MoEDAL, now expanded to the MoEDAL-MAPP, searches for highly ionizing particles, including feebly ionizing and long lived particles[224]. With data taking beginning in 2022, FASER and SND@LHC are the LHC's two newest experiments[204]. The ForwArd Search ExpeRiment, or FASER, searches for light and very weakly-interacting new particles[225]. Sensitive to the full spectrum of neutrinos, SND@LHC, or the Scattering and Neutrino Detector at the LHC, makes precision measurements of neutrinos, probing heavy flavour physics[226].

9.2.5 The High-Luminosity LHC

One avenue for improving the accuracy of measurements is to reduce statistical errors through increasing quantities of data. The LHC has already seen substantial upgrades, pushing nominal collision energies from $\sim$7 TeV, later $\sim$8 TeV, in Run 1 to 13 TeV in Run 2[199]; but while the current LHC has already reached twice its nominal design instantaneous luminosity, particles such as the Higgs are a rare occurrence[202].

Therefore in the coming years the LHC will undergo a far reaching upgrade and replacement program to extend its working lifetime at the cutting edge of physics. Dubbed the High-Luminosity LHC, or HL-LHC, this revitalised accelerator aims to reach a fivefold increase in the instantaneous luminosity over the nominal LHC design value, allowing scientists to push the boundaries of physics for many more years to come[199]. Not only does this pose a huge technical challenge in itself, but experiment detectors will also see their own significant upgrade programs in order to contend with the increased radiation damage and pileup[199,202,227].

Chapter 10

The LHCb Experiment

10.1 Physics at LHCb

With a core program of charge-parity violation and rare beauty and charm hadron decays, LHCb is the Large Hadron Collider's dedicated flavour experiment. Focusing on high precision measurement, the detector's single-arm spectrometer design exploits the prevalence of beauty and charm hadron decays (from which the 'b' in its name derives) in the forward region. Upgrades have seen LHCb broaden its capabilities as a general purpose forward acceptance detector, with interests ranging from electroweak physics to heavy ion collisions[5,6,23].

As a necessary condition for baryon asymmetry, understanding the origins and mechanisms of charge-parity violation is a key question in particle physics. In the standard model, all such violation is described by the CKM mechanism, but this alone is insufficient to explain the observed baryon asymmetry in the universe. Decay processes such as $B^{\pm} \to DK^{\pm}$, $B^0 \to DK^{*0}$ and $B_s^0 \to D_s^{\mp}K^{\pm}$ that are described by tree amplitudes alone provide a means to refine the angle γ of the CKM unitary triangle, and make precise tests of standard model quark mixing predictions[6].

New particles beyond the standard model may potentially enter loop-mediated processes such as flavour changing neutral current processes, or FCNC. FCNC transitions from b to s or d, such as B_s^0 mixing and loop mediated hadronic B decays, offer a window to search for new physics sources of charge-parity violation. Equally, rare FCNC processes involving electroweak box and penguin type diagrams, such as $B_s^0 \to \mu^+\mu^-$, and other exotic decays are highly suppressed in the standard model, making them sensitive to new physics. In the charm sector, D meson decays offer a means to investigate flavour changing neutral current processes involving u quarks. Potentially sensitive to different forms of new physics, precise measurements of de-

cays and particle-antiparticle mixing provides complementary constraints on new physics models[6].

Looking beyond flavour physics, LHCb is able to probe the mechanism of heavy quarkonium production and study the spectroscopy of bound states formed by heavy quark-antiquark pairs. With its unique forward coverage (compared to ATLAS and CMS) LHCb can examine electroweak boson production in a different regime, contributing to refining the mass of the W-boson and the sine of the effective electroweak mixing angle for leptons. Various models of new physics feature massive, long lived particles capable of macroscopic distances of flight, resulting in displaced vertices from the interaction point; something LHCb is well equipped to search for[6].

In addition, an internal gas target, originally conceived for novel beam-imaging, enables the detector to operate as a fixed-target experiment; and, with its notable capabilities in the forward region, make a range of unique physics measurements. Opening up the means to explore fixed-target physics with LHC beams, LHCb has a unique opportunity to study the production of particles carrying a large momentum fraction of a target nucleon, make novel probes of nucleon and nuclear structures, and take various measurements of interest to cosmic-ray physics[23,228].

10.2 Detector Design

With its distinctive conical silhouette, LHCb stands apart from the barrel-shaped designs of other large detector experiments around the LHC ring. Given that b- and $\bar{\text{b}}$-hadrons, particles of wide interest to flavour physics, are predominantly produced at small angular deflections from the beam line[5], LHCb has a nominal pseudorapidity range of $2 < \eta < 5$[23], covering approximately $\sim$15 mrad to $\sim$300 mrad in the horizontal bending plane and $\sim$250 mrad in the vertical non-bending plane[5]. This focus on measurements in a limited region allows for greater precision than that practically achievable when seeking to cover the whole angular range. An illustration of the current LHCb detector can be found in Figure 10.1[23].

The various component subdetectors can be roughly divided into the particle tracking system, which charts the passage of particles through the detector, and particle identification systems, whose measurements determine the identity and current properties of a particle. These components do not produce readily prepared measurements, and so it falls to the data acquisition systems to process, interpret and record the signals as they come.

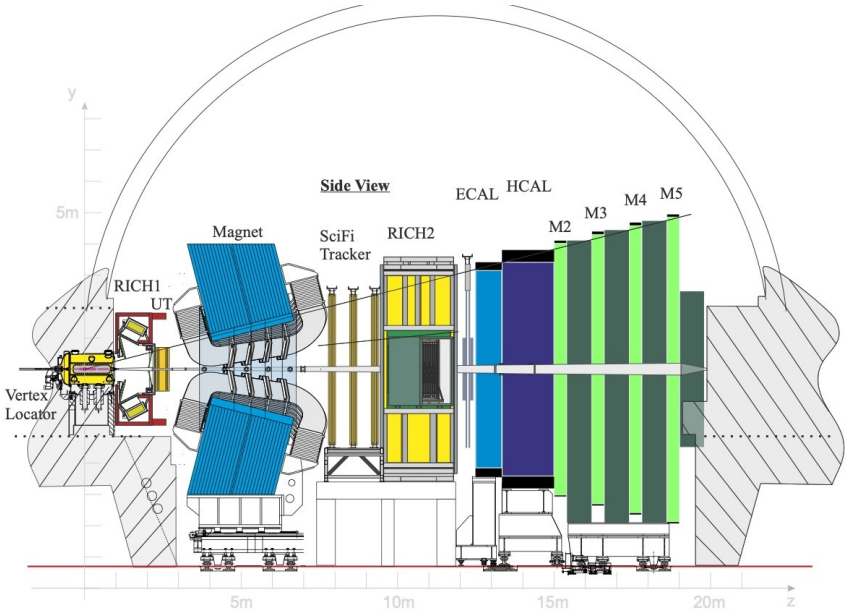


Figure 10.1: Side on representation of the current LHCb detector, after completion of the Phase-I upgrade. The interaction point is located within the vertex locator, or VELO, on the left hand side. Reproduced from [23].

10.2.1 Tracking System, Magnet, and Internal Gas Target

As in many experimental detectors, particles are subjected to a magnetic field, the resulting curved tracks providing a means from which to determine their momenta. To cover all particles produced, the other three large detector experiments at the LHC apply a magnetic field orientated parallel to the beam line, over a wide region encompassing the interaction point, one naturally occupied by various sensitive elements, through the use of solenoid magnets. In contrast, LHCb's forward arm spectrometer design and focused angular acceptance allows for the magnetic field to be applied to a region downstream of the interaction point, while still encompassing those particles within the detector's nominal acceptance. Consequently sensors, such as tracking systems close to the interaction point, can be located outside the magnetic field. This is performed using a dipole magnet of two saddle-shaped coils, mounted above and bellow the beam line, capable of generating a vertical magnetic field of approximately 4 Tm; resulting in curvature in the horizontal, accordingly referred to as the bending, plane. During regular data-taking runs, magnet polarity is periodically reversed, so that data is collected evenly for opposite field configurations[5,23,36].

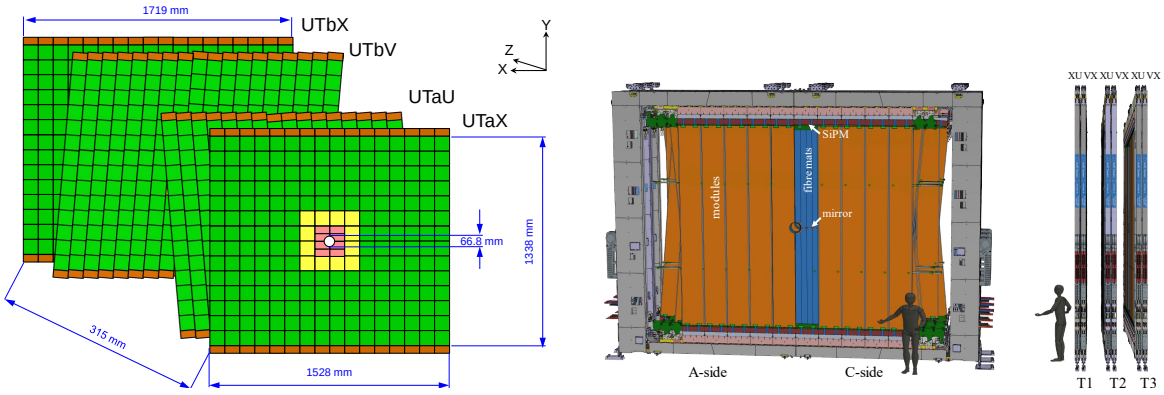
As it makes its way through the detector, a particle's passage is recorded at several

places along its route. The overall tracking system is composed of three installations; the VELO, UT and SciFi trackers. Surrounding the interaction region itself, the vertex locator, or VELO, provides high precision positional measurements in the first moments following a collision, through sequential layers of silicon pixel sensors arranged along the beam line; and is thus ideally placed for locating vertices[23]. Details on the VELO can be found in Chapter 11.

Originally conceived as a means to perform collision luminosity calibrations, the internal gas target and its injection system, SMOG, enables LHCb to operate as a fixed target detector experiment with negligible effect on other LHC activities. Injecting a low rate of noble gas into the centre of an open ended tube construct of heat-treated aluminium within the VELO vacuum vessel, the system produces a localised pressure bump, and a resulting increased beam-gas collision rate. With a capacity for multiple different gases of differing nuclear sizes, such as Helium, Neon and Argon, the system not only allows for making precise beam density profile measurements, but enables exploration of the range of opportunities offered by colliding LHC beams with a fixed target[23,228].

Sitting immediately before, or upstream of, the magnet, lies the second stage of the tracking system; the upstream tracker, or UT. As illustrated in Figure 10.2(a), four planes of vertically arranged silicon microstrip detectors are mounted across two stations, with the inner two planes inclined, in opposite inclinations, by 5°; enabling calculation of the vertical coordinate of each hit without ambiguity. 187.5 µm pitch sensors cover the majority of each plane's surface, with special half pitch sensors arranged around the innermost areas to maximise the active area near the beamline. The planes of each station are nominally spaced 55 mm apart, with 205 mm between the two innermost planes[23,229]. Through tracking particles as they enter the magnet region, the UT plays a significant role in identifying duplicate particle tracks, improving momentum resolution, and is vital for reconstructing long lived particles that decayed outside of the VELO[36,230].

On the far side of the magnet is the SciFi, LHCb's Scintillating Fibre tracker. Measuring particle trajectories after their passage through the magnetic field, this subdetector provides the means to determine their momentum. Twelve detection planes are arranged into three stations of four planes, each station following a similar pattern to the UT, with the middle two layers of each station again tilted at 5° in opposite rotation to one another. Each plane's active area is formed by fibre mats of six layers of densely packed, 250 µm diameter, blue-emitting scintillating fibres. Stations are separated approximately 700 mm apart[23,36,231]. A 3-dimensional render



(**a**) (**b**)

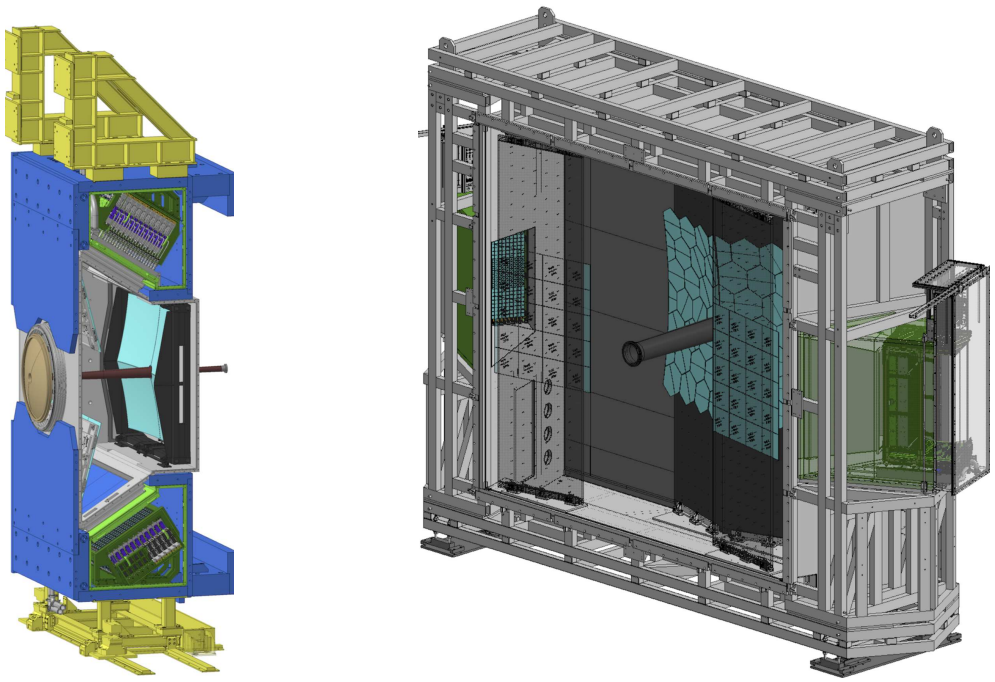
Figure 10.2: (a) Diagrammatic overview of the UT tracker as viewed from the downstream direction, showing the four layer arrangement including tilted layers. Green indicates regions using the regular sensors, while yellow and pink indicate those regions near the beamline using specially designed sensors with half pitch. Reproduced from [229]. (b) A 3-dimensional render of the SciFi tracker, shown both from the from the downstream direction (left) and side on (right). Reproduced from [23].

of the SciFi tracker is shown in Figure 10.2(b).

10.2.2 Particle Identification Systems

In the LHCb detector, differentiation between pions, kaons and protons is achieved through a pair of ring imaging Cherenkov detectors, RICH1 and RICH2 [23,232]. As a charged particle traverses through a dielectric medium, if its velocity exceeds that of light in said medium, electromagnetic radiation is emitted, overlapping waveforms forming a characteristic cone-like wavefront, in a process known as the Cherenkov effect [7]. Both RICH detectors exploit this process through use of fluorocarbon gas volumes; the resulting photons then focused through a system of spherical and planar mirrors onto detector planes formed of multi-anode photomultiplier tubes, positioned outside the detectors acceptance to avoid influencing the onwards passage of particles [23,232]. The radiation's cone-like wavefront results in ring images, from which the cone's opening angle, and so the particles velocity, can be determined; proving a means to relate a particles momentum and mass [7,23].

Located between the VELO and UT trackers, RICH1 utilises C_4F_{10} gas and covers an angular acceptance of 25 to 300 mrad in the horizontal (25 to 250 mrad in the vertical) plane, enabling it to identify particles in the 2.6 to 60 GeV/c momen-



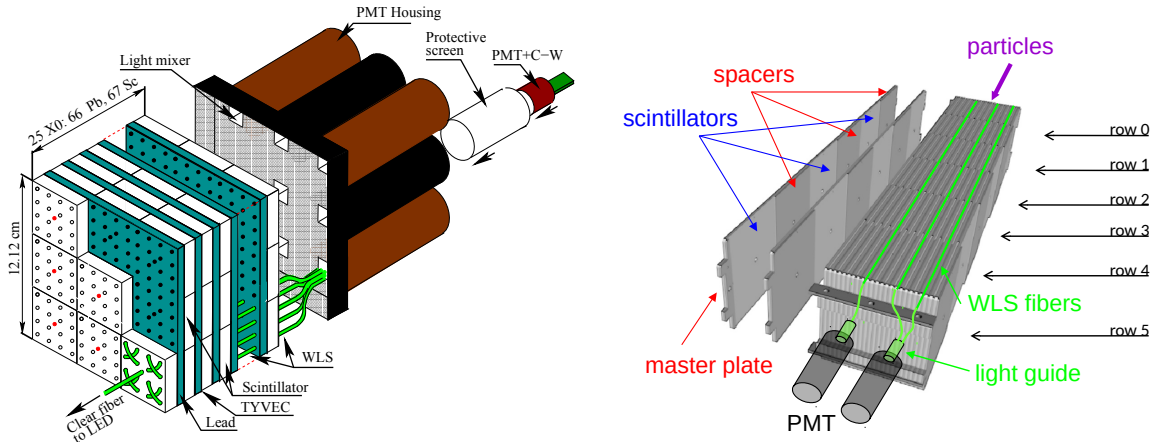
(**a**) (**b**)

Figure 10.3: A pair of 3-dimensional renders illustrating the (a) RICH1 and (b) RICH2 detectors after the Phase-I upgrade. Both reproduced from [233].

tum range. Positioned instead after the SciFi tracker, and so the magnet, the larger RICH2 subdetector provides identification for higher momentum particles in the 15 to 100 GeV/c range, using CF_4 gas and with a 15 to 120 mrad angular acceptance in the horizontal bending (15 to 100 mrad in the vertical non-bending) plane[23]. Illustrations of RICH1 and RICH2 can be found in Figure 10.3(a) and (b) respectively.

RICH2 is followed by the detector's calorimeter system, which determines the energy of a range of particles, and uses a classical electromagnetic calorimeter followed by hadronic calorimeter structure[23]. As particles impact material in the relevant calorimeter, they trigger a cascade of reactions, resulting in a shower of secondary particles across which their energy is distributed. These particle showers are stopped within the subdetector, producing scintillation light from whose measurement, using photomultiplier tubes, the original particles energy may be reconstructed[7,23]. Being a destructive method of measurement, the calorimeter system marks the end of many particle's journeys through the detector, and so lies near the end of overall detector; the front surface of the first calorimeter, the ECAL, located around 12.5 m from the interaction point[23].

As an electromagnetic calorimeter, the ECAL concerns itself with particles that

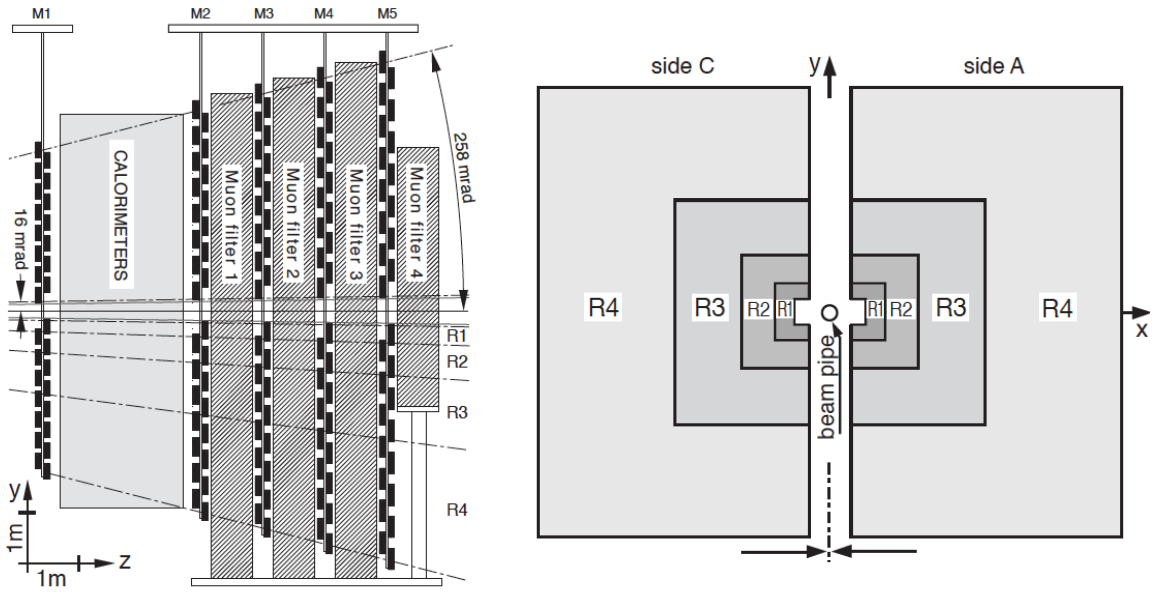


(a) (b)

Figure 10.4: Diagrams illustrating the compositions of (a) an ECAL and (b) a HCAL cell, tiled to form the respective calorimeters. Both reproduced from [234].

interact with the electromagnetic force, such as electrons, positrons and photons. Electromagnetic showers are provoked by 2 mm lead sheets, interspersed by 4 mm plastic scintillator, and threaded with wavelength-shifting fibre cables to collect and deliver scintillation light to the photomultiplier tubes[23,36]. An illustration of a ECAL cell can be seen in Figure 10.4(a). With the particle density significantly higher closer to the beam line, the ECAL is divided into three regions, with progressively larger cells employed further from the beam line[23,234]. The HCAL, as a hadronic calorimeter, conversely covers protons, kaons and other hadronic particles that interact by the strong nuclear force. As shown in Figure 10.4(b), staggered iron and plastic scintillator tiles are here arranged parallel to the beam line for improved light collection, and again collected by fibre cables[23,234]. Given the typical spread of hadronic showers, HCAL has a comparatively larger granularity compared to the ECAL, and is similarly divided into two regions with larger cells employed away from the beam line[23]. With the importance placed on electrons and photons energy resolution, the ECAL has a thickness covering 25 radiation lengths; while the HCAL, due to space limitations, is limited to 5.6 nuclear interaction lengths[7,23].

While electrons lie within the domain of the ECAL, owing to comparatively suppressed bremsstrahlung, muons do not develop electromagnetic showers as electrons do. Equally, muons do not experience the strong nuclear force, allowing them to pass through hadron-absorbing material such as the tiles of the HCAL[7]. Though it leaves muons outside the reach of the calorimeter system, this penetrating power



(**a**) (**b**)

Figure 10.5: (a) Side on diagram of the muon system, showing the arrangement MWPC's and thick iron filters. Note that this diagram specifically corresponds to the pre Phase-I upgrade detector. Aside from the removal of the pre calorimeter M1 station and introduction of additional shielding around the beam pipe, the detector's overall composition as shown is largely unchanged; with the upgrade program otherwise focusing on the electronics and readout systems[23,232]. (b) Diagram illustrating the four regions, as viewed from the downstream direction, with granularity increasing as regions are located further from the beam pipe. The region labels R1-4 correspond to the arcs indicated in (a). Both reproduced from [232].

is instead leveraged to provide robust muon identification[7,232]. Located after the calorimeter system, where few other particles will reach, LHCb employs a muon identification system comprising of four multi–wire proportional chambers, separated by thick, 80 cm iron walls to filter low energy particles[23,232]. Each is composed of four independent layers, consisting of anode wires between a pair of cathode plates, and act to pick up the passage of those particles that have penetrated that far through the detector. Diagrams of the muon system can be seen in Figure 10.5. As particle flux is higher closer to the beam line, each chamber is divided into four regions of differing granularity, again increasing further from the beam line to even out particle flux and channel occupancy[23,232].

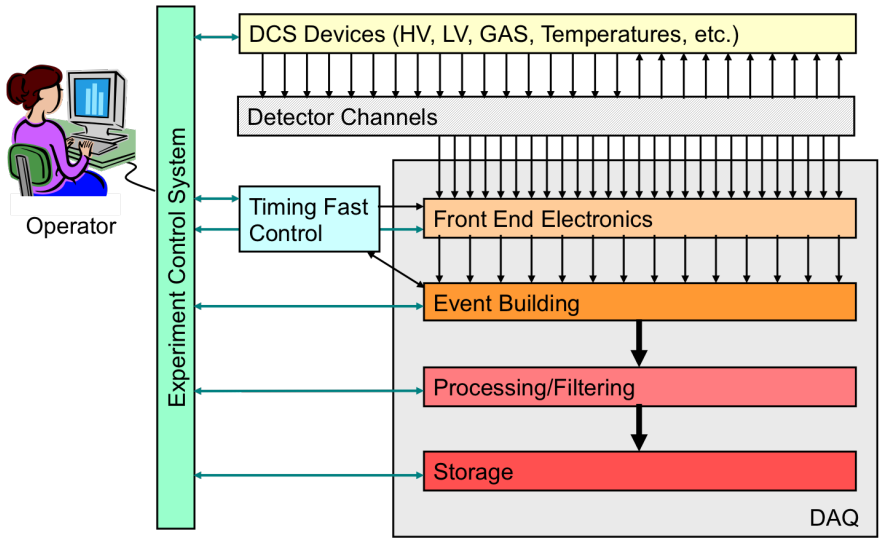


Figure 10.6: An illustration outlining how the LHCb readout chain is managed, including the Experiment Control System and Timing and Fast Control system. Reproduced from [23].

10.2.3 Control Systems

The 'brain' of the experiment if you will, centralised control systems manage and operate the many components of the detector in real time. A rough outline of the LHCb readout chain and its management is shown in Figure 10.6.

The Experiment Control System, or ECS, manages the configuration, monitoring and control of the experiment. As the system with which operators interact, it provides a coherent interface for manage all detector equipment, including trigger system; and is built on the joint control project, or JCOP, a common framework for detectors at the LHC[23,235]. The many readout elements that make up the LHCb detector are grouped in partitions, each of which may represent a part of a subdetector, a subdetector in its entirety, or a group of subdetectors. These partitions are independent of one another, allowing them to be controlled and operated separately[23].

Responsible for clock, timing and readout management, the Timing and Fast Control system, or TFC, keeps the experiment running in synchronous. The system handles generating and distributing signals from the experiment master reference clock throughout the detector. It also manages the flow of data through the entire readout chain to ensure coherent data taking across all readout elements; issuing commands to control processing within readout and front end electronics, along with

other calibration and subdetector-specific commands to detector electronics[23,36,235].

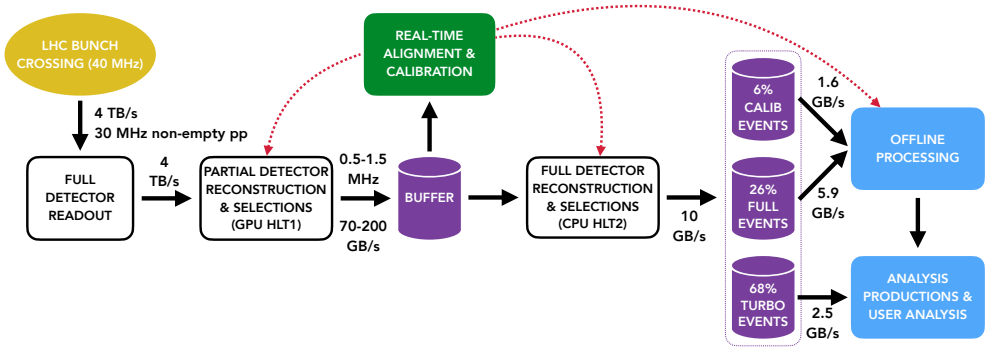
10.3 Data Acquisition, Trigger and Analysis

While traditional trigger approaches look for generic signatures that don't require significant analysis to identify, increasing event rates and the broad range of potentially interesting events involved in the LHCb physics program mean it is not possible to sufficiently reduce events in this way. Instead, to reduce events to recordable levels it is necessary to first reconstruct an event online in order to identify physics signals of interest. Though demanding, this has allowed the experiment to, where possible, develop a novel system where only a limited amount of information concerning the less interesting remainder of an event is propagated onwards to final storage[23,236].

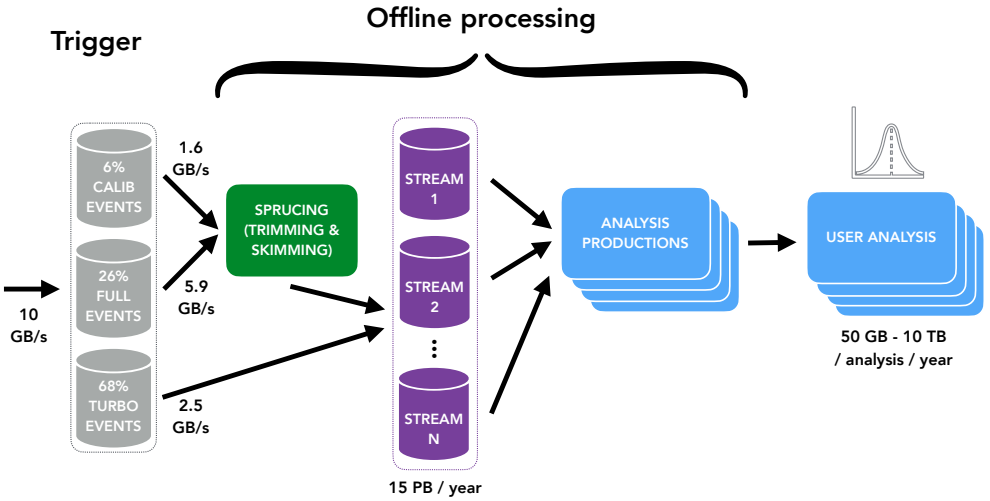
Therefore in order to achieve a readout of 40 Hz, LHCb has since 2022 operated a fully software event filtering system, using this real time analysis and selective persistence approach, to reduce incoming data down to a manageable 10 GB per second; including the demanding task of performing full offline quality reconstruction in real time[23,237]. Unlike the hardware stage of the previous trigger system, the new first selection stage notably now has access to tracking information with which to make its selections, as many signals of interest to the LHCb physics programs can be distinguished using momentum direction and vertex information[29]. Online processing has been designed so that any offline reconstruction or selection is a reconfiguration of the same algorithms; therefore, as of commencing data taking Run 3, no separate offline repetition of the reconstruction carried out online is foreseen, and LHCb offline computing is consequently dominated by simulation[23]. While the nature of this system, from the point of view of the front end electronics, means the detector can be justifiably considered 'trigger-less', in practice the event selection system is still referred to as the trigger, for convenience and as a continuation of naming schemes used by the previous system[23,237].

A representation of the full LHCb data processing pipeline can be found in Figure 10.7, with the online data acquisition and offline processing stages detailed in Figure 10.7(a) and (b) respectively. The LHCb codebase is largely written in C++, for data processing and algorithms, and python, for job configuration.[29]. The online trigger system incorporates two reconstruction and selection stages, HLT1 and HLT2 *. The experiment CPU codebase is built on the experiment wide Gaudi[240]

*HLT refers to High Lever Trigger. This naming convention is a legacy of the previous trigger



(a)



(b)

Figure 10.7: Illustrations of the LHCb dataflow from 2022 onwards, focusing on the (a) online or (b) offline stages. Reproduced from [238]. Indicated dataflow values in turn from [237] and [239].

framework, with HLT2 and sprucing handled by the Moore[241] application. HLT1 operates on GPU, implemented in CUDA within the Allen[242] framework[23,29].

10.3.1 Track Categorisation

Before proceeding, it is worth summarising how tracks within the LHCb detector are categorised. Tracks are divided into five types based on which trackers a given track has a presence in.

- Velo tracks are those tracks which appear only within the VELO, and so in-

system, which incorporated a Low Level Trigger stage.

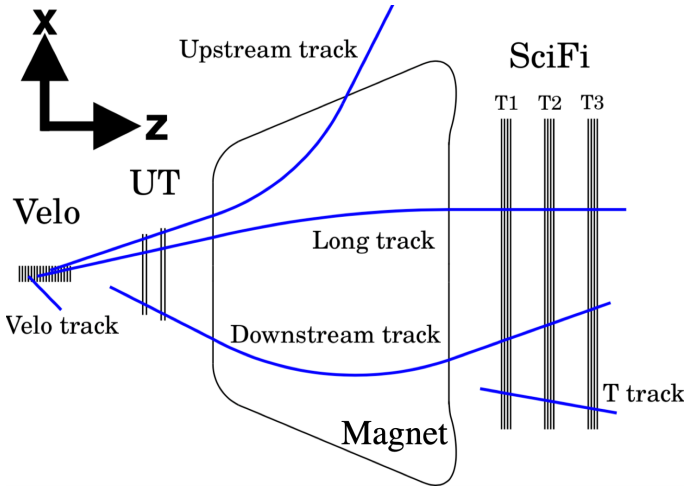


Figure 10.8: Categorisation of track types, depicted in the bending plane. Reproduced from [23], itself based on a diagram in [218].

cludes tracks outside the nominal LHCb angular acceptance and those corresponding to short-lived particles that do not reach the UT.

- Upstream tracks appear in both the VELO and UT trackers, but do not reach the SciFi.
- Long tracks pass through all three trackers, and as such are typically those for which the most accurate momentum determination can be made, and therefore are the most used for physics analysis.
- Downstream tracks are present in the UT and Scifi trackers, such as those originating from secondary vertices between the VELO and UT.
- T tracks are those that appear only in the SciFi, and are so called as a legacy of the previous TT tracking system.

An illustration of the above track types can be found in Figure 10.8[23,218,243].

10.3.2 Event Building and HLT1

As measurements are made, signals from the various subdetectors are dispatched to the dedicated Event Builder server farm, located in a containerised data centre above ground. Here, as the name suggests, the event building process is performed, in which data corresponding to each specific bunch crossing event is assembled together. Data corresponding to empty crossings are normally dropped at this stage, and a global cut is used to remove events featuring a very large quantity of tracks, as

these would both use a disproportionate amount of resources to reconstruct and the reconstruction quality is likely to be worse[23,36,244].

Once data from all sources has been received, an event is transferred on to be processed by HLT1, which runs on GPUs within the Event Builder farm. The first of two principal filtering stages, HLT1, performs partial reconstruction, focusing on long tracks, in order to perform an initial filtering pass. Though reconstruction including information beyond the trackers would allow for more precise selection, reducing calculations is very important at this stage, and a large efficiency can be achieved from examining tracks alone[23,244].

Beginning with the reconstruction of tracks and primary vertices within the VELO, tracks are subsequently extrapolated through the UT and SciFi Trackers, including accounting for deflection by the magnet; enabling particles' momenta to be deduced. A simplified Kalman filter method is applied to high momentum tracks to estimate details on them when they were near the beam line, and in turn better pinpoint where they originated. Muons among the particles are then identified, and finally tracks are fitted to common origin points to form displaced secondary vertex candidates[23,242,244].

Selection applied at this point are designed to reduce the number of events by roughly a factor of 20, a level at which full quality event reconstruction can feasibly take place. Designed to be inclusive, criteria are primarily driven by identifying signatures desirable by the bulk of the LHCb physics program, along with additional selections for other signatures that might otherwise not be picked by the principal selections, and to select events for technical and calibration purposes[23,242,244].

10.3.3 Calibration Buffer and HLT2

Successful events from HLT1 are subsequently transferred to the Event Filter Farm, or EFF, a large farm of general purpose x86 CPU servers located in the same data centre, which while the LHC is not taking data can be utilised for other computing tasks[23,36].

In order to perform event reconstruction, accurate calibration and alignment information is required. Thus events are sampled from the buffer to perform real time calibration and alignment studies. Parameters important to HLT1 are relatively quick to compute, and are updated as soon as possible. However while using old parameters is acceptable for a first reduction cut, it is not for detailed reconstruction, and computing the complete range of parameters necessary takes time. Events are therefore held in a large intermediary storage buffer as their corresponding calcula-

tions are completed. This also allows events to be buffered during event production and caught up on between LHC fills, spreading out processing to ease computing requirements[23].

Once prepared, events proceed to HLT2 and full offline-quality reconstruction. Similar to in HLT1, this commences with track reconstruction, utilising separate algorithms for different track types[23,237,243]. Long tracks, the most used in physics analysis, are reconstructed by two independent algorithms. Each takes a different approach; one extrapolating velo tracks into the UT and SciFi and picking up hits directly, the other matching already found tracks within the SciFi, themselves reconstructed with a hybrid seeding algorithm, to velo tracks and potential intermediary UT hits[23,243]. Said SciFi tracks are also extrapolated backwards to the UT in order to construct downstream tracks. Likely duplicate tracks sharing segments in a given tracker are analysed and pruned by individual pattern recognition algorithms[23,237].

The properties of charged particle trajectories are determined using a Kalman fit method. For most events, interactions with the detector material are parametrised, while a more detailed method, employing interaction tables for magnetic field and material distribution, is used for those events destined for alignment or analyses requiring enhanced precision. A global algorithm then seeks out and removes any overlapping tracks and other remaining duplicates[23].

Progressing to particle identification, separate reconstruction of ECAL[23], RICH detector[245] and muon system[246] measurements are performed; and with a combination of these results, identification of electrons, muons, pions, kaons and protons among the detected particles. Tau leptons and neutral particles decaying in the detector are treated as composite particles. This identification process is performed by multiple multivariate classifiers, tuned for different kinematic regions, and the choice which specific algorithm informs a particular selection lies with the relevant analyses[23].

Each reconstructed event is scrutinised by approximately one thousand different algorithms to identify whether it is of interest to retain for the various LHCb physics analyses, or for other purposes such as calibration. At the same time, these algorithms propose what set of information to retain on a selected event, and in accordance with the selective persistence approach used, the union of requested data is recorded[23,237]. Where successful, events are grouped based on the underlying physics present or event information recorded; and all necessary data is consolidated into files for storage on magnetic tape. All events are associated to one of three storage streams, depending on selecting algorithms and the data they request be

recorded. Where reduced information is retained, events are allocated to the Turbo stream, while those events stored in their entirety go to the FULL stream. Events for calibration are stored as the TurCal stream [23,247].

10.3.4 Offline Processing and Analysis

With most data processing activities performed online, offline processing consists largely of final data preparation for physics analysis, including minimising the amount of data analysis will need to access in order to be carried out. Known as sprucing, the process shares many algorithms and tools with HLT2 and principally consists of skimming, slimming, and streaming tasks. Where required, skimming applies further selections to events in the FULL stream, in order to reduce their number, while slimming reduces the amount of information stored for a particular event. Compared to selection within HLT2, the more relaxed limits on computing time allows for selections unachievable on an online timescale, such as those dependent on analysis of complex cascade or many-particle final state decays. Regardless of whether skimming or slimming was performed, a given FULL or Turbo event is then streamed according to its physics content, along with the creation of accompanying metadata files. The final data is then formatted and saved to disc, from which the datasets used by physics analyses are centrally produced. Sprucing is performed concurrently to data taking, and re-sprucing campaigns are also performed periodically [23,247].

10.4 Upgrade Programs

Despite successfully operating at approximately $\mathcal{L} = 4 \times 10^{32}$ cm^{-2}s^{-1}, twice the instantaneous luminosity for which it was originally designed, various measurements studied in LHCb's physics program are still constrained by statistical uncertainties [23]. Dubbed the Phase-I Upgrade, the 2018 to 2022 shutdown period therefore saw the LHCb detector undergo a extensive upgrade, enabling the detector to operate at a instantaneous luminosity a further factor of five higher and an increased bunch crossing rate of 40 MHz [6,23,229,232]. Central to this is the evolution of the previous readout and online systems, with replacement of the previous hybrid hardware and software triggers by the new flexible software based system; incorporating real-time offline quality reconstruction for event selection while building on the selective persistence approach of the previous system. This overhaul of the data systems

naturally involved improvements across the board, such as optimisation of reconstruction algorithms, implementation of HLT1 on GPU hardware, and migration of underlying software to operate on the latest C++ version[23,237,244].

In rising to meet the challenges posed by increased luminosity and readout, the Phase-I Upgrade also involved upgrades to all component subdetectors, representing a replacement of more than 90% of the active detector channels and including complete replacement of the tracking system[23,229,247]. The VELO was replaced by a new system, it's original silicon strip design exchanged for a silicon pixel system. Meanwhile, the previous Tracker Turicensis detector, or TT, had its duties taken over by two new system, with the UT replacing the first TT station, and SciFi the remaining three[23,229]. The internal gas target injection system saw upgrades focusing on its expanded role in use as a fixed-target experiment, particularly the introduction of a dedicated construct within which the gas is injected, tightening the achievable density to a more localised area[23,228]. Along with a redesign of RICH1's optics, the RICH system saw a new photon detection system, with its hybrid photon detector replaced with multi-anode photomultiplier tubes and accompanying front end electronics. The calorimeter system saw a comprehensive redesign and replacement of both front end and readout electronics; and having principally served the original hardware trigger, the Scintillating Pad Detector, or SPD, and the PreShower, or PS, calorimeter systems were removed. The muon system received new readout electronics, along with introduction of additional shielding around the beam pipe. Similarly to the SPD and PS systems, the muon M1 station largely provided for the previous hardware trigger, and so was removed[23,232].

As mentioned in Section 9.2.5, in the coming years the LHC will itself play host to a transformative upgrade to push its potential luminosity output even further, revitalising the accelerator as HL-LHC. Therefore work is underway towards a new LHCb upgrade program, the Phase-II Upgrade, in order to take advantage of this ambition program and address the challenges posed by the accelerators high luminosity phase[6,227]. As part of this, significant preparatory consolidation and enhancement work, dubbed Upgrade Ib, is planned for the 2026 to 2029 shutdown period[248].

Chapter 11

The LHCb Vertex Locator

11.1 Role

Positioned at the tip of the LHCb detector, the VELO, or vertex locator system, provides precision tracking in the immediate vicinity of the proton-proton interaction region, and is typically the first sensitive detector particles encounter. With sensitive elements sitting as little as 5.1 mm from the proton beams during data taking, the VELO is specifically designed with precise resolution of primary and secondary vertices in mind[23,249]. Displaced secondary vertices are a key signature of b- and c- hadron decays, and so of crucial importance to event selection for the LHCb physics program. Reconstruction of tracks within the VELO is among the first steps performed during both HLT1 and HLT2 stages, seeding further track reconstruction across the detector[23,30,36]. Though LHCb as a whole has a nominal $2 < \eta < 5$ acceptance, from its position surrounding the interaction point, the VELO is not limited to the forward region, and so incorporates sensitive elements both up and downstream; with additional tracks used to improve measurement of primary vertices[249].

11.2 Overarching Design

LHCb uses a global coordinate system, which places the origin at the nominal interaction point, with the positive z-axis extending along the LHC beam-line into the forward region accepted by the detector. The positive y-axis is orientated vertically, and the x-axis horizontally with direction following standard conventions[250]. This coordinate system can be observed superimposed in Figure 10.1, with the origin lying within the VELO and the positive z-axis extending towards the Muon system.

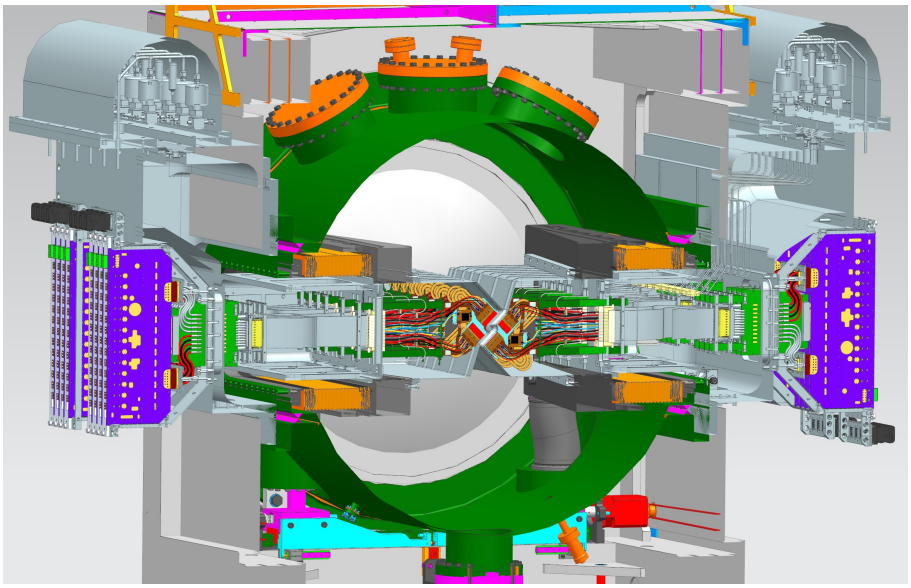


Figure 11.1: A 3-dimensional model of the overall VELO detector, post Phase-I upgrade, within the vacuum vessel. Reproduced from [249].

In addition, when facing down the beam-line into the detector, the left, $x > 0$, and right, $x > 0$, hand sides are conventionally referred to as the A and C sides respectively*, with side C lying towards the inside of the LHC ring. Many parts of the detector, VELO included, use to this nomenclature for components[23,250,251].

A render of the wider VELO is shown in Figure 11.1, with key components of the inner portion illustrated in Figure 11.2. The detector consists of 52 identical 'L' shaped modules, with correspondingly-shaped sensitive areas, arranged in pairs across 26 stations spaced along the beam line. Each pair together forms an approximate square, orientated perpendicular to the LHC beam line, with the beams themselves passing through the central aperture formed between them[23,249,251]. Stations are arranged to ensure that 99% of tracks originating within $\pm 2\delta$ lumi of the nominal interaction point, and within the detector's $2 < \eta < 5$ acceptance, will be measured by four stations[251]. To protect the sensors while still enabling them to be placed as close as possible to the beam-line, modules are mounted as two independently retractable halves, known as Side A and Side C. These halves are retracted during the fill procedure to 30 mm until the circulating beams have stabilised, before returning to the closed position for data taking[249,251]. The approximate squares are orientated at 45° about the z-axis, a module forming each side of the resulting diamond shape, in order to minimise collision risks during installation. In addition, this

*This convention owes itself to the detector's practical placement within the detector cavern, and is in reference to 'access' and 'cryogenics' sides[23].

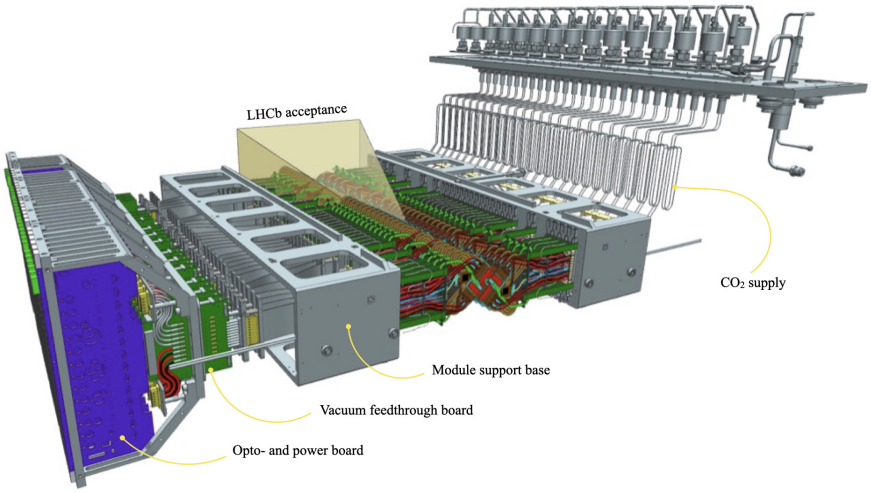


Figure 11.2: An illustration of the two VELO halves, depicting the modules and their supporting structure, corresponding to the inner portion of Figure 11.1. The electronics are included on the left side, while the right side instead shows the CO_2 coolant system. The RF foils are not shown, and the LHCb nominal acceptance region is indicated by the transparent pyramid. Reproduced from [23].

orientation is advantageous should the detector need to be operated with the halves not fully closed[23,252].

Each half is encased by a thin aluminium alloy enclosure known as the RF foil, corrugated in a stepped design following the modules, which acts to shield detector electronics from beam induced currents[249,251]. In order to maintain structural integrity while keeping material to a minimum, so as to reduce scattering, the foils are fabricated from solid blocks, painstakingly machined down to a typical thickness of 250 ± 100 µm[23,36,251].

The overall detector resides within a 1.4 m long by 1.1 m diameter vacuum vessel integrated into the LHC beam pipe and which houses the primary beam vacuum, with the detector assemblies entering through apertures on either side. Wakefield suppressors are installed at both ends of the vessel. Modules themselves operate within a secondary vacuum, separated by the RF foil together with a bellows system to accommodate for motion; while a pair of hoods seal the secondary vacuum from the external atmosphere[23,249,251].

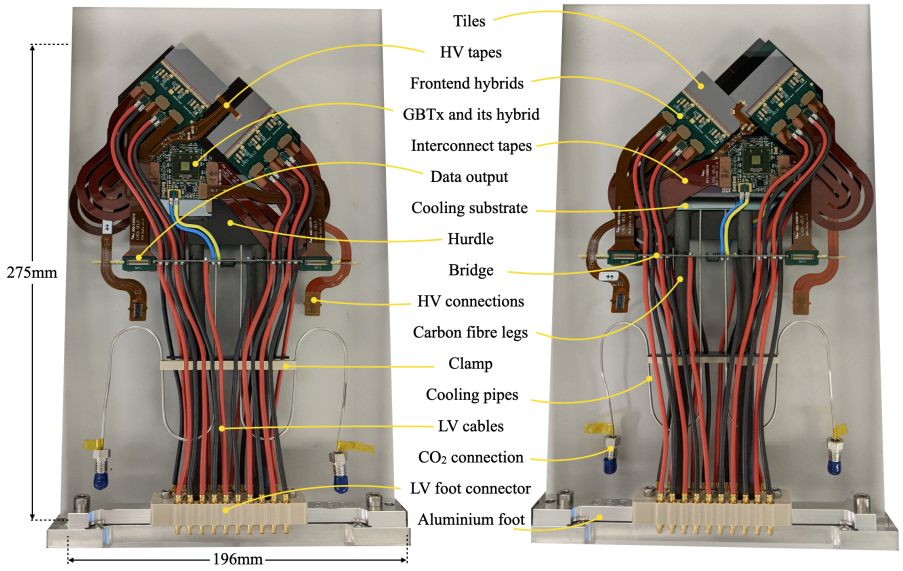
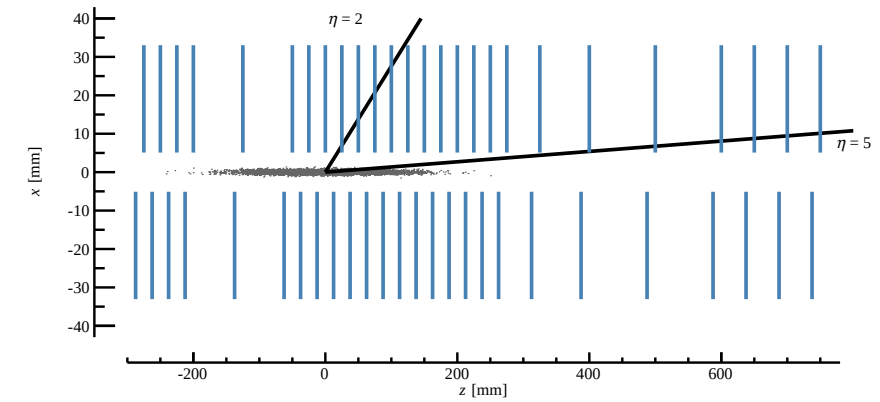


Figure 11.3: Photographs of a fully-assembled VELO module showing the (left) connector, as viewed from the upstream, increasing z direction, and (right) non-connector, as viewed from the downstream, decreasing z direction, sides. Various components are labelled. Reproduced from [249].

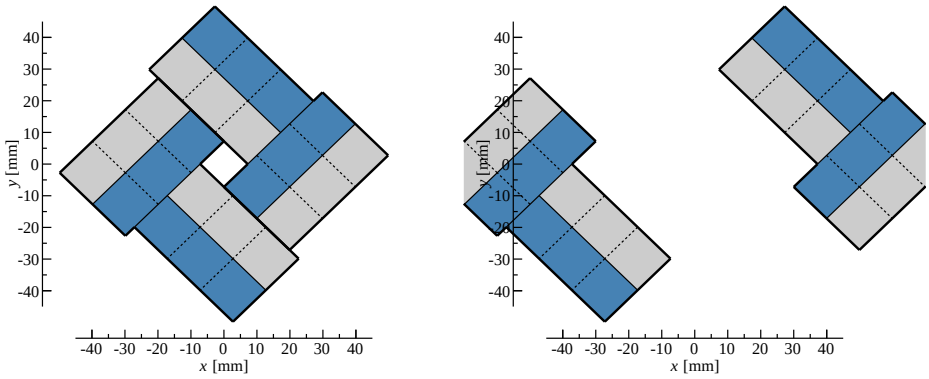
11.3 VELO Pixel Modules

An individual VELO pixel module is pictured in Figure 11.3. Each of the 52 modules is composed of a microchannel plate bearing four 14.080 mm by 42.570 mm active area sensor tiles, two mounted on either side so as to form the overall 'L' shape acceptance.[249,250]. In order to minimise the material present near the interaction region, the innermost tiles protrude from the plate by 5 mm, and to catch highly inclined tracks, tiles on opposite sides are displaced such that they overlap by 110 µm. Due to mechanical limitations however, there is a small unavoidable gap in acceptance between the outermost tiles[251,252]. The arrangement of tiles is illustrated in Figures 11.4(b) and 11.5.

This assembly is affixed, through an invar cooling connector, to a mechanical support known as the hurdle; and in turn mounted to the module support base using an aluminium foot[249]. Spread over just over 1 m, modules are arranged along each side with a tightly spaced central region, along with forward and backward region groupings, and a minimum spacing of 25 mm for modules on the same side. The layout is identical on each side, save that modules on Side A are displaced in z by +12.5 mm relative to Side C. This allows so that when in the closed



(**a**)



(**b**)

Figure 11.4: An overview of the VELO module setup. (a) Side on cross section at $y = 0$ showing the LHCb VELO module locations, with the interaction region and boundaries of LHCb's nominal acceptance region indicated. (b) Schematic of the sensitive region of a station in the x-y plane, in both the closed (left) and open (right) states. Reproduced from [250].

position, modules corresponding to the same station overlap by 200 µm to ensure coverage[23,249,251,252]. Modules are labelled 0-51, with module 51 both at largest z and farthest from the nominal interaction region[250]. The arrangement of modules is depicted in Figure 11.4(a), and is optimised based on simulation with spacings regularised to multiples of 12.5 mm for sake of the RF foil[252]. A table of module positions and numberings is provided in Appendix B and a detailed summary of module geometry can be found in [250].

A single tile consists of a planar silicon pixel sensor bump-bonded to three bespoke pixelated ASIC chips in line, which provide signal processing and digitisation[23,23,249]. Each sensor is divided into 768×256 square pixels of 55 µm pitch, with an additional

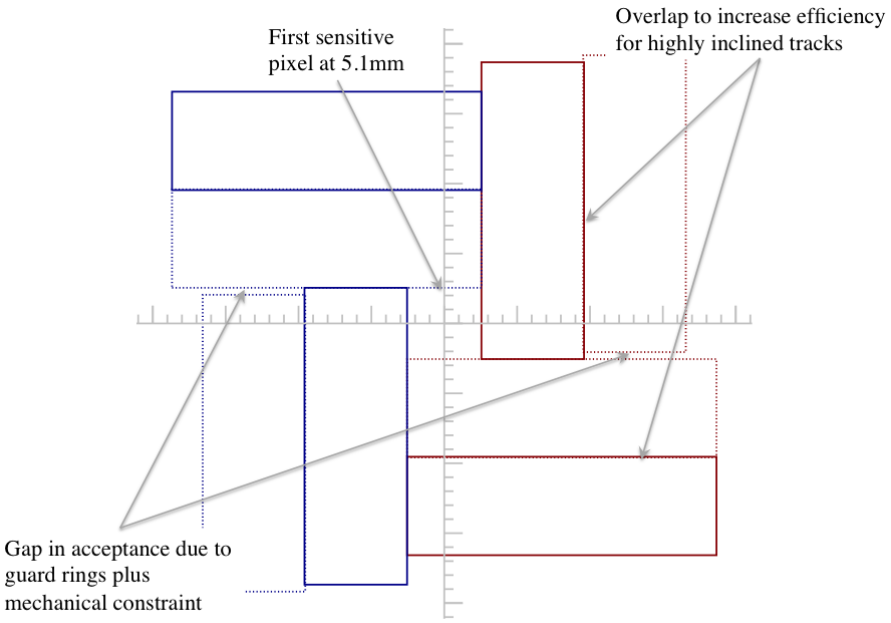


Figure 11.5: Positioning of tiles active areas for a pair of modules, in the closed position, in the x-y plane. Tiles located on the back side of a module are depicted with dotted outlines, tiles corresponding to the same module are depicted in red or blue respectively. Note that the axes shown do not correspond to the positioning of the x-y axes of the wider coordinate scheme detailed above. Reproduced from [251].

row on the outer side acting to tie the ASIC ground to the innermost sensor guard ring. Arranging the ASICs together in a line results in a 165 µm gap between the 256×256 pixel matrices corresponding to each ASIC; therefore pixels bordering the inter-chip regions are elongated to 137.5 µm to provide coverage[249,250].

Thermal cooling is achieved through an evaporative CO_2 cooling system, the liquid-vapor mixture circulating through 200 µm wide by 120 µm deep microchannels within the plate to which the active components are affixed. Channels interface with the wider coolant system through the invar cooling connector, with the local CO_2 itself located in a separate tertiary vacuum volume. To achieve an even coolant distribution, the first few centimetres of ingoing channels narrow to 60 µm by 60 µm, while the hurdle acts to thermally insulate modules from their mounting point[23,249].

The ASIC on each module side are managed by a pair of frontend hybrids, one serving each 3 ASIC tile, while a single accompanying GBTx hybrid per side decodes and distributes control and timing signals to each frontend hybrid pair. Placed outside the vacuum vessel, the opto- and power-board, or OPB, acts as the interface between modules and the wider off-detector electronics systems; transmitting data

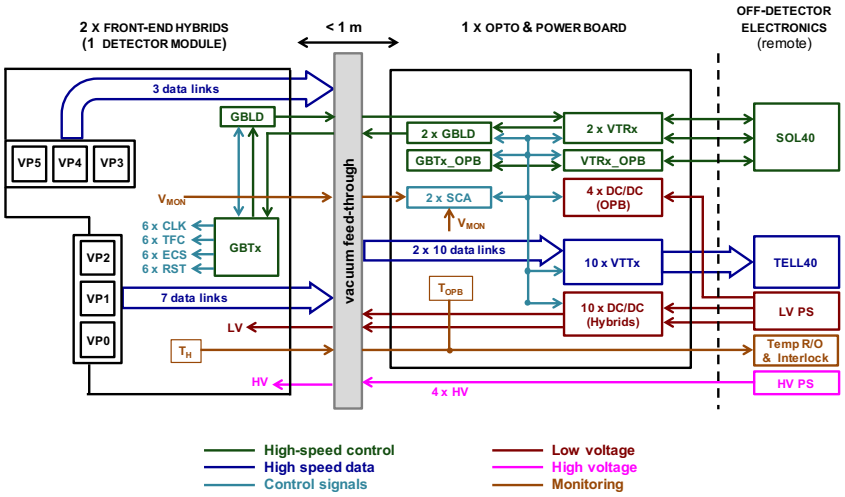


Figure 11.6: Diagram outlining the main parts of the VELO electronics system. Reproduced from [23].

and receiving control signals beyond the detector through optical fibres, while signals and voltages are routed across the vacuum barrier through a custom vacuum feedthrough board. Both high and low voltage supplies are received from elsewhere in the LHCb cavern, with the OPB converting and distributing the low supply voltage[23,249]. An overview of the electronics systems is depicted in Figure 11.6.

11.4 Track Reconstruction within the VELO

11.4.1 Development of the Pattern Recognition Algorithm

The first VELO pattern recognition algorithm was developed in 2002[253], and went on to see further tuning in 2004[254] and 2007[255]. Taking a track following approach, it was designed in context of the previous VELO detector geometry, which utilised a system of silicon strips arranged as concentric ring and radial strips around the beam-line, with each station similarly composed by two (in this case semi-circular) halves[29,253]. After clustering, a 2-dimensional search was performed in the r-z plane, hunting for potential triplets of three hits across consecutive stations based on slope and alignment. These seed tracks were then extended onwards to following stations, picking up additional hits to form track candidates, before tracking with corresponding angular measurements, and finally, fitting using a Kalman filter[29,253,255]. As the detector looked to push beyond its original design luminosity, and to accommodate unforeseen issues related to VELO positioning, a modified algorithm was introduced in 2011[256]; this revised method looking instead for quadruplets

of hits in the r-z plane, before seeking triplets among those left unused[29,256].

With the new VELO pixel detector and software only trigger system on the horizon, efforts began on redesigning the algorithm for the post Upgrade-I era, with the first version implemented in 2012[257]. Now working in Cartesian coordinates and with 3-dimensions from the beginning, it followed a similar approach to the previous method; only instead considering pairs of unused hits, then extending them upstream[29,257]. Representing a more radical departure, an alternative approach using a Hough transform method inspired by straight line recognition within the eye was also investigated around the same time[34,35]. Returning to look for triplets across neighbouring stations, a new local search algorithm designed with parallelisation and GPU architectures in mind was presented in 2014[258], with further improvements over the coming years[259]. An initial baseline tracking implementation for HLT1, featuring various execution time improvements, was introduced in 2018[29,260].

Presented in [30], with further discussion in [261], the current pattern recognition implementation continues on the methodology of previous algorithms. A revised search by triplet algorithm implemented in the Allen framework, it is designed with parallelisation in mind and with features to reduce combinatorics and track overlapping[30,242,244,261]. The algorithm was designed for use with the SIMT programming model of CUDA, for use on GPU, and is implemented in the HLT1 trigger stage. An implementation for CPU using the SPMD programming model, detailed in [29], has also been developed[261,262].

In what follows, we will focus on providing an outline of track reconstruction for HLT1 during Run 3. Currently, HLT2 reuses VELO tracks as found in HLT1, though other approaches are implemented. LHCb software is constantly evolving, and the following is predominantly based on the algorithm as described publications. It may therefore not describe the algorithms used for a given day.

11.4.2 Clustering

Given that most VELO pixel sensors are only 55 µm square in size, it is not uncommon for a particle's passage to activate more than one sensor, typically 1-4[36,263]. Therefore before pattern recognition, the reconstruction of particle tracks within the VELO begins with grouping connected activated pixel measurements into clusters[29,242]. Pixel data is read out of the VELO in 2×4 blocks, known as super pixels. Where a cluster is isolated to a single super pixel, it is processed using a lookup table of pre-calculated pixel combinations[29,36,263]. Otherwise, groupings are first built up on a series of 10×12 matrices, or 3×3 super-pixels, filled in

as neighbouring super-pixels are evaluated. Within these groupings, 3×3 cluster candidates are identified, and similarly processed using a lookup table[263].

11.4.3 Pattern Recognition

First, the angle ϕ with respect to the origin (as described in Section 11.2) is found for all hits in an event, and hits within each module ordered in increasing ϕ. This is used to define, for each hit, a window of ϕ acceptance on consecutive modules, and determine the hits which fall into them. An illustration of this is shown in Figure 11.7(a)[30].

Then, beginning farthest from the interaction point, the algorithm operates on three consecutive modules at a time. Each hit on the central module is selected, and three hit segments are formed using compatible hits from within the previously calculated ϕ windows on either side. All combinations are compared using a χ^2 least-squares fit, and if it passes a minimum threshold, the best triplet is retained as a track seed. By restricting the algorithm to look at only combinations falling within these windows, the number of calculations is greatly reduced, and by ordering the hits within a module by ϕ, those falling within a given window can be defined by the first and last which do so[30].

A forwarding stage then attempts to extend any candidates by extrapolating the segment formed by the last two hits onwards to the next module, and a ϕ window is determined. Each hit within the window is evaluated against an extrapolation function,

$$\text{extr} = \frac{dx^2 + dy^2}{dz^2} \tag{11.4.1}$$

where (dx, dy) is the displacement of the examined hit from the position predicted by the extrapolation, and dz the distance parallel to the beamline to the segment's last hit. The hit which minimises this function (again within a minimum threshold) is appended to the forming track. If a compatible hit is found, all hits composing the forming track are flagged as having been accounted for, and are then excluded from subsequent consideration. Candidates consisting only of the three initial hits are therefore left unflagged. Should a candidate go without compatible hits being found on a number of successive modules, it is stored and no longer propagated. The algorithm then progresses one module onwards and repeats the seeding-then-forwarding process. Figures 11.7(b)-(d) depict this process being carried out across five modules[30].

Finally, candidates of four or more hits are promulgated as tracks. Candidates

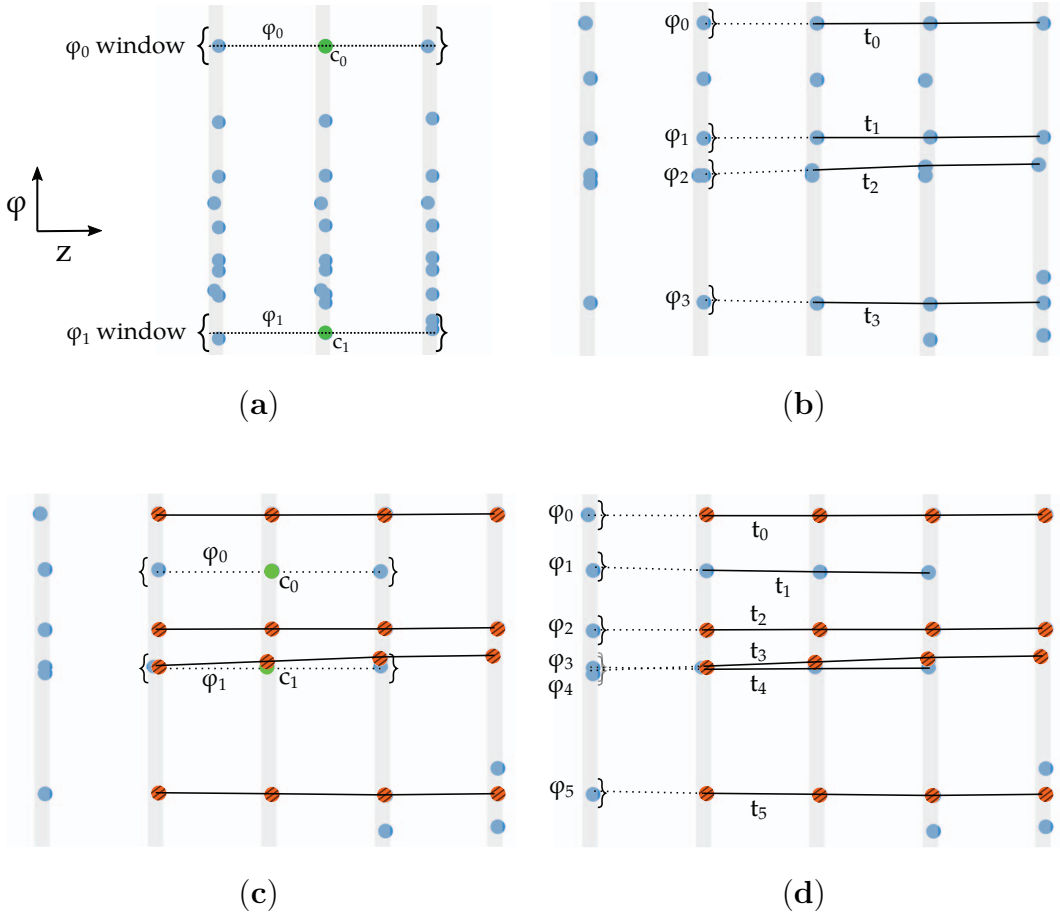


Figure 11.7: A sequence of illustrations of the stages of the pattern recognition algorithm, as described in [30]. (a) shows, for an example set of three consecutive modules, the process of determining the windows of ϕ acceptance, ϕ_i, corresponding to a middle module hits c_i, in order to generate seeds. (b)-(d) depicts the progressive building up of track candidates, beginning after an initial seeding, across five modules. (b) The set of candidate tracks, t_i, are forwarded onto the next module. Corresponding windows ϕ_i are determined and acceptable hits are subsequently added to the candidates, with all hits in those tracks flagged. (c) The algorithm progresses one module on, and attempts to seed more candidates around potential middle hits, c_i in green, using corresponding windows, ϕ_i. Flagged hits (in red) are not considered. (d) Candidates are again forwarded onwards, extrapolating the segment formed by the final two hits onto the next module, and defining a window ϕ_i for each track t_i [30]. Reproduced from [30].

of only three hits are first subject to additional scrutiny, and must both be under a χ^2 threshold, and contain no hits that were later flagged [30].

11.4.4 Track Fitting

Assuming straight line tracks, as there is negligible magnetic field within the VELO [237], candidates are fitted using a comparatively simple Kalman filter method. A transverse momentum of 400 MeV/c is used for all particles for covariance calculations, and multiple scattering is treated independently along x and y axes [242,244,262]. Particle states are estimated when closest to the beam line, for vertex reconstruction calculations, and, at the end of the VELO, for use in extrapolating tracks onwards to the other trackers [244].

In both HLT1 and HLT2, after tracks have been formed incorporating all three trackers, additional Kalman filters fittings are performed to determine the properties of particle trajectories to maximum accuracy [23,244].

11.5 Machine Learning at LHCb and the VELO

The LHCb experiment is no stranger to machine learning in its quest to understand matter; and such tools can be found in all stages of data processing [63]. Neural network classifiers are employed at various stages during and after track reconstruction to reduce fake tracks, and thus also combinatorics, particularly for long tracks [61,264]. Using data from across the detector, global particle identification is performed by neural networks, with separate algorithms estimating the probability of different identity hypotheses [265]. Boosted decision trees have served in topological selection triggers for many years [64,266], now with neural networks for the inclusive heavy flavour trigger [63,267]; and machine learning based selection triggers are responsible for the majority of data retained by the online trigger system [62]. Looking to the VELO specifically, a machine learning approach was developed for detection of anomalies in the previous VELO subdetector calibrations, and was incorporated into the detector monitoring software [268]. In addition, a neural network based pulse-shape reconstruction technique for the readout ASCIs was employed for optimisation [269].

With the upgraded Run 3 VELO on the horizon, an early investigation into integrating deep learning into pattern recognition for the new detector was published in [270], and outlined bellow in Section 11.5.1. This work was continued by the

authors, and one successor model is detailed in Section 11.5.2. Taking a different direction, a model drawing from [270] and featuring notable post processing and error reduction stages was developed, and is detailed in [36]. In addition, there has been exploration of a neural network based kalman filter algorithm for HLT1 [271].

11.5.1 An Early Pattern Recognition Model

Following previous pair seeding-based approaches, the model described in [270] employs a neural network as a classifier to estimate the likelihood a particular pair of hits forms a true track segment; acting as an alternative to both window techniques for reducing combinations and in place of other selection methods to chose the most likely pairing. Though this early model displayed significant promise, it suffered notable degrading performance approaching the interaction region.

Beginning from the station farthest from the interaction point, a seed hit (r_s, ϕ_s) on the given station is evaluated against N potential partner hits (r_i, ϕ_i) on the following station, using the neural network classifier. The classifier returns a set of N values, interpreted as the probabilities corresponding to the N examined hits on the following station. The hit with the highest predicted score and passing a threshold is taken as forming a candidate track with the seed hit. Repeating the procedure, the model works back towards the interaction region using each successful hit as the new seed for the following pair of stations, until a set of hits is encountered where no predicted score passes the threshold.

An independent instance is trained for each distinct station pairing, thus z coordinates were not included. The network itself is a fully connected feed forward neural network of four layers, with $2\,(N+1)$ inputs and layers of size $32N+1$, $32N+2$, $(32N+2)$ and N. Each layer uses a ReLU activation function, apart from the final layer which uses a sigmoid function.

11.5.2 The Hybrid Model

Building on their work, the authors of the above model developed a later algorithm, which eschewed a track forwarding approach for an additional neural network based stage. As it combines traditional pattern recognition with machine learning, for the purpose of this thesis we will refer to this model as the Hybrid Model.

After first scaling r and ϕ values by 50 mm and π respectively, as an approximate normalisation, the model commences by considering a pair of modules on successive stations at a time. For each such module pairing, every combination of a pair of

hits, one from each module, is compiled and evaluated using a pair classifier. The classifier returns a single value, interpreted as a probability score that the specified two hits form part of the same track, and is applied in batches of up to 4096 pairs at a time. A separate classifier is used for each particular combination of modules, and takes r, ϕ and the cluster size npix for both hits as its inputs. Pairs receiving a prediction of over 0.5 are retained, along with their scores, for the next stage.

Once all module pairings have been considered, the model proceeds to examine every combination of three modules that form a run across three successive stations. In each case, three hit segments, or triplets, are constructed from those retained hit pairs from the first and second pairings of modules forming the three module run, which share a common hit on the central module. These triplets are then similarly evaluated by a second form of classifier in batches of up to 4096, again returning a single value for each and using a separate classifier for each combination of three modules. The classifiers take as inputs the r, ϕ and npix for each of the three hits forming the triplet, the two probability scores of the pairs composing it received in the previous stage, and the explicit differences in r and ϕ between successive hits.

Both forms of classifier are fully connected, feed forward neural networks of three layers with ReLu activation functions, save for the final layers, which employ sigmoid functions. The pair classifiers use 6 input variables, and layers of sizes 16, 16, and 1; while the triplet classifiers use 15 input variables, and layers of sizes 64, 64, and 1. A final stage constructs tracks by connecting successful triplets where the final two hits of one triplet correspond to the initial two hits of another; while lone triplet tracks are required to pass a higher threshold. Where the last hit of a track match the first of another, it is assumed the two tracks were accidental split by a missed triplet and merged into a single track; a clone killing technique to reduce duplicate tracks.

Chapter 12

Aims and Method

12.1 Aims

Within a tracking detector, particle hits are highly relational, emerging from the complex chains of particle decays and interactions[162]. The presence or absence of other nearby hits has the potential to inform whether particular detections likely result from the same particle's flight as another. This can be seen in how many local pattern recognition methods which otherwise operate on specific hit combinations in isolation, confer to form candidates using only the combination judged most likely from among a selection, even if other hits would otherwise have been used. Not being tied to follow predetermined models, deep neural networks can infer and leverage complex patterns in data that are otherwise difficult to determine explicitly. Given the increased instantaneous luminosity of the HL-LHC and the move to full online track reconstruction, the timing requirements on LHCb VELO reconstruction present a significant challenge; and neural networks have shown great success at performing other pattern recognition tasks with fast execution times.

Therefore, our aim was to develop a proof of concept neural network model capable of performing the pattern recognition stage of VELO track reconstruction whilst examining an event holistically; and thus potentially draw inferences from the event as a whole. With graphs being well suited to describing sparse, irregular and relational data, this evolved into a graph neural network approach, and development of a network architecture agnostic framework with which to consider various GNN models. Given the typically slower execution speeds of GNN models compared to other neural networks, our focus became precision. Noting the success of architectures based on the Interaction Network[153] at tracking in the context of the ATLAS detector[180,181,182], this work culminated in a similarly inspired model.

12.1.1 The Challenge of Representation

A significant obstacle to a model which can consider an event as a whole, or at least a portion of one, lies in that a basic feed forward neural network (and therefore many other forms of neural network) takes a fixed and predetermined number of input variables; where a single event contains a variable number of hits. Therefore, the challenge is to represent the available data composing an event in a form accessible to a neural network.

Though global mapping based approaches such as the Hough Transform offer a representations in which solutions are more readily observable, they do not address this problem in themselves. A truly local approach considering subsets of a predefined size one at a time, by choosing say the nearest neighbours in real space, can circumvent this issue. However there is a potential to exclude a correct partner in high density events, or the need to pad out inputs if there are too few hits overall to fill the subset each time; and is nevertheless a rather unsatisfactory starting point considering our aim. Sequential neural network models such as LSTM are capable of operating on variable size inputs, but are liable to introducing unintended structure from the order in which information is presented.

Initially, we looked at the potential of translating the problem of track finding into a 'game' like state, and leveraging recent developments in deep reinforcement learning such as Deep Q-Learning. Reinforcement learning enables a machine learning model to be trained to act in order to maximise a final goal without an immediate payoff; in this case, the correct identification of hits that belong to the same track. By calibrating how the goal is evaluated, a model can be encouraged to place more or less importance on certain achievements over others, tuning its priorities. However, as track finding is ultimately a pattern recognition task, we were unable to find an approach that would not likely be better served as a direct classification task.

During these endeavours we investigated taking an image-like approach, in which the cloud of hit coordinates could be represented as a 3-dimensional 'image'*. As the VELO subdetector is itself fixed in size, this would represent an event of any number of hits as a fixed sized object. In this approach, the problem of tracking becomes one of instance segmentation; identifying objects within an image and which pixels are attributed to them, here identifying track instances and which hit-pixels are attributed to them. Convolutional neural network based models have proved

*Reinforcement learning approaches are generally agnostic to the specific form of neural network model used, and have been successfully used in conjunction with CNN [272], and even GNN [273], in other applications.

widely successful at instance segmentation tasks, and, depending on architecture, can operate with variable sized data, giving the opportunity to built on existing model architectures. While most instance segmentation work naturally addresses 2-dimensional images, advancements have been made into 3D image segmentation, though unstructured and haphazard 3D data remains significantly challenging [274].

However, this method presented immense issues with the size of representations. The naive approach is to represent the whole VELO using a cubic voxel grid, with a voxel size matching that of a sensor element; but the VELO's highly granular resolution, and the size of gaps between modules, leads to a representation running into Gigabyte size. Non-cubic voxels can be used, compressing the number of voxels in the z direction and using irregular lengths to minimise representation of the between module gaps. Yet even then, reaching manageable sizes necessitated down sampling, potentially merging hits, and approaches only considering only a portion of the detector at a time. Further, the sparsity of data makes representing an event this way extremely wasteful, with a significant quantity of empty voxels, and many CNN methods struggle with operating on sparse data. Ultimately, work following this approach struggled with untenable computational times for training, and as a graph based approach had began to show success, was not pursued further.

As a data structure, graphs are an efficient way to represent sparse data structures; making them a natural choice for depicting an event. By encoding each hit within the detector as a vertex, and joining all vertices which could form part of the same track by an edge, tracking becomes an edge classification problem, a task GNN have proved successful at tackling. Even though such graph representations do not have a consistent size, many recent GNN models can function on variable size graphs, and, through the message passing mechanism, remain sensitive to information encoded by a graphs structure.

12.2 Dataset Production

12.2.1 Monte Carlo Simulation

From detector design to physics analysis, Monte Carlo simulations are a widely established tool in high energy physics; and around 80% of LHCb CPU resources in the 2015 to 2018 period were employed in Monte Carlo related tasks [23]. Monte Carlo sample production for the LHCb experiment utilises a modular system, designed such that samples are processed through an identical dataflow as real data as far as

possible[23]. The Gauss package is responsible for modelling collision events, utilising dedicated physics generators such as Pythia8[275,276] or EvtGen[277], and transporting the resulting particles through the detector, typically using the Geant4 simulation toolkit[145]. Recreation of the detector and readout electronics response, known as digitisation, is performed by the Boole package, after which an emulator of the event building process is employed to produce raw data in an identical format to that output by the LHCb DAQ chain. Where desired, simulated data may subsequently proceed through the same online and offline processing, outlined in Section 10.3, as used for real data. As with other LHCb applications, both Gauss and Boole are built on the Gaudi framework[23].

12.2.2 Datasets

The sample data used in this work was produced through Monte Carlo simulation, as above, using a model of the (then future) Run 3 LHCb VELO subdetector. A detailed description of this model can be found in [257]. A large number of simulated minimum-bias events were generated, under conditions designed to reflect the events realistically recorded by the detector, and each accompanied by Monte Carlo truth information.

These events were divided into several independent batches, which we will refer to as our datasets. A first dataset was used for the example events with which neural networks were trained, a second for evaluation of a network at the end of an epoch during the training procedure, and a third for evaluating trained models. The same three datasets were used for all models, and each consisted of between 1.2 and 1.3×10^6 non-empty events.

12.2.3 Detector Scope

In order to reduce the computational burden from the size of graphs, in most instances we limited ourselves to considering only one half the full VELO detector, split lengthways along the beam-line. This is achieved by restricting our data to only even numbered modules, which with the module layout as described in Section 11.3, reduces us to a single module per station.

Approximately 10% of tracks include hits on both odd and even modules[251], and this was verified for the three datasets primarily used. At two modules per station, implementing the full detector would approximately quadruple the number of edges in a pair graph representation (see Section 12.3.1). As events are approximately

symmetric around the beam line, this still encompassed a full range and distribution of potential particle tracks, so is sufficient for a proof of concept investigation. The same datasets were employed regardless of how much of the detector was considered, with any restrictions on data applied at the time an event was loaded, reducing the number of tracks in a dataset as a whole.

A table summarising module potions is provided in Appendix B. Modules are arbitrarily labelled 0-51 from lowest to highest z potion, with module 51 closest to the remainder of the LHCb detector [250]. Unless otherwise stated, this even numbered modules scheme is used, though in some instances other combinations of modules were employed. Models are analysed using data restricted to the same combination of modules as they were trained. When only two modules are employed, modules 46 and 48 are used. For three modules, modules 44, 46 and 48 are used. For 8 modules, even numbered modules 36 to 50 inclusive are used. 26 and 52 modules refers to using all even numbered modules or all modules respectively.

12.3 Graph Neural Network Models

12.3.1 Overarching Model Framework

Building upon the approach followed by the Hybrid model (see Section 11.5.2), the various models presented here use a common overarching framework, in turn containing two independent component models. An overall model can therefore be characterised by the specific component models used, along with any other modifications to the general process.

Both component models perform edge classification, producing a single $[0, 1]$ value assigned to each edge of a given graph. This is interpreted as a probability or likelihood score, characterising the model's prediction that the hits, or combination of hits, represented by the vertices connected by the given edge constitute part of the same track. The two component models in the same overall model are always independent instances, and need not take the same form. The various component model architectures are themselves described in Section 12.3.2, and the component models used within the same overall model need not take the same form.

An overall model can be divided into three principal stages: the pair stage, the triplet stage, and the full tracking stage. An illustration outlining the overall model is provided in Figure 12.1. Models are written in Python, using a combination of the NumPy [278] (numpy.org), PyTorch [279] (pytorch.org) and DGL [280] (dgl.ai) frame-

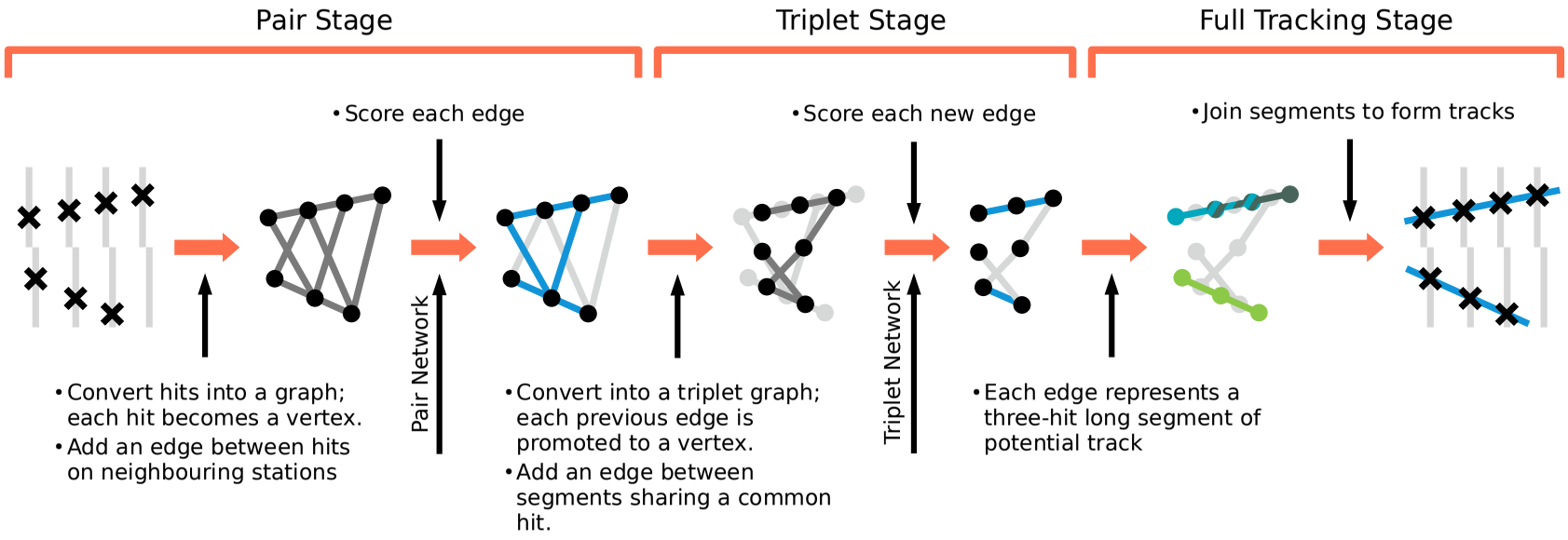


Figure 12.1: An illustration outlining the overarching model procedure. Remade from the poster "Machine Learning for LHCb VELO Track Reconstruction", Thomas Ackernley, May 2022.

works.

Pair Stage

The first stage begins with preparing a given event, with each event considered separately. Spatial coordinates are translated into a cylindrical coordinate system, in mm and degrees as appropriate, aligned along the direction of the detector, z, with the nominal interaction point as the origin. In order to approximately normalise the spacial coordinate values, unless otherwise stated, z, r and ϕ are scaled by 800 mm, 50 mm and π respectively.

From this the pair graph G^P, is constructed. Each hit within the detector is represented as a vertex (V^P), with the relevant properties encoded as the features of the vertex attributes ($\mathbf{v}_i^P$). Edges (E^P) are added between vertices representing hits on neighbouring stations. Several derived features, such as the difference between vertex variables, are encoded within the edge attributes ($\mathbf{e}_k^P$).

Due to the nature of the DGL package framework for handling graphs, all graphs are directed[281]. For neural networks, the order of input variables matters, so needs to be consistent, and in an undirected graph which vertex should be 'first' is ambiguous. We principally implement edges as directed from highest to lowest module number; for simple network models, this is sufficient as long as done consistently, and saves on computation. However, for more complex network models, this impacts message passing as the 'flow' of messages is dictated by the direction of edges. For such models, reverse direction edges are added to form a bidirectional graph, and similarly self loop edges connecting each vertex to itself. Attributes for reverse direction edges are replicated from the existing edges, but depending on what they represent, features within may then be swapped or inverted, in order to ensure a coherent description. For example, the order of variables corresponding to the initial or terminal vertex are swapped, as which vertex is initial or terminal is now the other way around; or taking the negative of the difference between vertex variables, as which value should be subtracted from the other have effectively been reversed. Self loop edge attributes are calculated as with the original edges.

The first component GNN model, denoted the pair network, is applied to this graph. Which features are exposed to the GNN model as inputs depend on the specific GNN model, but typically include z, r, ϕ, and npix, the number of activated pixels forming that hit's cluster. Original edge and vertex attributes are preserved, and any temporary 'working' vertex or edge attributes generated in their place during application of the network are discarded.

The network returns a new edge attribute of a single feature for each edge, the likelihood score, a prediction probability that the hits represented by the respective vertices connected by the edge form part of the same track. This is recorded by extending the edge attributes ($\mathbf{e}_k^P$) to include an additional feature. In order to reduce the number of combinations in the following stage, and so memory use, edges which do not meet a 0.5 threshold are discarded. For bidirectional graphs, the mean of the predicted scores for each pair of edges is used. Scores for self loop edges are ignored.

Triplet Stage

The triplet stage largely follows the same principals as those in the pair stage. Inspired by the hybrid model approach, the previous pair graph is promoted into a new graphs, the triplet graph G^T. Each pair graph edge (E^P) is now represented by a triplet graph vertex (V^T), with the attributes of the edge ($\mathbf{e}_k^P$), sender ($\mathbf{v}_{s_k}^P$) and receiver vertices ($\mathbf{v}_{r_k}^P$) concatenated to form the triplet graph vertex attributes ($\mathbf{v}_k^T = \mathbf{e}_k^P \cup \mathbf{v}_{s_k}^P \cup \mathbf{v}_{r_k}^P$)

An edge ($e_l^T \in E^T$) is added between any two triplet graph vertices (v_j^T,v_k^T) where the sender pair graph vertex ($v_{s_j}^P$) corresponding to one is the same as the receiver pair graph vertex ($v_{r_k}^P$) of the other ($v_{r_k}^P = v_{s_j}^P$). In this way, each triplet graph edge (e_l^T) corresponds to a proposed three-hit track segment. This is performed before any reverse direction edges are considered, ensuring a run of three sequential stations without doubling back. Similarly to the pair graph, various features, principally differences between vertex features, are encoded within the edge attributes ($\mathbf{e}_k^T$). So that consistent ordering is followed regarding underlying hits represented, for reverse direction edges some features are swapped around or inverted, similar to in the pair graph.

In the same manner as before, the second component GNN model, denoted as the triplet network, is applied to this graph. Again, which features are used by the GNN model as inputs depend on the specific model. They typically include the same features as for the pair model, but double the quantity due to each vertex representing two hits, and usually also including the corresponding pair likelihood score. This process returns a new score, the triplet likelihood score, for each edge; interpreted as the predicted probability that the three hits represented form a part of the same track.

Full Tracking Stage

In the final stage, discrete tracks are formed using the likelihood score predictions generated in the previous two stages. This stage was in large part taken directly from the Hybrid Model. From the triplet graph, the three hit segments each edge represents are extracted, along with the assigned likelihood scores. Where bidirectional edges are used, the mean of the two corresponding edges is again taken.

Tracks are constructed by joining these three hit segments to form chains of hits. Segments are joined where they share a common overlapping pair of hits (corresponding to a common edge in the pair graph, or a vertex in the triplet graph) and each has a triplet likelihood score in excess of 0.8. Should multiple combinations fulfil these criteria, the combination with the highest triplet likelihood score is taken. Remaining three hit segments with a triplet likelihood score of 0.95 or more are taken as complete three hit tracks. Other isolated segments are discarded. Finally, should the last hit of a track match the first of another, it is assumed the two tracks were accidental split by a missed triplet and merged into a single track. Thus, the final output of a model is a set of tracks, where each track is a variable length sequence of hits given in the original event.

To boost this process, we also implemented an additional filter, designed to identify and correct where tracks had become split by a missed segment. Each track was characterised as a straight line using its end points, and one by one extrapolated back across 3 stations. If another track was found following the same trajectory, with a tolerance of 1mm in the x or y directions and a pseudorapidity of 1, the two tracks were merged. This filter was applied only where explicitly stated.

12.3.2 Component Network Architectures

Basic Network

As the initial model implemented, the Basic Network is a straightforward fully connected feed forward neural network, consisting of a series of fully connected layers of neurons, followed by ReLU activation functions. The final layer, instead of a ReLU function, employs a sigmoid activation function.

Applied to each edge, various features are taken from the edge and connecting vertices as the inputs to the first layer, with the same features taken from each. Referring to the GN Block framework in Section 8.2.3, this GNN model in its entirety is a single performance of the first step, $\mathbf{e}'_k = \phi^e\left(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}\right)$ with the aforementioned neural network as ϕ^e.

When used as the Pair Network, a default of three layers are used, consisting of 16, 16, and 1 neuron respectively. The variables z, r, ϕ, and npix are taken as inputs, for a total of 8 inputs to the first layer.

As the Triplet Network, a default of three larger layers are used, consisting of 64, 64, and 1 neuron respectively. The variables z, r, ϕ, and npix for each of the hits encoded within a vertex are again used as inputs, along with the differences in z, r and ϕ, and the pair score assigned in the previous stage, taken from the edge attribute. As edges connect vertices representing a common hit between them, the second set of features representing the shared duplicate hit are discarded, for a total of 18 input variables to the first layer.

Several model variations are also examined, such as different numbers of layers with different numbers of neurons within them, or different combinations of inputs. In addition, variant was created employing separate network instances for each combination of modules.

Interaction Network

Inspired by the use of Interaction Networks for particle tracking in [180] and [181] (see Section 8.3.3), the Interaction Network model detailed here is based on the architecture introduced in [153]. From here on, any reference to the Interaction Network refers to the model described here, as opposed to the general model. A diagram outlining the overall algorithm is given in Figure 12.2.

Throughout this model, five fully connected feed forward neural networks of a similar design are used. These consist of three layers of neurons, each of 24, 24, and y neurons respectively, where y is the desired number of outputs in each specific case. Each layer is followed by a ReLU activation function, with one exception noted in what follows, and all such networks are independent instances.

Initially, a pair of encoder networks, ψ^e_{enc} and ψ^v_{enc}, are applied separately to the edge and vertex attributes, taking the designated input features and producing a new attribute of a corresponding number of latent features in place of the original ($\mathbf{e}'_k = \psi^e_{\text{enc}}(\mathbf{e}_k)$ and $\mathbf{v}'_i = \psi^e_{\text{enc}}(\mathbf{v}_i)$).

The model's core consists of the iterative section designed to exploit the message passing principal (see Section 8.2.4). Referring to the GN Block framework, this section corresponds to steps 1 though 3, and may be performed multiple times. Using a form of skip connection, the inputs taken for each edge or vertex use two sets of values, the current features ($\mathbf{e}_{k,t}$, $\mathbf{v}_{i,t}$), as produced by the previous iteration, along side the preserved initial latent features ($\mathbf{e}'_k$, $\mathbf{v}'_i$). For the first iteration, the current

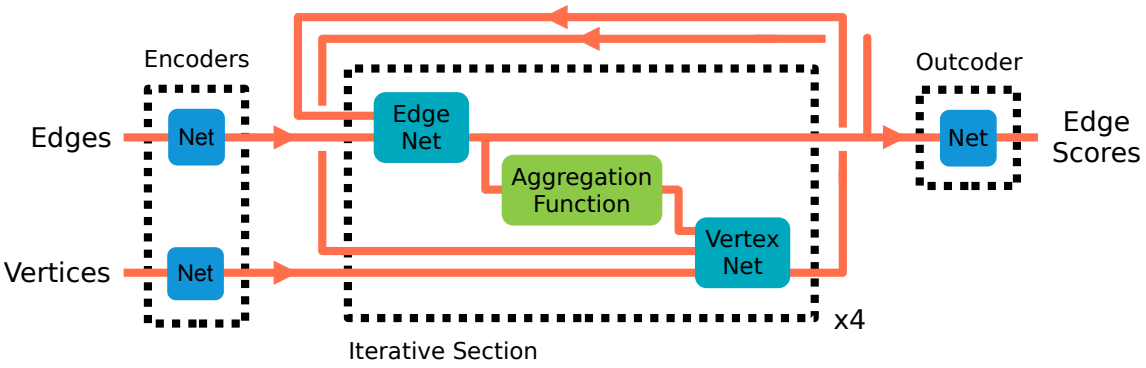


Figure 12.2: An outline of how the Interaction Network model operates. Blue boxes indicate the neural networks within the model, and solid, arrowed lines indicate how edge or vertex attributes are passed between them. The aggregation function operating on the relevant edge attributes for each vertex is in green. Dotted lines indicate descriptive groupings of components. The central section, denoted the iterative section, is repeated typically four times, with the previous output edge and vertex attributes fed back in. Note the diagram is representative of the baseline configuration of the model, and does not hold for some variants, such as those using alternative skip connections.

features are also the initial latent features, doubling up. First, the edge attributes are updated with the edge update network, ϕ^e, producing a new set of edge attributes, with an equal number of features, which act as the message to be passed onwards ($\mathbf{e}_{k,t+1} = \phi^e\left(\mathbf{e}_{k,t} \cup \mathbf{e}'_k, \mathbf{v}_{r_k,t} \cup \mathbf{v}'_{r_k}, \mathbf{v}_{s_k,t} \cup \mathbf{v}'_{s_k}\right)$). Using the aggregation function, $\rho^{e\to v}$, these new attributes are aggregated for each vertex ($\overline{\mathbf{e}}_{i,t+1} = \rho^{e\to v}(E_{i,t+1})$). Unless otherwise given, this took the form of summing the corresponding features within the incoming edge attributes. Subsequently, the vertex update network, ϕ^v, is then applied to each vertex, similarly producing a new set of vertex attributes.

Finally, the outcoder network ψ^e_{out} is applied to the edges, taking the last edge attributes and generating the final likelihood score. Contrary to the previous networks, this last network utilises a sigmoid activation function on the final layer.

The Interaction Network can be summarised, with input edge features $\mathbf{e}_k$ and vertex features $\mathbf{v}_i$, as;

1. $\mathbf{e}'_k = \psi^e_{\text{enc}}(\mathbf{e}_k)$ and $\mathbf{v}'_i = \psi^e_{\text{enc}}(\mathbf{v}_i)$ are performed per edge or per vertex respectively to generate latent attributes, of equal size to the original.

2. Iterating $t = 0, \ldots, t_{\text{final}}$, where initially $\mathbf{e}_{k,t=0} = \mathbf{e}'_k$ and $\mathbf{v}_{i,t=0} = \mathbf{v}'_i$

   (a) $\mathbf{e}_{k,t+1} = \phi^e\left(\mathbf{e}_{k,t} \cup \mathbf{e}'_k, \mathbf{v}_{r_k,t} \cup \mathbf{v}'_{r_k}, \mathbf{v}_{s_k,t} \cup \mathbf{v}'_{s_k}\right)$ is performed per edge

(b) $\overline{\mathbf{e}}_{i,t+1} = \rho^{e\to v}(E_{i,t+1})$ is performed per vertex.

(c) $\mathbf{v}_{i,t+1} = \phi^{v}\left(\overline{\mathbf{e}}_{i,t+1}, \mathbf{v}_{i,t} \cup \mathbf{v}'_{i}\right)$ is performed per vertex.

3. $\mathbf{e}_{k}^{\text{score}} = \psi_{\text{out}}^{e}\left(\mathbf{e}_{k,t_{\text{final}}}\right)$ is performed per edge, generating the final likelihood scores.

As a baseline, the iterative section was performed 4 times. When used as the Pair Network, the variables z, r, ϕ, and npix are taken as the input features for vertices, and the difference in z, r and ϕ for edges. For the edge and vertex update networks, this gives a total of 22 and 11 input variables respectively. When serving as the Triplet Network, more variables are used as inputs. For vertices, z, r, ϕ, and npix for both encompassed hits are used, along with the explicitly calculated differences in z, r and ϕ, and the corresponding pair likelihood score. For edges, the difference in the difference in z, r and ϕ is along side the difference in pair likelihood score. In the case of the edge and vertex update networks, this results in a total of 56 and 28 input variables respectively. Unlike in the Basic Network, features corresponding to the duplicated 'middle' hit are not discarded. The positions of which features within the triplet vertex attributes correspond to which of the two underlying hits, represented by each triplet graph vertex, are fixed. But with the presence of reverse direction edges, which underlying hit, and so features, should be interpreted as the shared hit depends on the direction of the given edge under consideration. Regardless, after the encoder network is applied we work with latent features, which no longer directly correspond to an original input, and so a specific hit.

As before several variations are also considered, though as a more complex model, more variations are possible. Different inputs, numbers of layers and neurons per layer can be used, and can be different for each of the five instances. Other parameters such as numbers of iterations can be changed. The encoder networks can be disabled, allowing the original input features to reach the first iteration stage. The skip connection can be removed, or alternatively, rather than the latent features, the the original pre-encoder inputs may be passed through to the iteration stage each time.

12.3.3 Training Procedure

The training procedure for network weights took place in two parts, with the pair and triplet networks trained separately. For a pair network, only the pair stage of the overarching model was performed each time. For a triplet network, both

the pair and triplet stages were performed, using a pre-trained pair network, as the pair stage is required to produce the triplet graph. Thus a specific triplet network is also dependent on what pair network it was used when trained. In either case, networks were trained based on their predicted likelihood scores. Edges where all hits corresponding to the connected vertices are from the same track were given a target likelihood score of 1, otherwise 0, resulting in a binary classification task. When included, self loop edges were also given a target of 1. For the variant of the Basic Pair Network which employed separate network instances for each module combination, each instance was trained separately, with the dataset restricted to hits on the specific pair of modules chosen each time.

Weight adjustment was performed using the Adam algorithm[56], with a learning rate of 0.002 and a weight decay of 10^{-5}. For the loss, the Binary Cross Entropy loss was effectively used. This was achieved using the Pytorch *BCEWithLogitsLoss* function[282] was used, a combination of a Sigmoid layer and Binary Cross Entropy loss function equivalent to

$$\text{BCEWithLogitsLoss} = -\frac{1}{M}\sum_{m}^{M}\left(p\ u_m \log\left(\sigma\left(\hat{u}_m\right)\right) + \left(1-u_m\right)\log\left(1-\sigma\left(\hat{u}_m\right)\right)\right) \tag{12.3.1}$$

where σ denotes a sigmoid function, M the number of samples, and u and $\hat{u}$ the predicted and truth values respectively. During training, the explicit final layer sigmoid activation functions were disabled, effectively moving them into the loss function. This combination function approach is more numerically stable than using a Binary Cross Entropy loss function directly with a final sigmoid layer[282] Events discarded during graph construction, for reasons such as for having no viable edges, were not included in the loss or other metrics. Back propagation and optimisation was performed using a graph, and so an event, as a batch.

Due to the often significant prevalence of false edges, and so negative training examples, particularly in a pair graph, models are likely to converge to predicting 0 for all edges, To counteract this, rather than using an unbalanced sample, *BCEWithLogitsLoss* allows for a positive weighting, p, to be used to bias the training effect of positive examples. For a binary classification problem with a dataset containing n^p positive and n^n negative examples, an equal effective proportion is achieved by

$$p = \frac{n^n}{n^p}\ . \tag{12.3.2}$$

The value of p was set as above using the training dataset. As pair graphs

are determined entirely by the dataset, this is therefore fixed regardless of the pair network trained. When limiting to the standard even module half, this weighting is approximately $p = 61.7$, or $p = 37.2$ if bidirectional and self loop edges are included. However for triplet network training, due to the 0.5 threshold on pair score used, exactly which vertices, and so edges, are present in the triplet graph depends on the performance of the respective pair network; requiring recalculation each time. For some models, the positive weighting was additionally modified by introducing a weighting multiplier, in order to encourage better performance at classifying true positives or negatives, likely at the expense of the other. Where this has been done will be indicated.

Each training was performed over a training dataset multiple times, until further passes no longer showed sufficient performance improvement. Due to practical constraints, rather than full randomisation, the order of events was pseudo-randomised for each epoch. Initially, the order of the 13 files that constituted the principal training dataset was randomised. Then within each file, events were loaded in batches of 1000 at a time and randomised within each batch.

After each epoch, the current instance was evaluated on a fixed 10000 event sample, drawn at the beginning of the process from a separate dataset to avoid overfitting, and the mean loss evaluated against a benchmark according to

$$\text{gain} = \frac{\text{benchmark loss} - \text{current loss}}{\text{benchmark loss}} \,. \tag{12.3.3}$$

The same loss function, including positive weighting bias, was used in optimisation. For the validation dataset as a whole, the optimum weighting for balance would be approximately $p = 62.1$, a negligible difference. However, one limitation is that as the validation sample is drawn at the beginning of training and retained throughout, there is the potential for said sample to have a notably different imbalance.

Should the current instance show a gain greater than a minimum threshold of 5×10^{-4}, training continues and the benchmark loss is updated to the new, most recent loss. Regardless of whether the threshold is met or benchmark updated, should the current model show an improved loss over previous tests, the model instance is recorded as a checkpoint. If, after four consecutive epochs, the model failed to reach the gain threshold, the training procedure was terminated and the model restored to the last recorded checkpoint. Otherwise, the network instance was preserved from one epoch to the next.

Training Performance

It was observed that, while a higher validation than training loss can be expected, the post-epoch validation loss for our models was always significantly higher, particularly in comparison to the scale of fluctuations. It was later discovered that this difference in scale was likely attributed to an issue in calculating the loss values when applied to the validation dataset. Note that is the post-epoch loss used to gauge a models training progress, not the loss used to optimise the model. As described above, the *BCEWithLogitsLoss* loss incorporates the effect of a Sigmoid layer with a Binary Cross Entropy loss function, which is why during training the Network's final Sigmoid layer was disabled. However during inference on the validation the model was being set with aforementioned Sigmoid layer enabled. By subsequently using the *BCEWithLogitsLoss* loss function, the predicted likelihood scores were in effect passed through a sigmoid function a second time. Given that this would shift prediction values into the range 0.5 to 0.73, against targets of 0 or 1, this inflated the validation loss. Due to the manner in which they were recorded during training, we were unable to recover the actual loss values for our models retroactively.

The training code was modified to correctly disable the sigmoid function. Initial test trainings of the Basic Pair Network yielded training and post-epoch losses of comparable size and shape; indicating that this was the primary contributor at the very least, and that there was not a fundamental issue with our approach. Given time constrains and technical issues, we were unable to sufficiently explore further, nor did we have the significant computational time required to retrain and evaluate all our models in response.

It is worth noting that the non-symmetric-about-0.5 and non-linear scaling this introduced will have altered the relative contributions of specific predictions to the loss, and the effective impact of positive and negative edges. As the post-epoch validation loss was used to gauge training progress, this may therefore have altered the number of epochs ultimately trained for. However the training loss, which was not affected, behaved as expected, usually plateauing after few epochs. Equally, the initial test trainings under the revised implementation trained for a comparable number of epochs. Therefore the impact of this issue appears to have been minor.

In addition, though our training procedure was predominantly successful, several variations of the Interaction Triplet Network suffered from issues in training. Despite the use of a positive weighting in the loss, calibrated for the specific Pair Network used, some Interaction Triplet Networks would converge to predicting that all segments presented were either all true or all false; appearing to have become stuck in

a local minima. In some cases, upon repeating the process a model would train as intended, while in others the issue would continually repeat. In our time frame, we were unable to ascertain exactly when or why this instability would occur.

12.4 Measurements and Analysis

12.4.1 Measurement Errors

All analysis was carried out using a separate dataset to both training and post-epoch assessment, and of roughly the same number of events. Performance is measured per event and the mean taken over all events in the dataset. The standard deviation given, usually denoted with σ or $\pm$, indicates the statistical error on this mean.

12.4.2 Component Network Performance

Component networks are themselves a form of binary classifier, and can be evaluated using standard methods. The network is regarded as having made a positive or negative prediction using a score threshold of 0.5. Edges themselves are categorised as positive if they correspond to a sequence of hits from the same track, else negative. For triplet networks, the same pair network as in training was used.

The accuracy, or error rate, is then the fraction of edges in an event correctly classified by the network,

$$\text{Accuracy} = \frac{\text{No. Correct Predictions}}{\text{No. Total Predictions}}\,. \tag{12.4.1}$$

The performance at classifying positive segments can be expressed as proportions,

$$\text{True Positive Rate} = \frac{\text{No. True Positives}}{\text{No. Positives}}\,, \tag{12.4.2}$$

$$\text{False Negative Rate} = \frac{\text{No. False Negatives}}{\text{No. Positives}}\,, \tag{12.4.3}$$

and similarly for negative segments,

$$\text{True Negative Rate} = \frac{\text{No. True Negatives}}{\text{No. Negatives}}\,, \tag{12.4.4}$$

$$\text{False Positive Rate} = \frac{\text{No. False Positives}}{\text{No. Negatives}}\,, \tag{12.4.5}$$

As the true positive and false negative rate add up to one, and similarly for the true negative and false positive rate, the false positive and false negative rates will not be explicitly given.

Due to the typical prevalence of false edges, and so negatives, it is useful to consider the balanced accuracy, given by,

$$\text{Balanced Accuracy} = \frac{\text{True Positive Rate} + \text{True Negative Rate}}{2} \tag{12.4.6}$$

which assigns equal weight to performance at classifying true and false examples.

Another common way to assess binary classifier performance is to consider the receiver operating characteristic, or ROC, curve; the true positive rate plotted against the false positive rate across a range of score threshold settings. To characterise this distribution as a single value, the area under the curve, which we will refer to as the ROC AUC, can be considered as the probability that, given a positive and negative example, a model will assign a higher score to the positive example[283].

12.4.3 Tracking Performance

In order to measure a full model's performance at tracking, we are going to borrow the terminology used in [251], which are commonly used throughout the experiment;

- The track corresponding to a particle is considered reconstructable if there are hits made by that particle's passage on at least three separate modules.
- A particle's track is considered successfully reconstructed if there is a predicted track using at least 70% of the hits made by said particle.
- A ghost, or fake, track is a predicted track where less than 50% of hits forming the track correspond to the same particle, and so is not associated to the passage of any particle.
- Alternatively, if any single particle has more than one reconstructed track associated to it, then any such predicted tracks beyond the first are counted as clone tracks.

For the purposes of the number of hits made by a track, for any one model we will only ever consider those on modules used for that specific model. Using these terms, we can therefore further introduce several corresponding means to measure tracking performance;

- The Efficiency is the proportion of reconstructable tracks which have successfully been reconstructed,

$$\text{Efficiency} = \frac{\text{No. Reconstructable}}{\text{No. Reconstructed}} \tag{12.4.7}$$

  providing a measure of how well a model performs at assigning groups of hits made by the same particles path into the same track.

- The Ghost, or Fake, rate is the proportion of predicted tracks that are ghost tracks,

$$\text{Ghost Rate} = \frac{\text{No. Ghost}}{\text{No. Predicted}} \tag{12.4.8}$$

  and measures how often a model predicts tracks, and so particles, within the data that are not there.

- The clone rate is similarly the proportion of predicted tracks that are clone tracks,

$$\text{Clone Rate} = \frac{\text{No. Clone}}{\text{No. Predicted}} \tag{12.4.9}$$

  and measures how often a model splits or duplicates tracks made by the same particle, essentially seeing duplicate particles.

An ideal model would therefore have an efficiency of 1, alongside a ghost and clone rate of 0. Code for the calculation of these metric was in large part taken from that developed for analysing the Hybrid Model.

12.5 Comparison Model

12.5.1 Hybrid Model Implementation

To provide a baseline against which to assess performance, we utilised an existing and pre-trained implementation of the Hybrid Model. Given this model had served as a starting point upon which the models described in the remainder of this chapter were developed, including having being trained using the same form of dataset, it could provide a straightforward comparison for both overall performance and individual stages. The model used in the following chapter was implemented predominantly using existing code, including trained weights, with modifications to work with our overall execution, data and statistics handling. The Hybrid Model is described in Section 11.5.2, and having served as the basis of the third stage of the

overarching model framework, further details of the model's final stage can be found in Section 12.3.1.

12.5.2 Training Procedure

To accommodate the overabundance of false hit combinations compared to true, a biased sampling method was employed during optimisation of the Hybrid model we utilised. The training sample for a given pair classifier was built up using example pairs over all events before training commences. For each event, a subsample was constructed using each hit on the module with the highest module number, of the two corresponding to the given classifier, once. Should a hit have a partner on the other module with which it forms a true track, that pair was always used, otherwise a partner hit was selected at random. Should more than half the pairs in a sample form true track segments, and there is at least one such pair, the sample was extended with a number of additional entries, equal in length to the number of true pairs. These additions were constructed by duplicating the first entries of the subsample and cycling their partner hits. Said subsamples were subsequently combined, and the resulting sample shuffled. This method resulted in samples with a varying ratio of fake to true pairs, always with more fake pairs than true, and sees further discussion in Section 13.1.4. Training of triplet classifiers used every valid triplet formed after a set of relevant prepared pair classifiers are applied to each event, similarly compiled and shuffled beforehand. In both pair and triplet training, every other entry in a sample was removed and set aside to form a separate sample for testing a given networks progress during its training.

As the Hybrid Model's training procedure was used as inspiration for those of the models described in the remainder of this chapter, it otherwise bears significant similarities to that described in Section 12.3.3. Weight adjustment was performed using the Adam algorithm[56], with a learning rate of 0.001 and a weight decay of 10^{-5}. The Pytorch *BCEWithLogitsLoss* function[282] was used for the loss.

Classifiers were trained over the training sample multiple times, and after each epoch was applied to the test sample and the mean loss recorded. Beginning with an initial benchmark loss of 1.0, the classifier was evaluated using Equation 12.3.1. If it failed to achieve a minimum gain threshold of 0.005 after four successive epochs, training was terminated. Otherwise, training continued for another epoch, and if the gain exceeded the threshold, the benchmark loss was updated to the previous mean test loss.

The version of the Hybrid Model utilised here was trained using only even num-

bered modules, as described in Section 12.2.3. However, as each module pair or triplet used a independent network instance the choice of modules considered has no effect save which combinations have corresponding networks trained.

Chapter 13

Results

Tables of results can be found in Appendix C.

13.1 Basic Network Models

13.1.1 Consistency with the Hybrid Model

The Basic Networks and overall Model use the same neural network architecture and roughly the same procedure as the Hybrid Model, as in effect they perform the same operations on the same inputs, just with one arranged in the context of a graph. Therefore their performance should be comparable, providing a consistency check to evaluate if the Basic Model is behaving as expected. However, one notable difference is that the Hybrid Model uses a separate network instance for each module pair combination.

Pair Network

As shown in in Figure 13.1(a), the Basic Pair Network accuracy is around 1σ lower than the Hybrid Pair Network, but exhibit a very similar balanced accuracy. Breaking down to the true positive and true negative rates, we can see explicitly that the Basic Network is comparably stronger at identifying true edges, at the expense of worse performance at discounting false edges.

Due to the calculation of the bias weighting being tuned to effectively equal quantities of positive and negative examples, the Basic model was in effect trained to maximise the balanced accuracy, not the accuracy. The Hybrid model instead used a sampling method, with batches with slightly higher quantities of negative examples, in effect marginally prioritising the true negative rate over the true positive rate. As

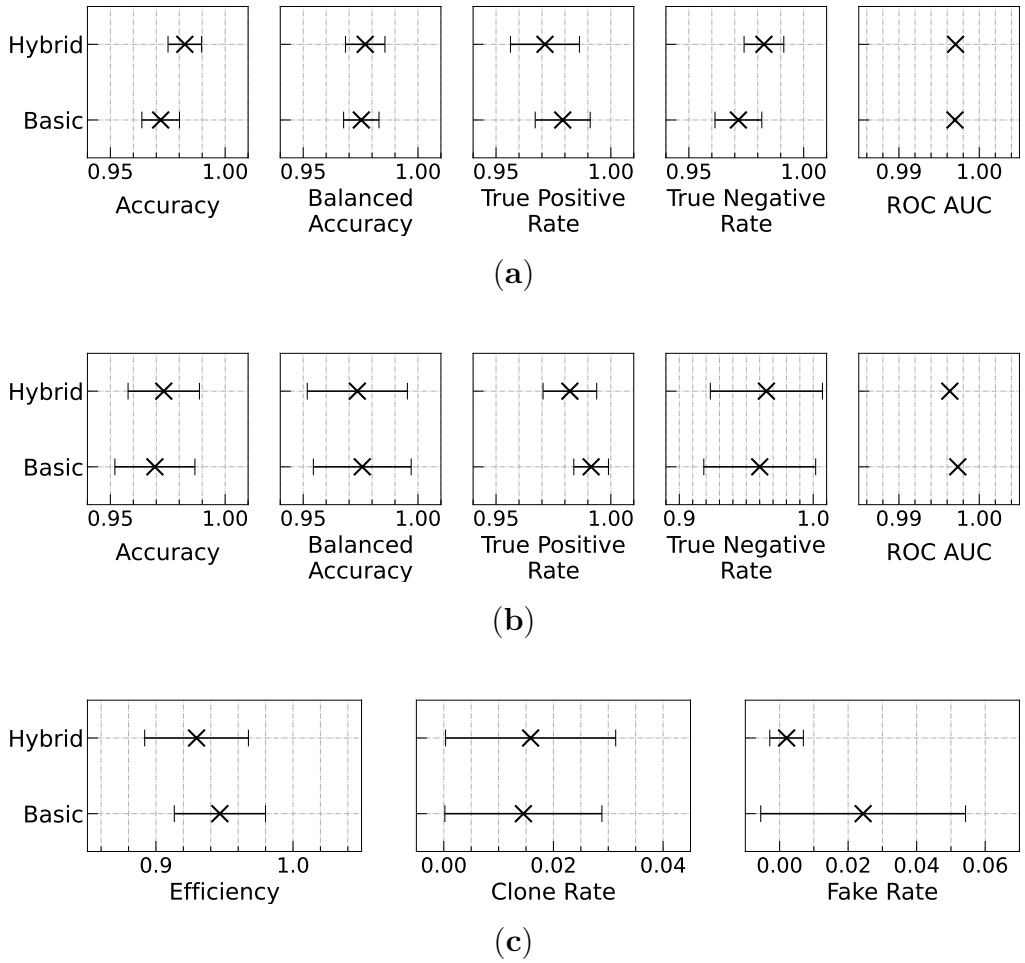


Figure 13.1: Performance metrics of the Hybrid and Basic (a) Pair and (b) Triplet Networks, and (c) the full models at the complete tracking problem.

events contained a preponderance of negative examples, this goes some way to explain the difference in behaviour. Ultimately though, the differences in performance are within 1.5σ.

Triplet Network

The Triplet Networks, as shown in Figure 13.1(b), performed similarly to the two one another, with very similar balanced accuracy and the Basic network exhibiting slightly lower accuracy. Apart from the true positive rate and ROC AUC, performance was generally closer between the two models compared to the Pair Networks, and errors larger. In particular, the true negative rates exhibited notably larger errors than for the Pair Network, suggesting both models were more erratic

at discounting false segments during this stage.

Though not likely relevant here, its worth noting that the strong performance of a Pair Network can make any subsequent Triplet Network's task comparatively more difficult, by potentially removing easier to classify segments from an event. Additionally, Triplet Networks generally include the relevant Pair Network scores as inputs, with more accurate scores likely aiding the Triplet Network in its own classification. Thus comparison between Triplet Network stage performances can be somewhat misleading, owing to having different Pair Networks effectively being as if comparing models trained and evaluated on slightly different datasets.

Full Model

Looking at performance of the complete models at the full tracking problem, shown in Figure 13.1(c), we can see the Basic Model efficiency is slightly higher, and the clone rate slightly lower, than the Hybrid Model. This lines up with what we would expect from the Pair and Triplet Networks individual performances; with the Basic Network's stronger performance at identifying true track segments leading to an improved efficiency, while the worse performance at mistakenly identifying incorrect segments as part of a track leads to a higher fake rate.

While the efficiency and clone rate are relatively close, the Basic Model's fake rate is approximately an order of magnitude larger. However, the error on the Basic Model's fake rate is also significantly larger, enough to encompass the Hybrid Model's fake rate and error.

13.1.2 Separate Network Instances

To investigate the impact of using separate network instances for edges between different combinations of modules, as the Hybrid Network does, a variant of the Basic Pair Network was created that behaved this way. This Pair Network, alongside the Hybrid and regular Basic Pair Networks, was evaluated for a range of quantities of modules, as described in Section 12.2.3, with the results shown in Figure 13.2. For the regular single instance Basic Pair Network, additional instances of the model were trained and evaluated on restricted numbers of modules to produce the additional data points.

Both Basic Networks performed very similarly. As the number of modules considered is increased, the single instance version does not appear to show significant deterioration in performance, suggesting that using a single network is sufficient.

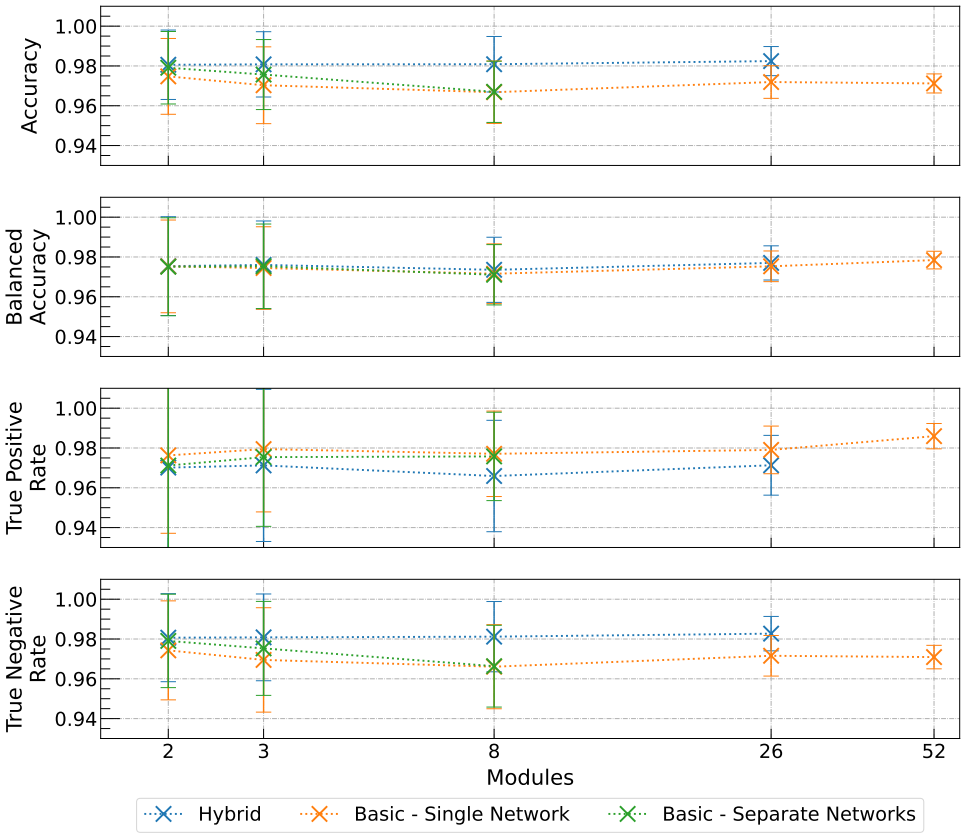


Figure 13.2: Performance of the Hybrid, Basic, and separate network Basic variant pair stage networks, trained and evaluated over different sized portions of the detector, as described in Section 12.2.3. Note the horizontal axis uses a logarithmic scale.

As before, both Basic Pair Networks shows the same performance relative to the Hybrid Pair Network, indicating that using separate instances is unlikely a factor in the performance differences between the Basic and Hybrid Models. In general, errors decrease as more modules are included, possibly a consequence of the increasing number of segments for statistics.

Also included are results of the Basic Pair Network trained over the full 52 modules. Performance is similar to that for only even numbers modules, confirming that working with only even numbered modules is largely sufficient to get an idea of how models generally perform.

13.1.3 Input Variables

In order to assess the effect of including the z coordinate, and using an explicit approximate normalisation scaling of the input variables (as described in Sec-

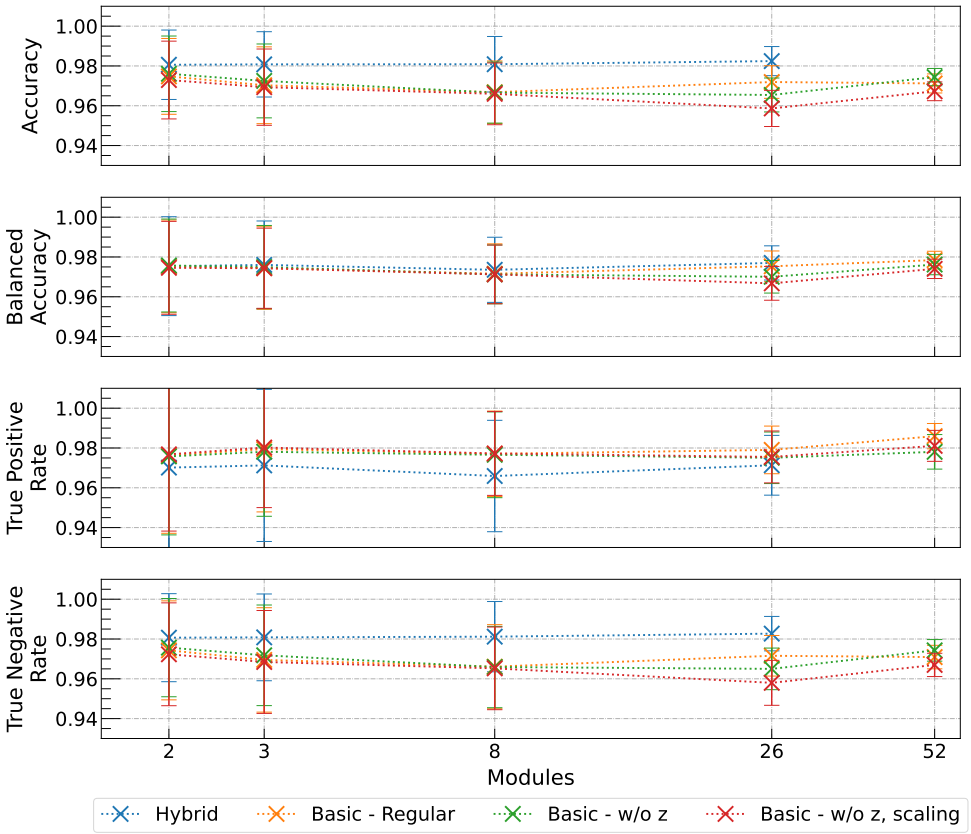


Figure 13.3: Performance of the Basic Pair Network using different combinations of including or excluding the z coordinate or the use of an approximate normalisation scaling, trained and evaluated over different sized portions of the detector, as described in Section 12.2.3. The regular Basic and Hybrid Pair Networks are included for comparison. Note the horizontal axis uses a logarithmic scale.

tion 12.3.1), additional Basic Pair Networks were trained without these features. This was performed, as above, for a range of different quantities of modules, with the results given in Figure 13.3.

As before, errors generally decreased across the board as more modules are included. While excluding the z coordinate alone also showed a reduction in performance as the quantity of modules was increased, the effect was not as significant as expected; and for the full 52 modules, actually performed slightly better at accuracy, though still within 1σ. Similarly excluding both the z coordinate and explicit normalisation showed a further drop in accuracy and balanced accuracy as the number of modules was increased, but with a slight rise for 52 modules. This, along with the separate networks results, suggests that, when considering only a single pair of hits at a time, the location of the two hits along the length of the detector appears to

be of surprisingly little value when assessing if the hits form a track segment. One possible influencing factor for this may be the layout of the VELO. The detector consists of a series of sensitive planes arranged in a row, as opposed to a all encompassing barrel design. Valid tracks, and so their segments, are likely to be at small angles to the beamline regardless of position along its length.

13.1.4 Training Bias

Motivated in large part by the differences in the balance of true positive rate and true negative rate performance between the Basic and Hybrid models, we introduced a multiplier to the positive weighting bias in the loss function. Values greater than 1 increase the effect of positive examples on the loss, effectively equivalent to increasing the proportion of positive examples encountered, thus encouraging more importance to be placed on classifying them correctly over negative ones, and the converse for lower values.

There is also a potential benefit for using a multiplier above 1 in training the pair stage, in order to ensure all true segments proceed to the triplet stage; as the triplet stage cannot correct for those mistakenly discarded by the pair stage, but can reject additional edges misclassified as being part of a track.

Pair Network

From Figure 13.4 we can see that, as expected, the true positive rate improves at the expense of the true negative rate as the multiplier is increased, and visa versa. Balanced accuracy remains consistent for multipliers bellow 1, at least well within 1σ, but begins to degrade above 1. For multipliers larger than 1, the true negative rate seems to be more sensitive, its performance decreasing faster than the improvement in the true positive rate, resulting in a reduction in balanced accuracy. Accuracy on the other hand generally improves as the multiplier is lowered. As an event contains significantly more false edges than true, the improvement in true negative rate performance appears to be sufficient to surpass the loss in the true negative rate.

The Basic Pair Network recovers the behaviour of the Hybrid Pair Network at low multiplier values. However, the combined Hybrid sampling scheme is statistically equivalent to using an effective multiplier of approximately 0.8, while the Basic Pair Network does not exhibit similar behaviour until around 0.5; though is within 1σ by 0.8. The Hybrid Network utilises different instances for different module combina-

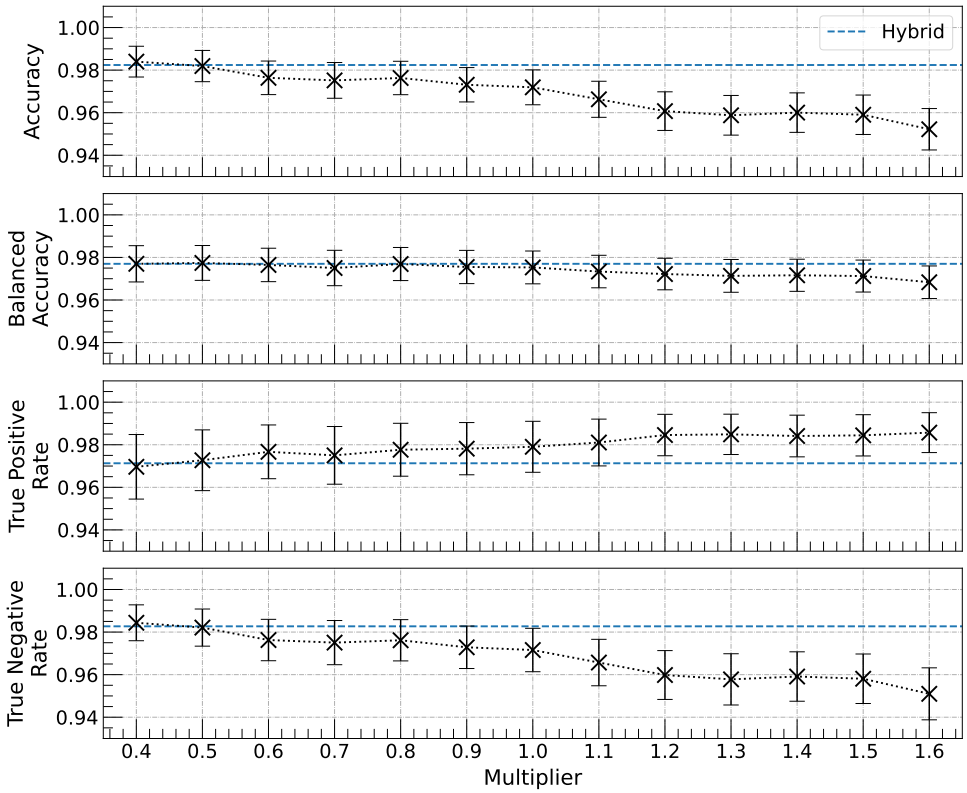
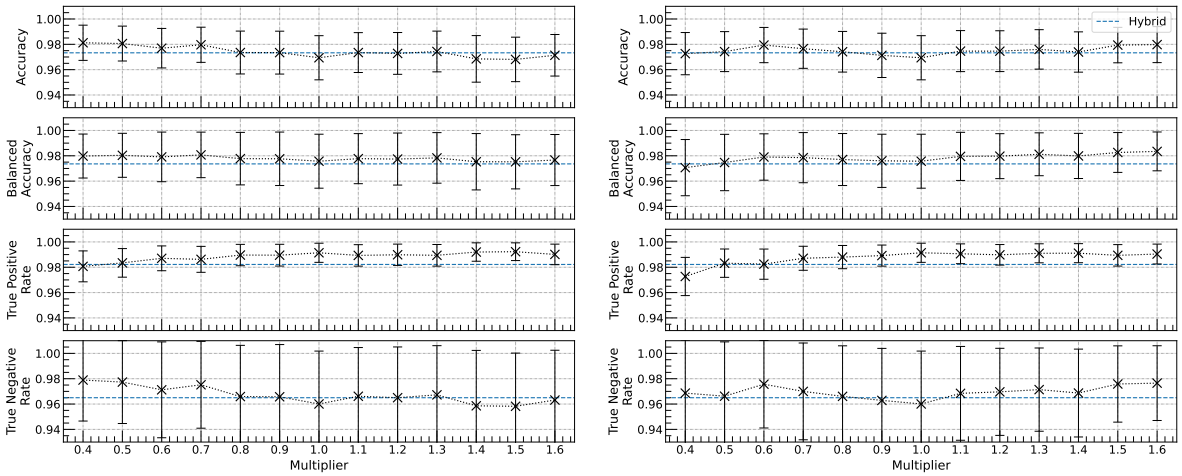


Figure 13.4: Performance of the Basic Pair Network with a multiplicative value applied to the positive weighting bias used in the loss during training. The Hybrid Pair Network performance is indicated by a dotted line.

tions, each trained with separate samples. These samples are individually equivalent to effective multipliers between 0.67 and 1.51 on the overall positive weighting bias equivalent to a balance. But, if we take the ratios of positive and negative examples for each module combination separately, and the positive weighting required for balance for each, the Hybrid sampling scheme has an effective multiplier of approximately 0.8 for all module combinations separately.

Triplet Network

Several sets of Basic Triplet Networks were trained using a multiplier applied to the loss positive weighting bias in the same manner. One set of networks, shown in Figure 13.5(a), used the regular Basic Pair Network, with no multiplier, for the pair stage. Other networks, shown in Figure 13.5(b), instead used the Basic Pair Network, as shown above, with the same corresponding multiplier. Additional Triplet Networks were trained with a multiplier applied during the Pair Network training, but with no multiplier during Triplet Network training. Due to the difference in



(**a**) (**b**)

Figure 13.5: Performance of the Basic Triplet Network with a multiplicative value applied to the positive weighting bias used in the loss during training, using (a) the regular Pair Network for the pair stage with no multiplier applied, and (b) the Pair Network with the same corresponding multiplier for each pair stage. The Hybrid Pair Network performance is indicated by a dotted line.

edges that are retained onwards to the triplet graph, any Triplet Network using a Pair Network with a multiplier applied has a different positive weighting bias, calculated for balance as usual, with any multiplier applied subsequently.

Where a multiplier is only applied to the Triplet Network, we see similar behaviour to that in Figure 13.4, with the true positive rate showing improvement at the expense of the true negative rate as the multiplier is increased; though the effect is in general less pronounced. Where a multiplier is applied during the training of both stages, accuracy and balanced accuracy are fairly flat, with a slight rise at high values, and a dip at low values due to a drop in the true positive rate. The true negative rate is erratic, but with large error values is relatively insignificant.

Full Model

Performance of the Basic Model, with a multiplier applied to the loss positive weighting during training of the Pair Network, the Triplet Network, or both, at tackling the full tracking problem is illustrated in Figure 13.6. Efficiency is fairly consistent across the board, except at low values where we see a drop, particularly when the multiplier was applied in both stages. The clone rate is particularly similar for all models, with a slight reduction as the multiplier increases, and a rise at low

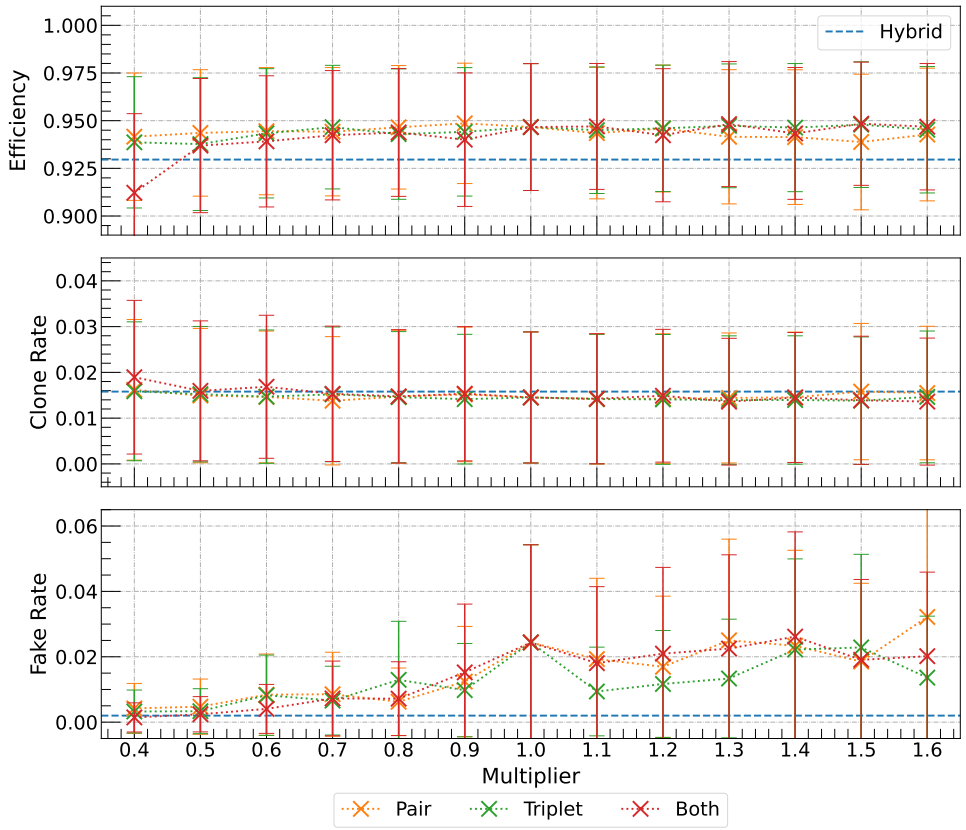


Figure 13.6: Performance of the Basic Pair Network with a multiplicative value applied to the positive weighting bias used in the loss during training of either the pair, triplet, or both networks. The Hybrid Pair Network performance is indicated with a dotted line.

values for the both-stage models with a corresponding dip in efficiency. However the fake rate notably increases as the multiplier increases, and shows considerably more, and erratic, variation between the different multiplier schemes. Errors similarly increase, and the multiplier schemes always remain within 1σ of each other. This may again be a largely statistical effect as the fake rate gets comparatively closer to 0, as the statistical contributions from any one event can never go below 0.

For all three models with a multiplier of 1, the fake rate appears an outlier. This data point is the regular Basic Model as used previously, as a multiplier of 1 is the same as not having a multiplier at all, which is why all three converge to the same result, which, with the erratic fake rate makes it stand out more. Even still, it is higher than several of its neighbouring data points and has a comparatively larger error. Looking back to Figures 13.5(a) and (b), the Triplet Networks corresponding accuracy and true negative rates do appear to deviate from their neighbours, however compared to the error the deviation is tiny.

In comparison to the Hybrid Model, the only time within the range examined here where the efficiency drops to match is for the variation employing multipliers to both parts of between 0.4 and 0.5. The Clone rate for all multiplier schemes reaches that of the Hybrid Model at low values, though it is so close throughout as to be negligible. For the fake rate, all multiplier schemes generally improve towards that of the Hybrid Model as the multiplier is decreased.

From the results of models in which only the pair stage was altered, it appears that biasing the pair stage to ensure true segments progress to the triplet stage had little or no effect at boosting the efficiency. Rather, the reverse may be true, and that a mild bias to improve the rejection fake segments, potentially in the triplet stage, may be desirable.

13.1.5 Network Configuration

Figure 13.7 illustrates the performance of Basic Pair Networks employing a range of different sized layers and numbers of layers for the fully connected feed forward neural network used within. Neurons per layer indicates the number of neurons in the hidden layers, while the final layer always uses a single neuron in order to produce a single score value for each edge.

The accuracy of the various Basic Pair Networks is erratic, while the balanced accuracy is much tighter between the configurations, with layer sizes of 24 neurons and above particularly close. Generally, we see a slight improvement when more layers are used, and somewhat for larger sized layers. Setups with few layers appear to struggle when partnered with larger layer sizes, and our default Basic Pair Network setup of 3 size 16 layers also struggled compared to the other setups, though with the smallest overall size examined this is somewhat expected. This would suggest that using at least 24 neurons per layer may be an improvement over the default Basic Pair Network setup, but with such close balanced accuracies from there and above, any further increase of either number or size of layers may provide little additional benefit.

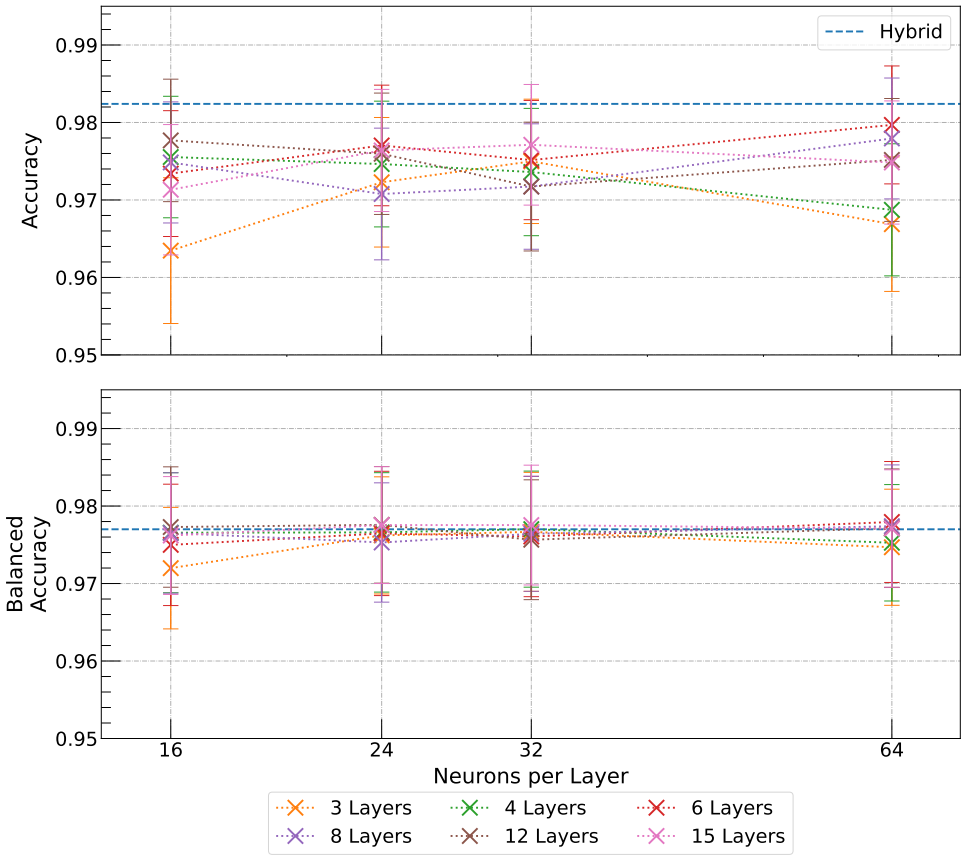


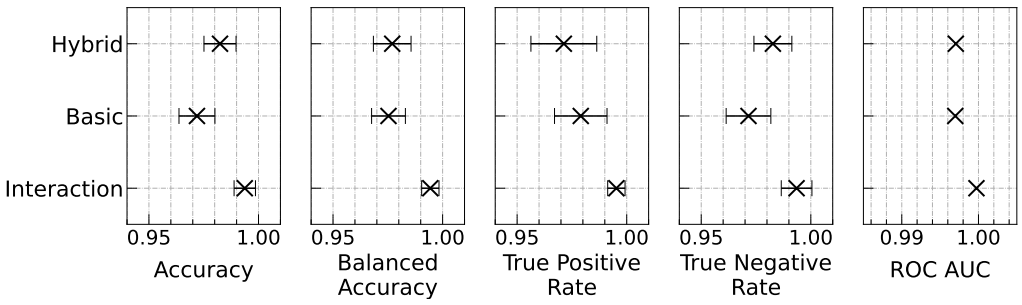
Figure 13.7: Performance of the Basic Pair Network with a range of different size neural network configurations. Neurons per layer indicates the number of neurons in the hidden layers, with the final layer always using a single neuron. The Hybrid Pair Network performance is indicated with a dotted line. Note the horizontal axis uses a logarithmic scale.

13.2 Interaction Network Models

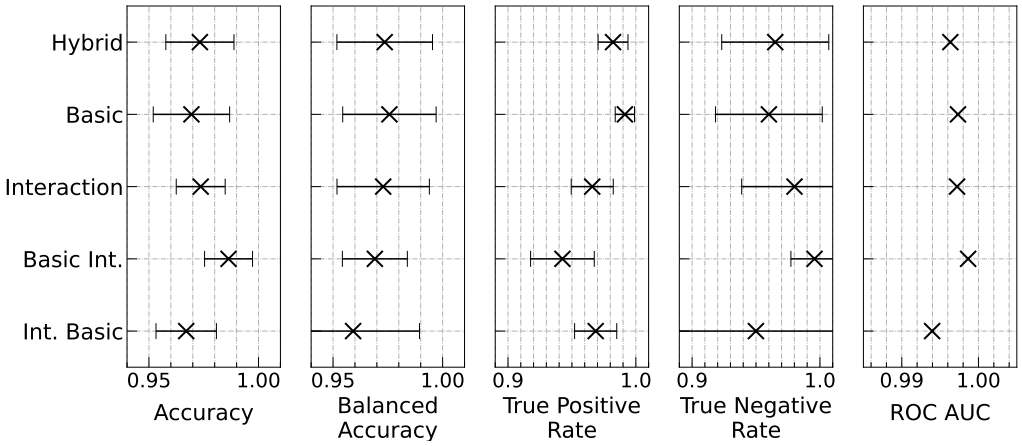
13.2.1 Comparison With the Hybrid and Basic Models

Pair Network

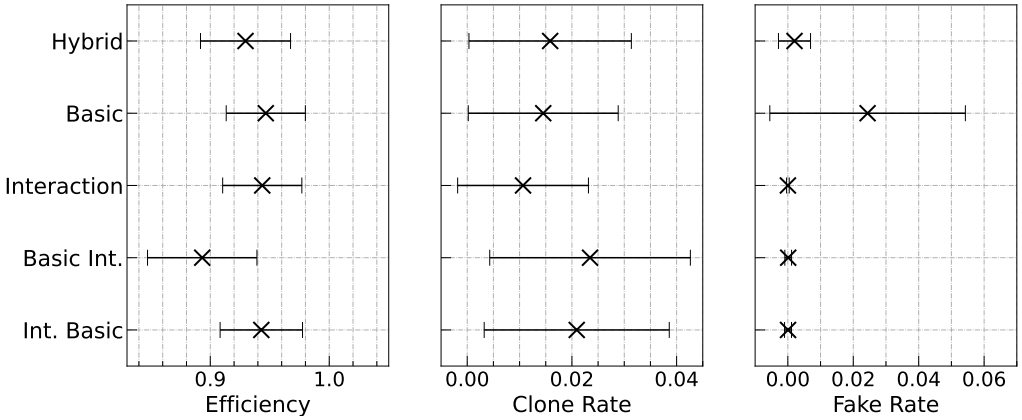
From Figure 13.8(a), we can see that the Interaction Pair Network shows a noticeable improvement in both accuracy and balanced accuracy over the Hybrid and Basic Pair Networks. The Interaction Pair Network true positive rate is a further improvement over the Basic Pair Network, and the true negative rate is better than the Hybrid Pair Network. These results also exhibit comparatively smaller errors, particularly the true positive rate, indicating a more consistent performance. Additionally, while other networks shown so far have had roughly comparable ROC AUC, the Interaction Pair Network is higher by an order of magnitude; from 0.9970 for



(**a**)



(**b**)



(**c**)

Figure 13.8: Performance metrics of the Interaction (a) Pair Network, (b) Triplet Network, and (c) the full model at the complete tracking problem, along side the Hybrid and Basic Models, and respective components. In addition, two mixed models using one of the Interaction or Basic for each stage are included in (b) and (c); where the two names given indicate the networks used for the Pair and Triplet stages respectively.

the Hybrid and regular Basic Pair Networks, to 0.9997. Together this suggests the Interaction Pair Network presents an improvement in both identifying parts of real tracks and discarding false segments.

Triplet Network

However, the Interaction Triplet Network, as shown in Figure 13.8(b), exhibits comparable accuracy and balanced accuracy to the Hybrid Triplet Network. Breaking this down, the true positive rate is noticeably worse than either model, with a slightly improved true negative rate. Unlike with Basic or Hybrid Pair Networks, the strong performance of the Interaction Pair Network leads to positive weighting under 1 when training subsequent Triplet Networks; inverting the impact of the true positive and negative rates on balance accuracy.

As mentioned in Section 13.1.1, significant differences in Pair Network performance can cloud comparison between Triplet Networks. Therefore to compare the Basic and Interaction Triplet Networks more directly, a model was trained using the Basic Pair Network for the pair stage, but with the Interaction Triplet Network for the triplet stage, and is indicated as Basic Int. This model returned the best accuracy of any triplet network examined here, but peculiarly a balanced accuracy lower than either of the three previous models. From the true positive rate, the network struggled noticeably at identifying true track segments among those returned by the Interaction Pair Network, possibly a consequence of a strong performance at identifying and excluding false segments in the previous stage. However, a high true negative rate, and with a comparatively small error, indicates the network outperformed the full Basic Model at classifying false segments in this stage, though well within 1σ error.

Additionally, an inverse version in which a Basic Triplet Network was coupled with the Interaction Pair Network was trained, and is indicated as Int. Basic. This presented a worse accuracy and balanced accuracy than using the regular Basic Triplet Network, with a poor true positive rate as with the full Interaction version and a slightly lower true negative rate coupled with a significantly larger error. It also gave a noticeably lower ROC AUC value.

Overall, these results suggest the Interaction Triplet Network struggles compared to the Hybrid or Basic Triplet Networks, regardless of what Pair Network it is partnered with. Though the Interaction Networks feature more layers overall than the Basic Networks, the layers themselves contain less neurons and are spread across subcomponent neural networks, which may not be sufficient when encountering the

triplet classification task; perhaps as the triplet task involves more input variables due to encompassing three hits.

Full Model

Performance of the Hybrid, Basic and Interaction Models at the full tracking problem, along with the mixed network models, is given in Figure 13.8(c). Noticeably, the Interaction Model produced a comparatively negligible fake rate of 0.000019 ± 0.000404; two orders of magnitude smaller than the Hybrid Model. This is accompanied by a slightly improved clone rate, and an efficiency close, but slightly less than, that of the Basic Model, with both well within 1σ. While the full Interaction Model generally appears an improvement over the other models, the Pair Network's strong true positive rate is not translating through to an improved efficiency or a particularly significant reduction in clone tracks.

Using the Interaction Pair Network with a Basic Triplet Network ultimately performed worse than the complete Interaction Model in all three metrics. On the other hand, using a Basic Pair Network with the Interaction Triplet Network provided a similar efficiency and fake rate to the complete Interaction Model, but also a slightly higher clone rate.

13.2.2 Message Reduction Function

Figure 13.9 describes the performance of a set of Interaction Pair Stage Networks using a variety different aggregation functions to carry out message passing reduction process, as described in Section 12.3.2; in which the sum, mean, maximum or minimum of respective features of the incoming edge attributes is taken. All four aggregation functions examined performed remarkably similarly. Naïvely, we would have expected the choice of function to have been more impactful, and perhaps that summation, which is used throughout the other Interaction Networks presented, may have performed the worst, as it is at the mercy of variations in the number of hits.

While companion Interaction Triplet Network were trained using the same set of aggregation functions, the mean, maximum and minimum variants consistently converged to predicting true for all segments presented, as mentioned in Section 12.3.3.

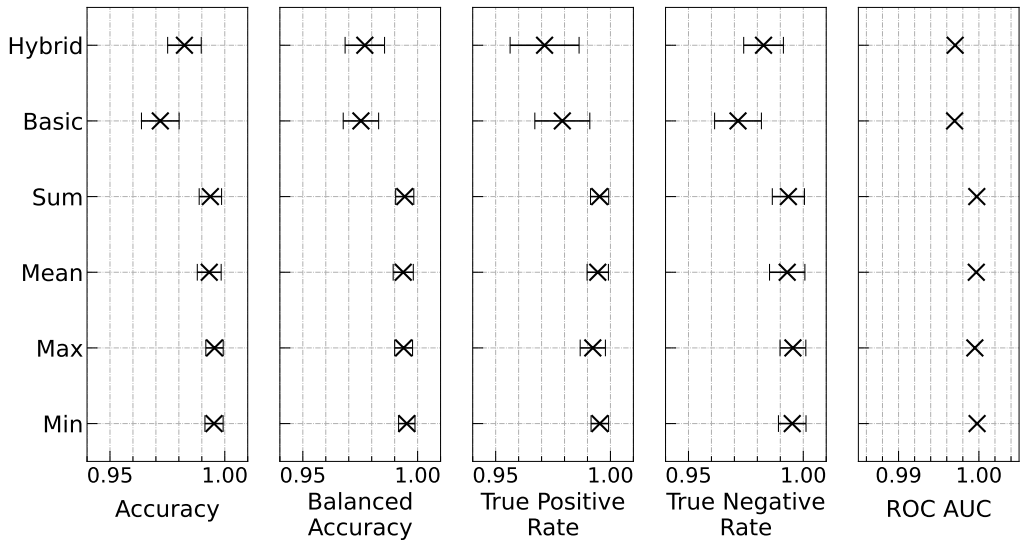


Figure 13.9: Performance metrics of the Interaction Pair Stage Network using different message reduction functions, alongside the Hybrid and Basic Models.

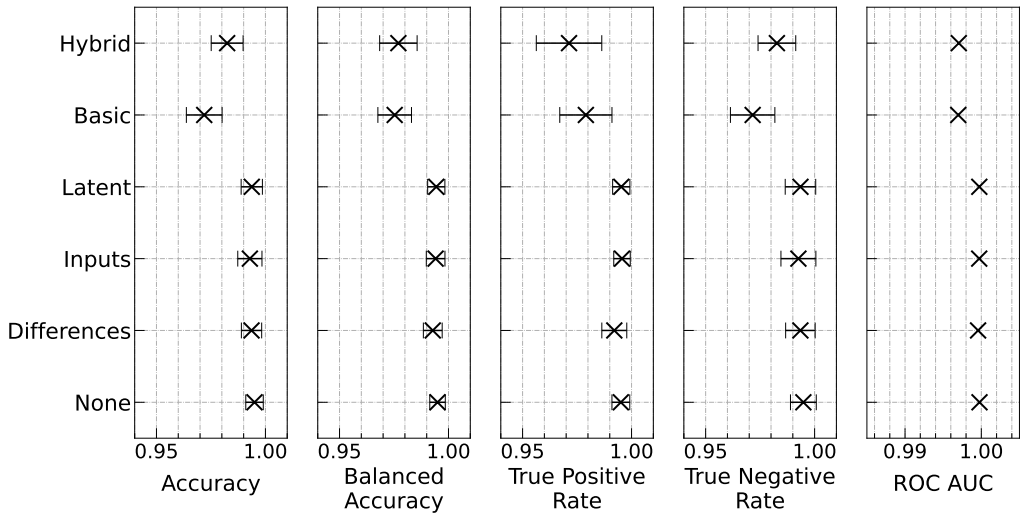


Figure 13.10: Performance of the Interaction pair stage network with different skip connections, alongside the Hybrid and Basic pair stage networks for comparison.

13.2.3 Skip Connection

Several Interaction Pair Networks using alternative forms of skip connection were examined, within which additional inputs are passed to the edge and vertex update networks from an earlier part of the overall network, with the results shown in Figure 13.10. In the default configuration, denoted as latent, the initial latent features as produced by the encoder subnetworks are passed as additional inputs. Alternatively, in the variant denoted as inputs, the original features taken as inputs from the edges or vertices are passed, bypassing the encoder subnetworks. Denoted as differences, another instead passes the difference between the output of the previous iteration, which the update subnetworks receive as their usual inputs, and the inputs that previous iteration received. For the first iteration, 0's are passed. Finally, a variant not using any skip connection was also trained, denoted as none. In this case, the edge and vertex update subnetworks have a smaller number of inputs, as there are no skip connection inputs.

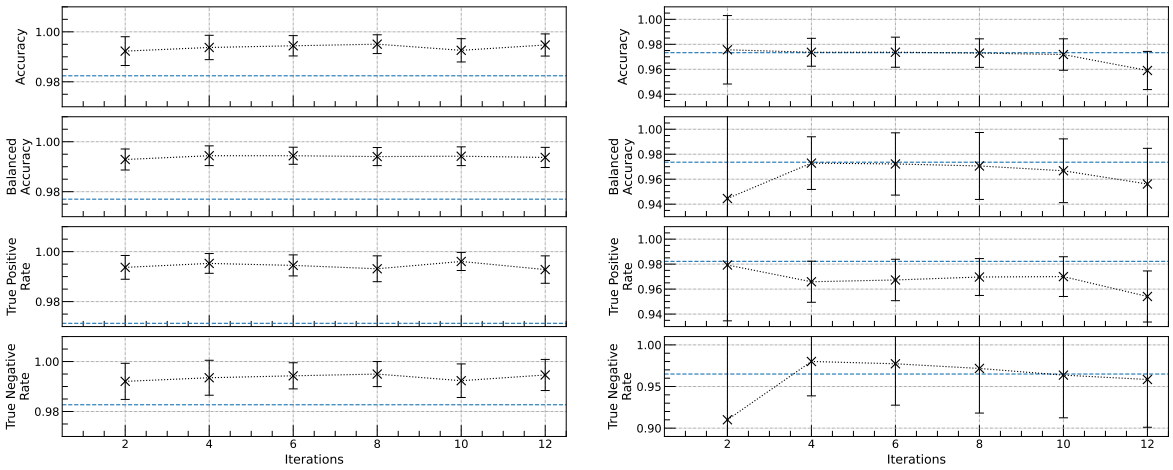
All four forms of skip connection led to remarkably consistent results. Using the original inputs showed a marginally lower and larger error in the true negative rate, leading to a small decrease in accuracy. Due to the ambiguity in ordering when considering reverse direction edges, we would naïvely have expected a larger drop in performance. The use of differences led to a drop in the true positive rate, particularly impacting the balanced accuracy. Not using a skip connection if anything showed a small improvement in true negative rate, leading to slightly improved accuracy. This suggests that the skip connection, for the Pair Network at least, is not providing a meaningful contribution, and so gives a potential direction to reduce computational costs by removing it.

13.2.4 Iteration Section

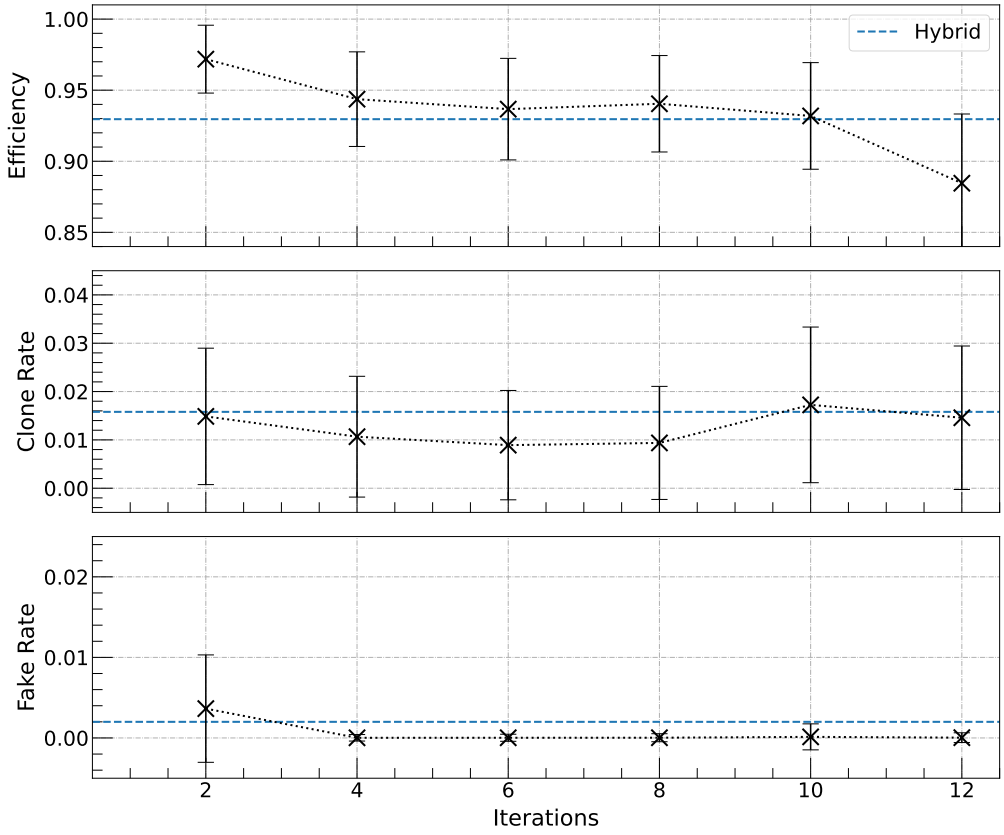
Several Interaction Models were trained with the iterative section performed a varying number of times, with the results displayed in Figure 13.11. Within the Interaction Model, the iterative section is where the message passing paradigm occurs, with each iteration essentially propagating information one edge further. Each model used the same number of iterations for both Pair and Triplet Networks.

Pair Network

From Figure 13.11(a) we can see the Interaction Pair Networks performed consistently to one another, with some variation in accuracy owing to fluctuations between



(a) (b)



(c)

Figure 13.11: Performance metrics of the Interaction (a) Pair Network, (b) Triplet Network, and (c) the full model at the complete tracking problem with the iterative section performed a variable number of times. Each variant was separately trained, and the corresponding Hybrid Pair Network performance is indicated with a dotted line.

the true positive and true negative rates.

Triplet Network

However, as can be seen in Figure 13.11(b), the corresponding Interaction Triplet Networks exhibited far more variety in performance. Using only 2 iterations performed comparatively poorly, and with all metrics exhibiting significantly larger errors. While demonstrating the highest true positive rate among the Triplet Networks, and a marginally higher accuracy, a low true negative rate significantly brings down its balanced accuracy; its difficulty in identifying fake track segments washing out any gain from identifying true segments.

Going to 4 iterations and above, as the number of iterations increases the true positive rate slowly improves, while the true negative rate tends steadily downwards; resulting in a consistent accuracy, but a slowly dropping balanced accuracy, though the differences are relatively small. However at 12 iterations, the true positive rate drops sharply, bringing both the accuracy and balanced accuracy down. Though 2 iterations appears to be insufficient for useful information to spread, too many iterations also causes the Interaction Triplet Network to struggle, perhaps washing out local information.

Full Model

In the full model results, given in Figure 13.11(c), we can see the behaviour of the Triplet Network continuing through. At low iterations, efficiency is improved, but clone and fake rates also rise. With high numbers of iterations, efficiency begins to drop and clone rate increase, suggesting tracks are being split by missing segments.

13.2.5 Network Configurations

Pair Network

Figure 13.12 describes the performance of several Interaction Pair Networks using a range of different sized layers and numbers of layers for each of the five subcomponent fully connected feed forward neural networks employed within. Neurons per layer indicates the number of neurons in the hidden layers, while the final layer always uses the number of neurons needed to produce the required number of output values.

The various Pair Networks shown gave fairly consistent results. As the number of neurons per layer increases, configurations employing 3 layers displayed marginally

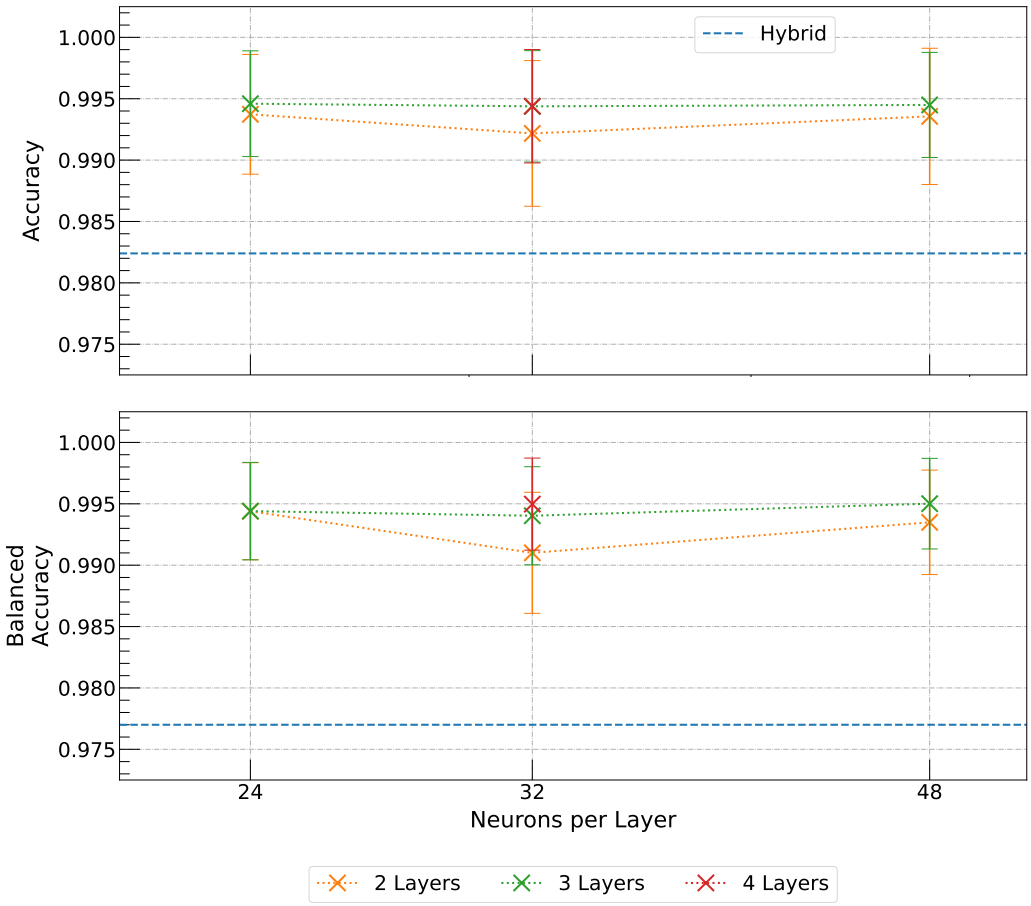


Figure 13.12: Performance of the Interaction Pair Network, using a range of different configurations for each of the five subcomponent neural networks within. Neurons per layer indicates the number of neurons in the hidden layers, with the final layer always using the required number of outputs. The Hybrid Pair Network performance is indicated with a dotted line. Note the horizontal axis uses a logarithmic scale.

higher performance than using 2, though they all still remain well within 1σ of each other. The divergence in behaviour between the two with 32 neurons per layer is a peculiar outlier, but again still within 1σ. The single 4 layer configuration presented closely matches the performance of the largest 3 layer configuration, though with only 32 neurons per layer compared to 48. Additional 4 layer configurations were trained, along with 6 configurations. However it was later realised they had failed to train satisfactorily and would continually predict all edges as part of a track; and so these models are not presented.

Overall, increasing the size and number of layers might offer some improvement over the default Basic Pair Network configuration, but only marginally. Equally as the Pair Network employs five subcomponent neural networks, this has a significant

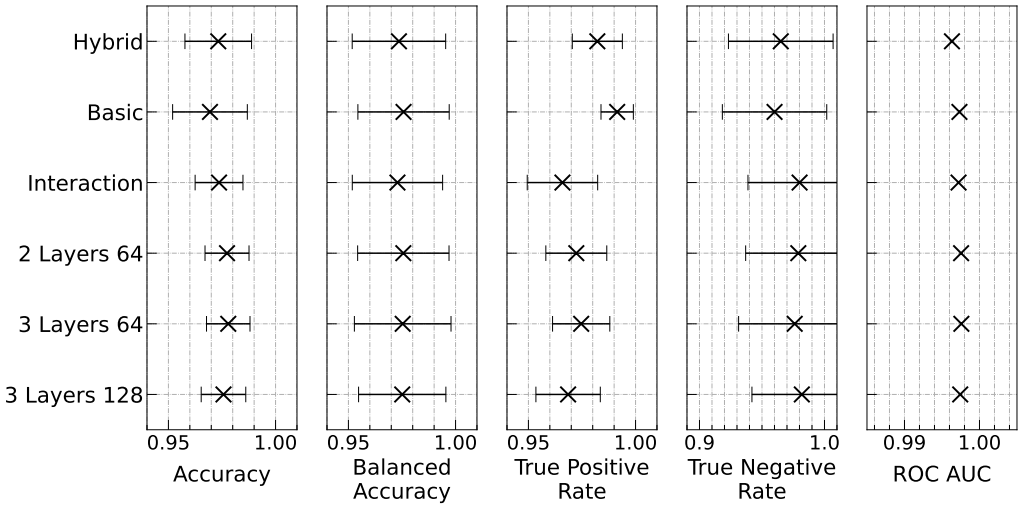


Figure 13.13: Performance metrics of the Interaction Triplet Stage Network, using a range of different configurations for each of the five subcomponent neural networks within, along side the Hybrid, Basic and default configuration Interaction Triplet Networks. a layers b denotes the use of a layers, with b neurons in the hidden layers, with the final layer always using the required number of outputs.

impact on the overall size of the model; and given the failure of the larger models trialled here, more neurons may potentially be a cause of, or exacerbate, issues in training.

Triplet Network

To accompany Section 13.2.1, several Interaction Triplet Networks were created using larger network configurations, closer to those used in the Basic Triplet Network. These were all trained in conjunction with the default Interaction Pair Network, and their performance is described in Figure 13.13.

Increasing the number of neurons in the hidden layers brought an improvement to the true positive rate, while the true negative rate dropped only very slightly; leading to an improvement in both the accuracy and balanced accuracy. Adding an additional hidden layer left the accuracy and balanced accuracy relatively unchanged, the true positive and negative rates marginally increasing and decreasing respectively. Further doubling the neurons per layer caused the true positive rate to drop, and the true negative rate to rise, leading to a small drop in the accuracy. Nevertheless, any changes were small compared to the errors, so the significance of any improvement or decline is relatively minor.

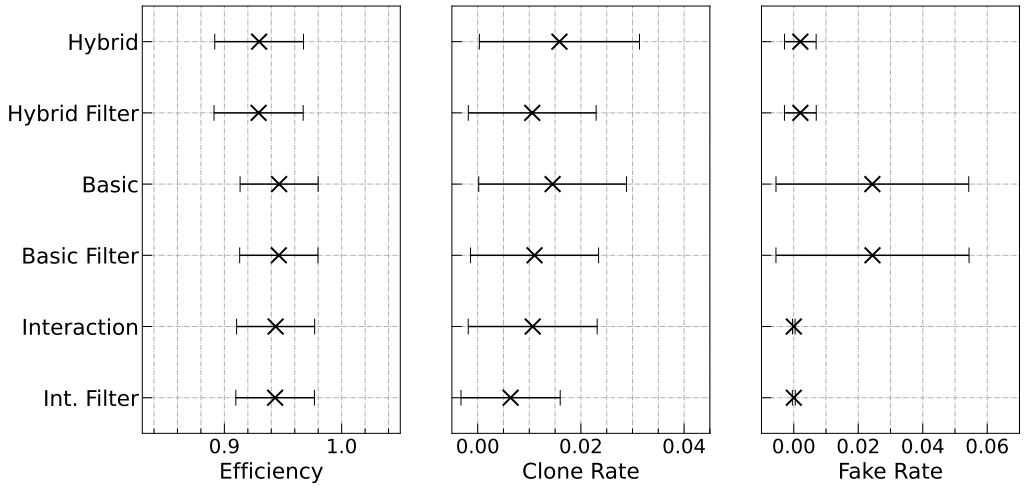


Figure 13.14: Performance metrics of the full Hybrid, Basic and Interaction Models at the complete tracking problem, both with and without the use of an additional missing segment filter.

13.2.6 Missing Segments Filter

The full Hybrid, Basic and Interaction Models were also evaluated with the inclusion of a missing segments filter, designed to correct for tracks that had been split by a missed segment, as described in Section 12.3.1. As the filter is applied at the end of the full tracking stage, the same component networks were used.

From the results shown in Figure 13.14, we can see that in all three models the filter was successful in reducing the number of duplicate tracks. One concern was that the filter may reduce a model's efficiency, by effectively overriding where a model had identified two distinct tracks. However this appears not to be the case, and the efficiency of all three models remains constant. Similarly the fake rate is essentially uneffected by use of the filter.

Chapter 14

Discussion

With models such as that in [270] and the Hybrid Model focusing on augmenting approaches similar to that currently employed by the detector, our aim was to develop a machine learning based pattern recognition with the capacity to draw inferences from an event as a whole. This led us to explore a Graph Neural Network based approach, which through the message passing paradigm could conceivably be sensitive to all hits within an event when making its assessments; elevating an otherwise local approach into a global one. Noting the success of an Interaction Network architecture[153] in context of the ATLAS experiment[180,181,182], this lead to implementing a Interaction Network based model, which demonstrated potential for improved performance over a non-graph based approach.

Given it served as the basis upon which our overarching framework was designed, an existing neural network model for VELO tracking, referred to here as the Hybrid Model, was utilised as a baseline against which to evaluate performance. Overall, the Basic graph model performed on a similar, abet marginally worse, level to the Hybrid model. The two neural network stages traded a better performance at identifying true track segments for worse at false segments, translating for the full model into slightly improved efficiency and reduction in duplicate tracks, but at the cost of an increase in fake tracks. Given that the Basic Model was designed to intentionally emulate the actions of the Hybrid Model, but in the context of a graph based representation of event data, similar behaviour was to be expected; significant differences would indicate issues in the Basic model's implementation. Neither the use of separate networks for each module combination, nor introducing a bias to the loss weighting to approximate the sampling based learning method used in training the Hybrid networks, was able to adequately account the differences in balance between true positive and true negative performances of the component stages; through, these

two proposals were not examined together. Beyond that, results suggest a multiplier of .5-.8 may be preferential to perfect balance in order to keep the fake rate down.

The variations we examined generally produced only small deviations in performance. Increasing the complexity of the component network architectures provided little improvement, though in the context of a model encompassing the full 52 modules it appears increasing the size of the networks may prove more impactful. Biasing the loss during training to favour identification of false edges seems potentially beneficial in reducing the fake rate, and excluding the z coordinate as an input performed unexpectedly well.

The Interaction model on the other hand showed a tentative improvement over the Hybrid and Basic models with a significant reduction in the number of fake tracks, though coupled with only a slight improvement in reducing duplicates, and a very slightly worse efficiency over the Basic Model. The Interaction Network notably outperformed both compared models when used for the pair stage, but struggled as the triplet stage. From the worse performance of using the Basic Network for the triplet stage with the Interaction Pair Network, we can infer that this was at least somewhat down to the Interaction Pair Network's strengths providing a more difficult dataset. Increasing the size of the component neural networks for either stage had relatively limited effect, and potentially presented training issues.

Altering or removing the skip connection had little effect on the Interaction Pair Network performance, suggesting that this element contributed little to the improved performance over the Basic Pair Network. Equally using different message reduction functions did not make a particular impact. Naïvely we expected summation to have perform worse, however the Interaction Network based model examined for ATLAS track finding in [181] also employed summation, and in our context it proved the more stable for training the triplet stage network. The number of iterations performed did have a noticeable impact on performance of the Triplet stage, and subsequently on the final tracking, suggesting that message passing indeed played a role in the Interaction Pair Network's strong performance, and that too few or too many iterations would lead to degraded performance.

Including an additional stage to look for broken tracks had the potential to decrease the efficiency, through amalgamating two otherwise correct and separate tracks. However for the settings trialled this was not the case, and it successfully reduced the number of duplicate tracks. Although these trails offer some insight into optimising the configuration of the Interaction Networks, most aspects were only examined in isolation. It is plausible, for example, that the use of a skip connection

may have a greater effect when more iterations are used, as current features become further and further disconnected from their original meanings and values in later iterations.

14.1 Limitations

A significant limitation of our results lies with the relative size of errors compared to the differences in performance between models. In many cases, differences are well within 1σ of each other, and so any conclusions drawn must be tentative. Given the naïve way in which our errors are calculated (see Section 12.4.1), an increased sample size may not alleviate this. The spread characterised by σ is due at least in part to the variation in performance between different events, not of certainty in measurement of said performance. Increased statistics can provide a more accurate picture of this spread, but does not change it. Further work would likely be better served in seeking alternative approaches to assessing the accuracy of such performance measurements. Ultimately though, a significant factor in judging performance, both between variations and in general, lies in the relative importance placed on the efficiency, clone and fake rates to one another. Many of the parameters varied in these studies shifted the balance of the pair and triplet stage networks' performance between identifying true and fake track segments.

In hindsight, it would have been informative to have followed the lead of other studies on VELO pattern recognition algorithms, and have evaluated our performance on important subsets of tracks, such as those within the detector's nominal $2 < \eta < 5$ acceptance, or on events of particular interest, such as those featuring B hadrons. While we have focused on the VELO, LHCb overall is a forward arm detector, and an algorithms performance at certain tracks is natural of greater interest to its physics program. Equally, it would have been beneficial to compare against the current pattern recognition stage algorithm employed by LHCb, applied to our datasets, but which unfortunately proved unfeasible.

Though the majority of models presented here were restricted to only half the detector, those trained and evaluated on the full 52 modules performed relatively consistent to their 26 module counterparts. Thus the results here can be considered at least indicative of full detector performance, which is sufficient for a proof-of-concept. However, while our models considering the full 52 modules accounted for tracks involving hits between modules on opposite sides, we failed to account for tracks, if in a minority, with hits on both modules of the same station, due to angled

trajectories or the overlap between the modules sensitive areas.

Many of our studies focused on the Pair Network stage. While it could be considered indicative of a potential triplet stage implementation of the same model, the performance of the triplet stage is arguably a more important consideration, as its output is that subsequently used to construct tracks. While the triplet stage was dependant on the choice of pair stage employed in training, as the first step the pair stage was more convenient to experiment with. Then again, we focused predominantly on modules using the same configuration for both stages, but this need not be the case. Not only do the two component models not need to be the same, they can take completely different forms, a direction we did not fully explore.

As discussed in Section 12.3.3, issues were identified relating to the post-epoch validation loss used to gauge a model's performance. Early tests indicate this is a technical issue than could be solved with further work. Separate unresolved issues were encountered in training the Interaction Model for use in the triplet stage, suggesting the network is becoming stuck in a local minima. From issues encountered in training the Interaction Model for the pair stage, using additional layers and neurons per layer, there are tentative indications that the overall number of layers or neurons, or for a specific component network, might have a bearing on this. Optimisation of the training process was an area we did not greatly explore after establishing a working approach with the Basic Network. Given the issues encountered, lowering the learning rate or implementing a steadily decreasing learning rate, are among other potential directions that may prove beneficial. Equally, the issue may simply lie with our implementation, though efforts were of course taken to hunt for corresponding issues in our code.

14.2 Execution Time

While the results presented here suggest a Graph Neural Networks based approach to tracking finding within the VELO may offer improved reconstruction quality, there is a significant issue we have so far not discussed. Due to the high throughput requirements brought by LHCb's move to full online reconstruction, current implementations of machine learning are limited to simpler methods such as fully connected feed forward neural networks and boosted decision trees. Though research with more advanced approaches is ongoing, GNN are generally slower than other more direct methods, and the need for graph construction, twice, in our model caries a significant time cost [63].

We approached the problem of applying GNN to VELO tracking without an architecture initially in mind, and so our implementation focused on providing a flexible system with which to explore many models without significant rewriting of code. A dedicated model could be implemented in a significantly more efficient manner than was used, and given this we did not look to make dedicated measurements of execution time. For example, due to the computing packages used, all graphs were implemented as directed graphs. For models seeking to exploit message passing, as reverse edges have essentially duplicate features, an 'effective' undirected graph might be used to reduce the size of an event, identifying when an edge is to be used in the reverse direction and performing any necessary modifications as they are read. However, this would be a material change for any model using the message passing principal, as after the first application of a component network to any edge, the updated features on opposite directed edges need not correspond to one another; though the approach could be used only at the beginning to minimise the data stored at that time. Going further, the nature of our data means we know the rough structure and edge relations a graph representation will take; edges do not, for example, have the potential to connect just any vertices, but must follow the station structure of the VELO. By employing this domain knowledge, it may be possible to further translate the problem into a more efficient form than an explicit graph. Ultimately, what matters is how the data is handled and the calculations carried out, not that it is done in a form which can readily be interpreted as a graph. Looking to potential hardware, there has already been efforts to implement Interaction Network style models for tracking on FPGAs[185].

Nevertheless, GNN are still relatively slow in comparison to other neural network methods, and our algorithm was not designed with efficiency in mind; significant development and refinement would be needed to achieve viable timing efficiency.

One approach used in various existing algorithms, such as in [30], is the introduction of a search window in ϕ, limiting the combinations of hits considered to within a range of ϕ of one another. In our context, this would amount to constructing edges in the pair graph only between vertices representing hits within the window, or pairs with comparable slopes in the triplet graph. However, such windows are usually used for seed tracks, rather than on all potential pairings, with separate windows calculated when looking to extend each seed to hits on another station; while graph construction is carried out across the length of the VELO. Then again, the surprisingly small effect that excluding z as an input had on the Basic Network performance suggests that location along the detectors length is not of particular

importance. The significant issue is that this approach has the potential to exclude highly inclined tracks who, if while small in number, are important to LHCb's physics program [29]. A potential alternative would be a nearest N approach, connecting a vertex to its nearest neighbours in ϕ each way, instead of a window. This does however carry a similar potential to exclude correct partners in highly dense regions.

Another direction would be to consider a more hybridised approach, using the model to construct pair seeds and forwarding them as is performed in other algorithms; relying on the improved performance of the Interaction Network in the pairing stage alone and circumventing construction of a triplet graph. Alternatively, the triplet stage could be performed without the need for a triplet graph by using a more basic neural network as in the Hybrid Model.

14.3 Potential Directions

Given the number of parameters within and other variations possible with the Interaction Model, it has not been feasible to investigate all possibilities. As mentioned before, we did not particularly examine optimisation of our training process. Neither did we examine the variations of the final construction of tracks, such as the thresholds for combining triplets. As discussed previously, we often focused on using the same configuration for both the Pair and Triplet stages. The neural networks within the Interaction Model architecture itself do not need to be the same either; as neural networks can be both too big or too small for a given task, so too may the various components networks the have different optimum architectures.

In terms of the Interaction Model implemented here, there are numerous further directions that can be explored. The original interaction network model [153] allows for a global attribute, enabling general information on a given graph to be encoded and utilised. As there were no clear properties which may have proved useful, this was not included in our implementation. However, some form of event characterising information could prove useful to pattern recognition, perhaps detector configuration information, or potentially using a graph classification neural network to generate a set of characteristic latent features without having to explicitly identify what they are.

To ensure consistency, some edge features in the graphs were swapped or inverted compared to their opposite direction partner. But alternatively, using our knowledge of how our graphs will always be structured, the aggregation of incoming edge attributes could be split in two; with edges connected to vertices representing up-

stream and downstream hits aggregated separately and used as separate inputs to the vertex update network. This would not only sidestep the issue of needing to alter features so that the edge update network does not see edges 'backwards', but indirectly incorporate the domain knowledge distinction between modules up and down stream of each other. It would even be possible to utilise separate network instances of the edge update network for upstream and downstream directions, though it is doubtful, given the nature of tracks, that this would be advantageous.

Though we implemented a track forwarding style filter stage to reduce clone tracks, other potential post-track construction extensions are possible, and existing algorithms offer various examples. However, given the increased calculations such methods bring, filtering methods to reduce combinatorics prior to graph construction, such as the ϕ window or N nearest neighbours approaches discussed earlier, are likely of more interest initially.

In the context of tracking within the VELO, the complexity, and so required calculations, introduced by a GNN method over simpler architectures may not be worthwhile under current constraints for straight tracks uninfluenced by a magnetic field. However, as demonstrated in [182], a graph is not limited to representing data solely within a single tracking subdetector. A logical extension would be to consider if a graph and GNN based method could instead perform pattern recognition across the entire LHCb tracking system. This would in a way sidestep some of the cost of constructing graph representations, as such an algorithm would be in place of track finding across multiple subdetectors. One consideration is that doing so would naturally increase the size of graphs, and similarly calculations performed over them.

Going further, GNN have been adapted to operate on heterogenous graphs [284,285]; graphs which contain distinct types of edge and vertex. Each type of edge or vertex possesses its own form of attribute, with features representing different properties. This conceivably opens up the more radical possibility of incorporating multiple forms of measurement from non tracking subdetectors into a single graph representation of an event, such as calorimeter measurements. In [187], a GNN model, explored in the same paper for track finding in tracking, was modified to not only perform calorimeter clustering, but identify particle type at the same time [187]. With this in mind, it is conceivable that, through incorporating the measurements of multiple subdetectors into a heterogenous graph, one could potentially incorporate other event reconstruction tasks into a GNN based algorithm of some sort; or potentially boost track reconstruction efforts. Indeed, a study along these lines has already been published in [187]. Combining tracking, electromagnetic and hadron calorime-

ter measurements into a single heterogeneous graph representation, they explored the potential of a general purpose GNN for event reconstruction [162,183].

It is also worth considering that, rather than an edge classification task, track finding can instead be interpreted as a segmentation problem. Image segmentation is a well-trodden field in computer vision, and with the relation between CNN and GNN there is a growing interest in graph segmentation [286], and already there have been moves to approach tracking in this manner [162,287].

14.4 Related Projects

In recent years, a formal project to adapt the Exa.TrkX tracking pipeline (see Section 8.3.3) to the LHCb VELO has independently been underway. Named ETX4VELO, the model utilises nearest neighbour-based graph construction, and implements additional stages with respect to the Exa.TrkX model [63,288]. An exploration of ETX4VELO performance can be found in [288], and documentation on the model can be found at [289].

Part IV

Conclusion

Chapter 15

Conclusion

Fundamental particles are small, so small we must rely on specialised machines to examine what takes place at their scale. Among these, tracking detectors allow us to pinpoint particles' locations through their interactions. Though research has often driven their development, particle trackers can be found across a broad range of practical applications. But even possessing the tools to make such measurements, we still need to interpret our observations in order to understand their meaning, and deduce what has taken place.

Able to learn patterns from data without explicit prior knowledge and capable of considerable computational speeds compared to more direct methods, neural networks are eminently suitable as a tool for data-handling tasks, and are thus seeing increasing use in cutting edge research. Not so much a single model but a core concept and common elements, neural networks are a versatile tool than can be employed in many forms. With this in mind, we undertook, and have explored here, a pair of projects looking to leverage neural network's strengths within the context of particle tracking.

While proton computed tomography has great potential for sidestepping issues with x-ray based treatment planning in proton therapy, a long standing problem lies in the need to reconstruct proton tracks individually, carrying a significant computational burden. Through the use of a neural network, we successfully demonstrated that machine learning could be leveraged to match, and in some cases exceed the accuracy of the standard algebraic method, and do so at a significantly shorter time scale.

Within the LHCb experiment sits the VELO, a cutting edge tracking detector nestled immediately around the interaction point itself. Making measurements vital to the experiments event reconstruction efforts, individual particle tracks must be

distinguished from among the multitude of measurements made in each collision event. Seeking a neural network model with the capacity to draw inferences from an event as a whole, we implemented a proof-of-concept graph neural network based design. This approach showed potential improvement in reconstruction quality over existing trials. It comes at the cost of computational demand, but one which will likely shrink as technology and GNN methods develop further.

Though both lie firmly within the domain of tracking, they present fundamentally different tasks, and thus lend themselves to different approaches. In pattern recognition, we are presented with analysing a field of individual measurements to deducing those made by a common particle, a form of classification or segmentation task. On the other hand, predicting a particle's path given its end points is an interpolation task, and in many ways more akin to track fitting in an event reconstruction context. Nevertheless, with the versatility of neural networks and myriad of approaches available, we have demonstrated they can be useful tools with which to address these different challenges. Given machine learning, and neural networks in particular, is an active and rapidly developing field, with the aid of research, such as that detailed here, we will doubtlessly see many new and varied applications developed for particle tracking, fundamental research and beyond.

List of Figures

List of Tables

Bibliography

[1] Robert P Johnson. Meeting the detector challenges for pre-clinical proton and ion computed tomography. *Physics in Medicine & Biology*, 69(11):11TR02, may 2024. doi: 10.1088/1361-6560/ad42fc.

[2] KM Hanson, JN Bradbury, RA Koeppe, RJ Macek, DR Machen, et al. Proton computed tomography of human specimens. *Physics in Medicine & Biology*, 27(1):25, 1982.

[3] T. Ackernley, G. Casse, and M. Cristoforetti. Proton path reconstruction for proton computed tomography using neural networks. *Physics in Medicine & Biology*, 66(7):075015, apr 2021. doi: 10.1088/1361-6560/abf00f.

[4] *LHCb : Technical Proposal*. CERN, Geneva, 1998.

[5] Roel Aaij, Bernardo Adeva, Marco Adinolfi, Anthony Affolder, Ziad Ajaltouni, et al. LHCb Detector Performance. *Int. J. Mod. Phys. A*, 30:1530022, 2015. doi: 10.1142/S0217751X15300227.

[6] Roel Aaij, Bernardo Adeva, Marco Adinolfi, Christine Angela Aidala, Ziad Ajaltouni, et al. Physics case for an LHCb Upgrade II - Opportunities in flavour physics, and beyond, in the HL-LHC era. Technical report, CERN, Geneva, 2016. ISBN 978-92-9083-494-6.

[7] Hermann Kolanoski and Norbert Wermes. *Particle Detectors: Fundamentals and Applications*. Oxford University Press, 06 2020. ISBN 9780198858362. doi: 10.1093/oso/9780198858362.001.0001.

[8] E. R. Davies. Role of photography in the detection and measurement of radiation. *Nature (London)*, 149(3781):430–432, 1942. ISSN 0028-0836.

[9] Nadine Barrie Smith and Andrew Webb. *Introduction to Medical Imaging:*

Physics, Engineering and Clinical Applications. Cambridge Texts in Biomedical Engineering. Cambridge University Press, 2010.

[10] Claus Grupen. *Introduction to Radiation Protection: Practical Knowledge for Handling Radioactive Sources*. Graduate Texts in Physics. Springer Nature, Berlin, Heidelberg, 1 edition, 2010. ISBN 3642025862.

[11] Irka Hajdas, Philippa Ascough, Mark H. Garnett, Stewart J. Fallon, Charlotte L. Pearson, et al. Radiocarbon dating. *Nature Reviews Methods Primers*, 1, 62, 2021.

[12] United States Enviromental Protection Agency. Neutron borehole logging, 2025. https://www.epa.gov/environmental-geophysics/neutron-borehole-logging, Accessed 08/05/2025.

[13] Are Strandlie and Rudolf Frühwirth. Track and vertex reconstruction: From classical to adaptive methods. *Rev. Mod. Phys.*, 82:1419–1458, May 2010. doi: 10.1103/RevModPhys.82.1419.

[14] Fabio Sauli. From bubble chambers to electronic systems: 25 years of evolution in particle detectors at cern (1979–2004). *Physics Reports*, 403-404:471–504, 2004. ISSN 0370-1573. doi: https://doi.org/10.1016/j.physrep.2004.08.023.

[15] Fabio Sauli. *Gaseous Radiation Detectors: Fundamentals and Applications*. Cambridge Monographs on Particle Physics, Nuclear Physics and Cosmology. Cambridge University Press, 2014.

[16] Roger Bouclier, M Capéans-Garrido, C Garabatos, G Manzin, Gilbert Million, et al. On some factors affecting discharge conditions in micro-strip gas chambers. *Nucl. Instrum. Methods Phys. Res., A*, 365:65–69, 1995. doi: 10.1016/0168-9002(95)00453-X.

[17] F. Sauli. Gem: A new concept for electron amplification in gas detectors. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 386(2):531–534, 1997. ISSN 0168-9002. doi: https://doi.org/10.1016/S0168-9002(96)01172-2.

[18] Upgrade of the ALICE Time Projection Chamber. Technical report, 2013.

[19] Robert Münzer and The ALICE Collaboration. Upgrade of the alice time projection chamber. *Nuclear Instruments and Methods in Physics Research*

Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 958:162058, 2020. ISSN 0168-9002. doi: https://doi.org/10.1016/j.nima.2019.04.012. Proceedings of the Vienna Conference on Instrumentation 2019.

[20] B.T. King, T. Albahri, S. Al-Kilani, D. Allspach, D. Beckner, et al. The straw tracking detector for the fermilab muon g-2 experiment. *Journal of Instrumentation*, 17(02):P02035, feb 2022. doi: 10.1088/1748-0221/17/02/P02035.

[21] Technical Design Report for the Phase-II Upgrade of the ATLAS Muon Spectrometer. Technical report, CERN, Geneva, 2017.

[22] The Phase-2 Upgrade of the CMS Muon Detectors. Technical report, CERN, Geneva, 2017.

[23] Roel Aaij, Ahmed Sameh Wagih Abdelmotteleb, Carlos Abellan Beteta, Fernando Jesus Abudinén, Cyrille Achard, et al. The LHCb upgrade I. *JINST*, 19(05):P05065, 2024. doi: 10.1088/1748-0221/19/05/P05065.

[24] ALICE collaboration The. Technical Design report for the ALICE Inner Tracking System 3 - ITS3 ; A bent wafer-scale monolithic pixel detector. Technical report, CERN, Geneva, 2024.

[25] James R. Janesick. *Scientific Charge-Coupled Devices*. 2001. ISBN 9780819480392. doi: 10.1117/3.374903.

[26] H Leutz. Scintillating fibres. *Nucl. Instrum. Methods Phys. Res., A*, 364: 422–448, 1995. doi: 10.1016/0168-9002(95)00383-5.

[27] LHCb Collaboration. LHCb Tracker Upgrade Technical Design Report. Technical report, 2014.

[28] Thomas Ackernley, Alexander Bitadze, Themis Bowcock, Irene Cortinovis, Vadym Denysenko, et al. Mighty Tracker: Design studies for the downstream silicon tracker in Upgrade Ib and II. Technical report, CERN, Geneva, 2019.

[29] Arthur Hennequin, Benjamin Couturier, Vladimir Gligorov, Sebastien Ponce, Renato Quagliani, and Lionel Lacassagne. A fast and efficient SIMD track reconstruction algorithm for the LHCb Upgrade 1 VELO-PIX detector. *JINST*, 15(06):P06018, 2020. doi: 10.1088/1748-0221/15/06/P06018.

[30] Daniel Hugo Cámpora Pérez, Niko Neufeld, and Agustin Riscos Nuñez. A fast local algorithm for track reconstruction on parallel architectures. In *2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 698–707, 2019. doi: 10.1109/IPDPSW.2019.00118.

[31] M. Hansroul, H. Jeremie, and D. Savard. Fast circle fit with the conformal mapping method. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 270(2):498–501, 1988. ISSN 0168-9002. doi: https://doi.org/10.1016/0168-9002(88)90722-X.

[32] P. V. C. Hough. Machine Analysis of Bubble Chamber Pictures. *Conf. Proc. C*, 590914:554–558, 1959.

[33] T. Alexopoulos, M. Bachtis, E. Gazis, and G. Tsipolitis. Implementation of the legendre transform for track segment reconstruction in drift tube chambers. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 592(3):456–462, 2008. ISSN 0168-9002. doi: https://doi.org/10.1016/j.nima.2008.04.038.

[34] A Abba, F Bedeschi, M Citterio, F Caponio, A Cusimano, et al. A specialized track processor for the LHCb upgrade. Technical report, CERN, Geneva, 2014.

[35] A. Abba, F. Bedeschi, F. Caponio, R. Cenci, M. Citterio, et al. An "artificial retina" processor for track reconstruction at the full lhc crossing rate. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 824:260–262, 2016. ISSN 0168-9002. doi: https://doi.org/10.1016/j.nima.2015.10.048. Frontier Detectors for Frontier Physics: Proceedings of the 13th Pisa Meeting on Advanced Detectors.

[36] Phillip John Marshall. *Developing a Hybrid Machine Learning Model for VELO Upgrade Track Reconstruction*. PhD thesis, University of Liverpool, 2022. Presented 07 Jul 2022.

[37] R. Frühwirth. Application of kalman filtering to track and vertex fitting. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 262(2):444–450, 1987. ISSN 0168-9002. doi: https://doi.org/10.1016/0168-9002(87)90887-4.

[38] J. J. Thomson. Xl. cathode rays. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 44(269):293–316, 1897. doi: 10.1080/14786449708621070.

[39] Lyndon Evans and Philip Bryant. Lhc machine. *Journal of Instrumentation*, 3(08):S08001, aug 2008. doi: 10.1088/1748-0221/3/08/S08001.

[40] H. Wiedemann. *Particle Accelerator Physics, Fourth Edition*. Springer, 2015. ISBN 978-3-319-18316-0.

[41] Dudley Creagh. Chapter 1 synchrotron radiation and its use in art, archaeometry, and cultural heritage studies. In Dudley Creagh and David Bradley, editors, *Physical Techniques in the study of Art, Archaeology and Cultural Heritage*, volume 2 of *Physical Techniques in the Study of Art, Archaeology and Cultural Heritage*, pages 1–95. Elsevier, 2007. doi: https://doi.org/10.1016/S1871-1731(07)80003-0.

[42] Shayna E. Rich and Kavita V. Dharmarajan. *Introduction to Radiation Therapy*, pages 319–328. Springer International Publishing, Cham, 2019. ISBN 978-3-319-99684-4. doi: 10.1007/978-3-319-99684-4_36.

[43] M. S. Livingston and J. P. Blewett. *Particle Accelerators: Principles and Applications*. McGraw-Hill, 1962. ISBN 9780070381407.

[44] Maurizio Vretenar, J Vollaire, R Scrivens, C Rossi, F Roncarolo, et al. *Linac4 design report*, volume 6 of *CERN Yellow Reports: Monographs*. CERN, Geneva, 2020. doi: 10.23731/CYRM-2020-006.

[45] E. J. N. Wilson. *An introduction to particle accelerators*. 2001. ISBN 978-0-19-852054-2.

[46] Bruno Benedetto Rossi. *High-energy particles*. Prentice-Hall physics series. Prentice-Hall, New York, NY, 1952.

[47] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. *Dive into Deep Learning*. Cambridge University Press, 2023. https://D2L.ai.

[48] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition (Springer Series in Statistics)*. 02 2009. ISBN 0387848576.

[49] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[50] Sajid Ali, Tamer Abuhmed, Shaker El-Sappagh, Khan Muhammad, Jose M. Alonso-Moral, et al. Explainable artificial intelligence (xai): What we know and what is left to attain trustworthy artificial intelligence. *Information Fusion*, 99:101805, 2023. ISSN 1566-2535. doi: https://doi.org/10.1016/j.inffus.2023.101805.

[51] Pudi Dhilleswararao, Srinivas Boppu, M. Sabarimalai Manikandan, and Linga Reddy Cenkeramaddi. Efficient hardware architectures for accelerating deep neural networks: Survey. *IEEE Access*, 10:131788–131828, 2022. doi: 10.1109/ACCESS.2022.3229767.

[52] Shiv Ram Dubey, Satish Kumar Singh, and Bidyut Baran Chaudhuri. Activation functions in deep learning: A comprehensive survey and benchmark. *Neurocomputing*, 503:92–108, 2022. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2022.06.111.

[53] Andrinandrasana Rasamoelina, Fouzia Adjailia, and Peter Sincak. A review of activation function for artificial neural network. pages 281–286, 01 2020. doi: 10.1109/SAMI48414.2020.9108717.

[54] Juan Terven, Diana-Margarita Cordova-Esparza, Julio-Alejandro Romero-González, Alfonso Ramírez-Pedraza, and E. A. Chávez-Urbiola. A comprehensive survey of loss functions and metrics in deep learning. *Artificial Intelligence Review*, 58(7), April 2025. ISSN 1573-7462. doi: 10.1007/s10462-025-11198-7.

[55] Stanford university computer science department, cs231n: Deep learning for computer vision, 2018 slides *Backpropagation and Gradients*, 2018. *Stanford University*, https://cs231n.stanford.edu/slides/2018/cs231n_2018_ds02.pdf, Accessed 12/09/2025.

[56] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations (ICLR) 2015*, 2014.

[57] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997. doi: 10.1162/neco.1997.9.8.1735.

[58] Zewen Li, Wenjie Yang, Shouheng Peng, and Fan Liu. A survey of convolutional neural networks: Analysis, applications, and prospects. *CoRR*, abs/2004.02806, 2020.

[59] Ashish Kumar Shakya, Gopinatha Pillai, and Sohom Chakrabarty. Reinforcement learning algorithms: A brief survey. *Expert Systems with Applications*, 231:120495, 2023. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2023.120495.

[60] NobelPrize.org. Nobel prize outreach ab 2024. fri. 8 nov 2024., 2024. *New York University*, https://www.nobelprize.org/prizes/physics/2024/press-release/, Accessed 08/11/2024.

[61] Marian Stahl. Machine learning and parallelism in the reconstruction of LHCb and its upgrade. *J. Phys.: Conf. Ser.*, 898:042042, 2017. doi: 10.1088/1742-6596/898/4/042042. Submitted to Journal of Physics: Conference Series as proceedings for the 22nd International Conference on Computing in High Energy and Nuclear Physics, CHEP 2016, 10-14 October 2016, San Francisco.

[62] Alexander Radovic, Mike Williams, David Rousseau, Michael Kagan, Daniele Bonacorsi, et al. Machine learning at the energy and intensity frontiers of particle physics. *Nature*, 560(7716):41–48, 2018. doi: 10.1038/s41586-018-0361-2.

[63] Anton Poluektov. Faster, more efficient, more robust: Machine learning in LHCb's real-time processing. 2025.

[64] V V Gligorov and M Williams. Efficient, reliable and fast high-level triggering using a bonsai boosted decision tree. *Journal of Instrumentation*, 8(02):P02013, feb 2013. doi: 10.1088/1748-0221/8/02/P02013.

[65] A. Aurisano, A. Radovic, D. Rocco, A. Himmel, M.D. Messier, et al. A convolutional neural network neutrino event classifier. *Journal of Instrumentation*, 11(09):P09001–P09001, September 2016. ISSN 1748-0221. doi: 10.1088/1748-0221/11/09/p09001.

[66] P. Adamson, L. Aliaga, D. Ambrose, N. Anfimov, A. Antoshkin, et al. Constraints on oscillation parameters from ν_e appearance and ν_μ disappearance in nova. *Phys. Rev. Lett.*, 118:231801, Jun 2017. doi: 10.1103/PhysRevLett.118.231801.

[67] B. Abi, R. Acciarri, M. A. Acero, G. Adamov, D. Adams, et al. Neutrino interaction classification with a convolutional neural network in the dune far detector. *Physical Review D*, 102(9), nov 2020. ISSN 2470-0029. doi: 10.1103/physrevd.102.092003.

[68] Abtin Narimani Charan. Particle identification with the belle ii calorimeter using machine learning. *Journal of Physics: Conference Series*, 2438(1):012111, February 2023. ISSN 1742-6596. doi: 10.1088/1742-6596/2438/1/012111.

[69] Fedor Ratnikov. Deep Learning Approaches for LHCb ECAL Reconstruction. 2023.

[70] Gilles Jean Grasseau. A deep neural network method for analyzing the CMS HGCal events. Technical report, CERN, Geneva, 2020.

[71] Johann Christoph Voigt. Machine Learning for Real-Time Processing of ATLAS Liquid Argon Calorimeter Signals with FPGAs. Technical report, CERN, Geneva, 2024.

[72] Giuseppe Costantino. Machine learning for real-time muon identification of the lhcb experiment at cern, May 2025.

[73] Identification of Jets Containing b-Hadrons with Recurrent Neural Networks at the ATLAS Experiment. Technical report, CERN, Geneva, 2017.

[74] Daniel Guest, Julian Collado, Pierre Baldi, Shih-Chieh Hsu, Gregor Urban, and Daniel Whiteson. Jet flavor classification in high-energy physics with deep neural networks. *Phys. Rev. D*, 94:112002, Dec 2016. doi: 10.1103/PhysRevD.94.112002.

[75] Graph Neural Network Jet Flavour Tagging with the ATLAS Detector. Technical report, CERN, Geneva, 2022.

[76] Huilin Qu, Congqiao Li, and Sitian Qian. Particle transformer for jet tagging, 2024.

[77] Javier Mauricio Duarte. Novel ML technique applications. Technical report, CERN, Geneva, 2024.

[78] The CMS Collaboration. Observation of a new boson at a mass of 125 gev with the cms experiment at the lhc. *Physics Letters B*, 716(1):30–61, 2012. ISSN 0370-2693. doi: https://doi.org/10.1016/j.physletb.2012.08.021.

[79] The ATLAS Collaboration. Observation of a new particle in the search for the standard model higgs boson with the atlas detector at the lhc. *Physics Letters B*, 716(1):1–29, September 2012. ISSN 0370-2693. doi: 10.1016/j.physletb.2012.08.020.

[80] Vardan Khachatryan, A.M. Sirunyan, A. Tumasyan, W. Adam, Thomas Bergauer, et al. Observation of the rare $B_s^0 \to \mu^+\mu^-$ decay from the combined analysis of cms and lhcb data. *Nature*, 522, 05 2015. doi: 10.1038/nature14474.

[81] R. Aaij, B. Adeva, M. Adinolfi, Z. Ajaltouni, S. Akar, et al. Measurement of the $B_s^0 \to \mu^+\mu^-$ branching fraction and effective lifetime and search for $B^0 \to \mu^+\mu^-$ decays. *Phys. Rev. Lett.*, 118:191801, May 2017. doi: 10.1103/PhysRevLett.118.191801.

[82] R. Aaij, B. Adeva, M. Adinolfi, A. Affolder, Z. Ajaltouni, et al. Observation of $j/\psi p$ resonances consistent with pentaquark states in $\Lambda_b^0 \to j/\psi K^- p$ decays. *Phys. Rev. Lett.*, 115:072001, Aug 2015. doi: 10.1103/PhysRevLett.115.072001.

[83] Steven Schramm. Machine learning at CERN: ATLAS, LHCb, and more. *PoS*, ICHEP2018:158, 2019. doi: 10.22323/1.340.0158.

[84] Lucio Anderlini. Machine learning for the lhcb simulation, 2022.

[85] Matteo Barbetti. The flash-simulation paradigm and its implementation based on deep generative models for the lhcb experiment at cern, July 2024.

[86] G. Aad, B. Abbott, D. C. Abbott, A. Abed Abud, K. Abeling, et al. Deep generative models for fast photon shower simulation in atlas. *Computing and Software for Big Science*, 8(1), March 2024. ISSN 2510-2044. doi: 10.1007/s41781-023-00106-9.

[87] Reweighting of simulated events using machine learning techniques in CMS. Technical report, CERN, Geneva, 2024.

[88] David Richard Shope. Software Upgrades for the High-Luminosity LHC. *PoS*, LHCP2024:193, 2025. doi: 10.22323/1.478.0193.

[89] Hyuna Sung, Jacques Ferlay, Rebecca L. Siegel, Mathieu Laversanne, Isabelle Soerjomataram, Ahmedin Jemal, and Freddie Bray. Global cancer statistics 2020: Globocan estimates of incidence and mortality worldwide for 36 cancers

in 185 countries. *CA: A Cancer Journal for Clinicians*, 71(3):209–249, 2021. doi: https://doi.org/10.3322/caac.21660.

[90] Freddie Bray, Mathieu Laversanne, Elisabete Weiderpass, and Isabelle Soerjomataram. The ever-increasing importance of cancer as a leading cause of premature death worldwide. *Cancer*, 127(16):3029–3030, 2021. doi: https://doi.org/10.1002/cncr.33587.

[91] Juliette Thariat, Jean-Michel Hannoun-Levi, Arthur Sun Myint, Te Vuong, and Jean-Pierre Gérard. Past, present, and future of radiotherapy for the benefit of patients juliette thariat, jean-michel hannoun-levi, arthur sun myint, te vuong and jean-pierre gérard. *Nature Reviews Clinical Oncology*, 10, 11 2012. doi: 10.1038/nrclinonc.2012.203.

[92] Robert L Foote, Scott L Stafford, Ivy A Petersen, Jose S Pulido, Michelle J Clarke, et al. The clinical case for proton beam therapy. *Radiation Oncology*, 7(1):1–10, 2012.

[93] Cancer and radiation therapy: current advances and future directions. *International journal of medical sciences*, 9(3):193–199, 2012. ISSN 1449-1907.

[94] NDRS. National disease registration service cancer data: Cancer treatments 2013 - 2021., 2024. https://digital.nhs.uk/ndrs/data/data-outputs/cancer-data-hub/cancer-treatments, https://nhsd-ndrs.shinyapps.io/treatment_by_demographic_factors/, https://github.com/NHSE-NDRS/Cancer-Treatments, Accessed 22/07/2024.

[95] Deborah Citrin. Recent developments in radiotherapy. *New England Journal of Medicine*, 377:1065–1075, 09 2017. doi: 10.1056/NEJMra1608986.

[96] Radhe Mohan and David Grosshans. Proton therapy – present and future. *Advanced Drug Delivery Reviews*, 109:26–44, 2017. ISSN 0169-409X. doi: https://doi.org/10.1016/j.addr.2016.11.006. Radiotherapy for cancer: present and future.

[97] Laura Beaton, Steve Bandula, Mark N Gaze, and Ricky A Sharma. How rapid advances in imaging are defining the future of precision radiation oncology. *British journal of cancer*, 120(8):779–790, 2019.

[98] Man Hu, Liyang Jiang, Xiangli Cui, Jianguang Zhang, and Jinming Yu. Proton beam therapy for cancer in the era of precision medicine. *Journal of hematology & oncology*, 11(1):136, 2018.

[99] Stereotactic radiotherapy (srt), 2023. *Cancer Research UK*, https://www.cancerresearchuk.org/about-cancer/treatment/radiotherapy/external/types/stereotactic-body-radiotherapy-sbrt, Accessed 1/08/2024.

[100] Michele M. Kim, Arash Darafsheh, Jan Schuemann, Ivana Dokic, Olle Lundh, et al. Development of ultra-high dose-rate (flash) particle therapy. *IEEE Transactions on Radiation and Plasma Medical Sciences*, 6(3):252–262, 2022. doi: 10.1109/TRPMS.2021.3091406.

[101] Robert R. Wilson. Radiological use of fast protons. *Radiology*, 47(5):487–491, 1946. doi: 10.1148/47.5.487. PMID: 20274616.

[102] Cornelius A. Tobias, John H. Lawrence, James L. Born, Rollin K. McCombs, J. E. Roberts, et al. Pituitary irradiation with high-energy proton beams: a preliminary report. *Cancer research*, 18 2:121–34, 1958.

[103] Xiufang Tian, Kun Liu, Yong Hou, Jian Cheng, and Jiandong Zhang. The evolution of proton beam therapy: Current and future status. *Molecular and clinical oncology*, 8(1):15–21, 2018.

[104] Stereotactic radiosurgery, 2019. *Mayo Clinic*, https://www.mayoclinic.org/tests-procedures/sbrt/pyc-20446794, Accessed 5/08/2024.

[105] Shelby Lane, Jason Slater, and Gary Yang. Image-guided proton therapy: A comprehensive review. *Cancers*, 15:2555, 04 2023. doi: 10.3390/cancers15092555.

[106] Robert P Johnson. Review of medical radiography and tomography with proton beams. *Reports on Progress in Physics*, 81(1):016701, 2017.

[107] Jessika Contreras, Tianyu Zhao, Stephanie Perkins, Baozhou Sun, Sreekrishna Goddu, et al. The world's first single-room proton therapy facility: Two-year experience. *Practical Radiation Oncology*, 7(1):e71–e76, 2017. ISSN 1879-8500. doi: https://doi.org/10.1016/j.prro.2016.07.003.

[108] Michael F. Moyers, Milind Sardesai, Sean Sun, and Daniel W. Miller. Ion stopping powers and ct numbers. *Medical Dosimetry*, 35(3):179–194, 2010. ISSN 0958-3947. doi: https://doi.org/10.1016/j.meddos.2009.05.004.

[109] Walter Huda. *Review of radiologic physics*. Wolters Kluwer Health, Lippincott Williams & Wilkins, Philadelphia, 3rd ed. edition, 2010. ISBN 9780781785693.

[110] X-ray mass attenuation coefficients. *U.S. Physical Measurement Laboratory*, https://physics.nist.gov/PhysRefData/XrayMassCoef/tab4.html, Accessed 20/03/2025.

[111] Nora Hünemohr, Steffen Greilich, Harald Paganetti, Joao Seco, and Oliver Jäkel. Tissue decomposition from dual energy ct data for mc based dose calculation in particle therapy. *Medical Physics*, 41(6), 6 2014. ISSN 0094-2405. doi: 10.1118/1.4875976.

[112] A. M. Cormack. Representation of a Function by Its Line Integrals, with Some Radiological Applications. *Journal of Applied Physics*, 34(9):2722–2727, 09 1963. ISSN 0021-8979. doi: 10.1063/1.1729798.

[113] PJ Doolan, M Testa, G Sharp, EH Bentefour, G Royle, and HM Lu. Patient-specific stopping power calibration for proton therapy planning based on single-detector proton radiography. *Physics in Medicine & Biology*, 60(5):1901, 2015.

[114] Elena Fogazzi, Diego Trevisan, Paolo Farace, Roberto Righetto, Simon Rit, et al. Characterization of the infn proton ct scanner for cross-calibration of x-ray ct. *Physics in Medicine & Biology*, 68(12):124001, jun 2023. doi: 10.1088/1361-6560/acd6d3.

[115] Matthias Fippel and Martin Soukup. A monte carlo dose calculation algorithm for proton therapy. *Medical Physics*, 31, 2004.

[116] Particle Data Group, R L Workman, V D Burkert, V Crede, E Klempt, U Thoma, et al. Review of Particle Physics, 34. Passage of Particles Through Matter. *Progress of Theoretical and Experimental Physics*, 2022(8):083C01, 08 2022. ISSN 2050-3911. doi: 10.1093/ptep/ptac097.

[117] Alan C Bovik. *The essential guide to image processing*. Academic Press, 2009.

[118] Tianfang li, Zhengrong Liang, Jayalakshmi Singanallur, Todd Satogata, David Williams, and Reinhard Schulte. Reconstruction for proton computed tomography by tracing proton trajectories: A monte carlo study. *Medical physics*, 33:699–706, 04 2006. doi: 10.1118/1.2171507.

[119] Feriel Khellaf, Nils Krah, Jean Michel Létang, Charles-Antoine Collins-Fekete, and Simon Rit. A comparison of direct reconstruction algorithms in proton computed tomography. *Physics in Medicine & Biology*, 65(10):105010, 2020.

[120] RW Schulte, SN Penfold, JT Tafas, and KE Schubert. A maximum likelihood proton path formalism for application in proton computed tomography. *Medical physics*, 35(11):4849–4856, 2008.

[121] DC Williams. The most likely path of an energetic charged particle through a uniform medium. *Physics in Medicine & Biology*, 49(13):2899, 2004.

[122] Charles-Antoine Collins Collins-Fekete, Paul Doolan, Marta F Dias, Luc Beaulieu, and Joao Seco. Developing a phenomenological model of the proton trajectory within a heterogeneous medium required for proton imaging. *Physics in Medicine & Biology*, 60(13):5071, 2015.

[123] Uwe Schneider and Eros Pedroni. Multiple coulomb scattering and spatial resolution in proton radiography. *Medical physics*, 21(11):1657–1663, 1994.

[124] Charles-Antoine Collins-Fekete, Lennart Volz, Stephen KN Portillo, Luc Beaulieu, and Joao Seco. A theoretical framework to predict the most likely ion path in particle imaging. *Physics in Medicine & Biology*, 62(5):1777, 2017.

[125] Charles-Antoine Collins-Fekete, Esther Bär, Lennart Volz, Hugo Bouchard, Luc Beaulieu, and Joao Seco. Extension of the fermi–eyges most-likely path in heterogeneous medium with prior knowledge information. *Physics in Medicine & Biology*, 62(24):9207, 2017.

[126] Nils Krah, Jean-Michel Létang, and Simon Rit. Polynomial modelling of proton trajectories in homogeneous media for fast most likely path estimation and trajectory simulation. *Physics in Medicine & Biology*, 64(19):195014, 2019.

[127] Mark D Brooke and Scott N Penfold. An inhomogeneous most likely path formalism for proton computed tomography. *Physica Medica*, 70:184–195, feb 2020.

[128] Jarle Rambo Sølie, Lennart Volz, Helge Egil Seime Pettersen, Pierluigi Piersimoni, Odd Harald Odland, et al. Image quality of list-mode proton imaging without front trackers. *Physics in Medicine & Biology*, 65(13):135012, jul 2020. doi: 10.1088/1361-6560/ab8ddb.

[129] SA McAllister, KE Schubert, R Schulte, and S Penfold. General purpose graphics processing unit speedup of integral relative electron density calculation for proton computed tomography. In *2009 IEEE Nuclear Science Symposium Conference Record (NSS/MIC)*, pages 4085–4087. IEEE, 2009.

[130] Scott Alan McAllister. *Efficient proton computed tomography image reconstruction using general purpose graphics processing units*. PhD thesis, California State University, San Bernardino, 2009.

[131] Dongxu Wang, T Rockwell Mackie, and Wolfgang A Tomé. Bragg peak prediction from quantitative proton computed tomography using different path estimates. *Physics in Medicine & Biology*, 56(3):587, jan 2011. doi: 10.1088/0031-9155/56/3/005.

[132] Dimitrios Lazos, Charles-Antoine Collins-Fekete, Philip Evans, and Nikolaos Dikaios. Molière maximum likelihood proton path estimation approximated by cubic bézier curve for scatter corrected proton ct reconstruction. *Physics in Medicine & Biology*, 65(17):175003, aug 2020. doi: 10.1088/1361-6560/ab9413.

[133] Dimitrios Lazos, Charles-Antoine Collins-Fekete, Miroslaw Bober, Philip Evans, and Nikolaos Dikaios. Machine learning for proton path tracking in proton computed tomography. *Physics in Medicine & Biology*, 66(10):105013, may 2021. doi: 10.1088/1361-6560/abf1fd.

[134] Alexander Schilling, Max Aehle, Johan Alme, Gergely Gábor Barnaföldi, Tea Bodova, et al. Uncertainty-aware spot rejection rate as quality metric for proton therapy using a digital tracking calorimeter. *Physics in Medicine & Biology*, 68(19):194001, sep 2023. doi: 10.1088/1361-6560/acf5c2.

[135] Helge Egil Seime Pettersen, Max Aehle, Johan Alme, Gergely Gábor Barnaföldi, Vyacheslav Borshchov, et al. Investigating particle track topology for range telescopes in particle radiography using convolutional neural networks. *Acta Oncologica*, 60(11):1413–1418, 2021. doi: 10.1080/0284186X.2021.1949037. PMID: 34259117.

[136] Yusuke Nomura, Sodai Tanaka, Jeff Wang, Hiroki Shirato, Shinichi Shimizu, and Lei Xing. Calibrated uncertainty estimation for interpretable proton computed tomography image correction using bayesian deep learning. *Physics in Medicine & Biology*, 66(6):065029, mar 2021. doi: 10.1088/1361-6560/abe956.

[137] Cong Sheng, Yu Ding, Yaping Qi, Man Hu, Jianguang Zhang, Xiangli Cui, Yingying Zhang, and Wanli Huo. A denoising method based on deep learning for proton radiograph using energy resolved dose function. *Physics in Medicine & Biology*, 69(2):025015, jan 2024. doi: 10.1088/1361-6560/ad15c4.

[138] Nils Krah, Vincenzo Patera, Simon Rit, A Schiavi, and Ilaria Rinaldi. Regularised patient-specific stopping power calibration for proton therapy planning based on proton radiographic images. *Physics in Medicine and Biology*, 64, 03 2019. doi: 10.1088/1361-6560/ab03db.

[139] Johan Alme, Gergely Gábor Barnaföldi, Rene Barthel, Vyacheslav Borshchov, Tea Bodova, et al. A high-granularity digital tracking calorimeter optimized for proton ct. *Frontiers in Physics*, 8, 2020. ISSN 2296-424X. doi: 10.3389/fphy.2020.568243.

[140] Robert Johnson, Vladimir Bashkirov, Langley DeWitt, Valentina Giacometti, Ford Hurley, et al. A fast experimental scanner for proton ct: Technical performance and first experience with phantom scans. *IEEE Transactions on Nuclear Science*, 63:1–1, 12 2015. doi: 10.1109/TNS.2015.2491918.

[141] Michela Esposito, Chris Waltham, Jonathan Taylor, Sam Manger, Ben Phoenix, et al. Pravda: The first solid-state system for proton computed tomography. *Physica Medica*, 55, 11 2018. doi: 10.1016/j.ejmp.2018.10.020.

[142] Ethan DeJongh, Don DeJongh, Igor Polnyi, Victor Rykalin, Christina Sarosiek, et al. Technical note: A fast and monolithic prototype clinical proton radiography system optimized for pencil beam scanning. *Medical Physics*, 48, 02 2021. doi: 10.1002/mp.14700.

[143] F. Ulrich-Pur, T. Bergauer, A. Hirtl, C. Irmler, S. Kaser, F. Pitters, and S. Rit. Novel ion imaging concept based on time-of-flight measurements with low gain avalanche detectors. *Journal of Instrumentation*, 18(02):C02062, feb 2023. doi: 10.1088/1748-0221/18/02/C02062.

[144] Sébastien Jan, D Benoit, E Becheva, T Carlier, F Cassol, et al. Gate v6: a major enhancement of the gate simulation platform enabling modelling of ct and radiotherapy. *Physics in Medicine & Biology*, 56(4):881, 2011.

[145] Sea Agostinelli, John Allison, K al Amako, John Apostolakis, H Araujo, et al. Geant4—a simulation toolkit. *Nuclear instruments and methods in physics re-*

search section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 506(3):250–303, 2003.

[146] The Geant4 Collaboration. Em opt0, . *Geant 4 v10.5 User Documentation*, https://geant4-userdoc.web.cern.ch/UsersGuides/PhysicsListGuide/BackupVersions/V10.5-2.0/html/electromagnetic/Opt0.html, Accessed 24/03/2025.

[147] The Geant4 Collaboration. Qgsp_bic, . *Geant 4 v10.5 User Documentation*, https://geant4-userdoc.web.cern.ch/UsersGuides/PhysicsListGuide/BackupVersions/V10.5-2.0/html/reference_PL/QGSP_BIC.html, Accessed 24/03/2025.

[148] M. J. Berger, M. Inokuti, H. H. Andersen, H. Bichsel, D. Powers, S . M. Seltzer, D . Thwaites, and D. E. Watt. Report 49. *Journal of the International Commission on Radiation Units and Measurements*, os25(2):NP–NP, 04 2016. ISSN 1473-6691. doi: 10.1093/jicru/os25.2.Report49.

[149] Kent Wong, Bela Erdelyi, Reinhard Schulte, Vladimir Bashkirov, George Coutrakon, Hartmut Sadrozinski, Scott Penfold, and Anatoly Rosenfeld. The effect of tissue inhomogeneities on the accuracy of proton path reconstruction for proton computed tomography. *AIP Conference Proceedings*, 1099, 03 2009. doi: 10.1063/1.3120078.

[150] Zhiyuan Liu and Jie Zhou. *Introduction to graph neural networks / Zhiyuan Liu and Jie Zhou.* Synthesis lectures on artificial intelligence and machine learning, #45. Morgan & Claypool, San Rafael, California, 2020. ISBN 1-68173-766-3.

[151] J. A. (John Adrian) Bondy and U. S. R. Murty. *Graph theory J.A. Bondy, U.S.R. Murty.* Graduate texts in mathematics, 244. Springer, New York, 2008. ISBN 9781846289705.

[152] Martin Grandjean. A social network analysis of twitter: Mapping the digital humanities community. *Cogent Arts & Humanities*, 3:1171458, 04 2016. doi: 10.1080/23311983.2016.1171458.

[153] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Rezende, and Koray Kavukcuoglu. Interaction networks for learning about objects, relations and physics. 12 2016. doi: 10.48550/arXiv.1612.00222.

[154] Michael B. Chang, Tomer Ullman, Antonio Torralba, and Joshua B. Tenenbaum. A compositional object-based approach to learning physical dynamics, 2017.

[155] Richard J. Trudeau. *Introduction to Graph Theory*. Dover Books on Mathematics. Dover Publications, Newburyport, 2013. ISBN 0-486-31866-4.

[156] Norman. Biggs, E. Keith Lloyd, and Robin J. Wilson. *Graph theory 1736-1936*. Clarendon Press, Oxford, 1976. ISBN 0198539010.

[157] Reinhard Diestel. *Graph theory by Reinhard Diestel*. Graduate Texts in Mathematics, 173. Springer Berlin Heidelberg, Berlin, Heidelberg, fifth edition. edition, 2017. ISBN 9783662536223.

[158] Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, et al. Relational inductive biases, deep learning, and graph networks, 2018.

[159] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, January 2021. ISSN 2162-2388. doi: 10.1109/tnnls.2020.2978386.

[160] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. *CoRR*, abs/1704.01212, 2017.

[161] Petar Veličković. Message passing all the way up, 2022.

[162] Gage DeZoort, Peter Battaglia, Catherine Biscarat, and Jean-Roch Vlimant. Graph neural networks at the large hadron collider. *Nature Reviews Physics*, 5, 04 2023. doi: 10.1038/s42254-023-00569-0.

[163] A. Sperduti and A. Starita. Supervised neural networks for the classification of structures. *IEEE Transactions on Neural Networks*, 8(3):714–735, 1997. doi: 10.1109/72.572108.

[164] M. Gori, G. Monfardini, and F. Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734 vol. 2, 2005. doi: 10.1109/IJCNN.2005.1555942.

[165] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009. doi: 10.1109/TNN.2008.2005605.

[166] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020. ISSN 2666-6510. doi: https://doi.org/10.1016/j.aiopen.2021.01.001.

[167] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks, 2017.

[168] Xiaodan Liang, Xiaohui Shen, Jiashi Feng, Liang Lin, and Shuicheng Yan. Semantic object parsing with graph lstm, 2016.

[169] Nanyun Peng, Hoifung Poon, Chris Quirk, Kristina Toutanova, and Wen tau Yih. Cross-sentence n-ary relation extraction with graph lstms, 2017.

[170] Kai Sheng Tai, Richard Socher, and Christopher D. Manning. Improved semantic representations from tree-structured long short-term memory networks, 2015.

[171] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs, 2014.

[172] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2017.

[173] David Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gómez-Bombarelli, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P. Adams. Convolutional networks on graphs for learning molecular fingerprints, 2015.

[174] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs, 2018.

[175] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2018.

[176] Jiani Zhang, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and Dit-Yan Yeung. Gaan: Gated attention networks for learning on large and spatiotemporal graphs, 2018.

[177] HEP.TrkX Collaboration. Hep advanced tracking algorithms with cross-cutting applications (project hep.trkx), . https://heptrkx.github.io/, Accessed 28/11/2024.

[178] Steven Farrell, Paolo Calafiura, Mayur Mudigonda, Prabhat, Dustin Anderson, et al. Novel deep learning methods for track reconstruction, 2018.

[179] Exa.TrkX Collaboration. Hep advanced tracking algorithms at the exascale (project exa.trkx), . https://exatrkx.github.io/, Accessed 28/11/2024.

[180] Nicholas Choma, Daniel Murnane, Xiangyang Ju, Paolo Calafiura, Sean Conlon, et al. Track seeding and labelling with embedded-space graph neural networks, 2020.

[181] Xiangyang Ju, Daniel Murnane, Paolo Calafiura, Nicholas Choma, Sean Conlon, et al. Performance of a geometric deep learning pipeline for hl-lhc particle tracking. *The European Physical Journal C*, 81, 10 2021. doi: 10.1140/epjc/s10052-021-09675-8.

[182] Catherine Biscarat, Sylvain Caillou, Charline Rougier, Jan Stark, and Jad Zahreddine. Towards a realistic track reconstruction algorithm based on graph neural networks for the hl-lhc. *EPJ Web of Conferences*, 251:03047, 2021. ISSN 2100-014X. doi: 10.1051/epjconf/202125103047.

[183] Joosep Pata, Javier Duarte, Jean-Roch Vlimant, Maurizio Pierini, and Maria Spiropulu. Mlpf: efficient machine-learned particle-flow reconstruction using graph neural networks. *The European Physical Journal C*, 81(5), May 2021. ISSN 1434-6052. doi: 10.1140/epjc/s10052-021-09158-w.

[184] Hewes, Jeremy, Aurisano, Adam, Cerati, Giuseppe, Kowalkowski, Jim, Lee, Claire, et al. Graph neural network for object reconstruction in liquid argon time projection chambers. *EPJ Web Conf.*, 251:03054, 2021. doi: 10.1051/epjconf/202125103054.

[185] Aneesh Heintz, Vesal Razavimaleki, Javier Duarte, Gage DeZoort, Isobel Ojalvo, et al. Accelerated charged particle tracking with graph neural networks on fpgas, 2020.

[186] Collaboration ATLAS. Technical Design Report for the Phase-II Upgrade of the ATLAS Trigger and Data Acquisition System - Event Filter Tracking Amendment. Technical report, CERN, Geneva, 2022.

[187] Xiangyang Ju, Steven Farrell, Paolo Calafiura, Daniel Murnane, Prabhat, et al. Graph neural networks for particle reconstruction in high energy physics detectors, 2020.

[188] Rashmi Shivni. The deconstructed standard model equation, 2016. *Symmetry Magazine*, https://www.symmetrymagazine.org/article/the-deconstructed-standard-model-equation.

[189] CERN. The standard model, . *CERN*, https://home.cern/science/physics/standard-model, Accessed 17/01/2025.

[190] Julia Woithe, Gerfried J Wiener, and Frederik F Van der Veken. Let's have a coffee with the standard model of particle physics! *Physics Education*, 52(3): 034001, mar 2017. doi: 10.1088/1361-6552/aa5b25.

[191] David J. (David Jeffery) Griffiths. *Introduction to elementary particles / David Griffiths.* Physics textbook. Wiley-VCH, Weinheim, 2nd rev. ed. edition, 2008. ISBN 9783527406012.

[192] S. F. Novaes. Standard model: An introduction, 2000.

[193] S. Navas, C. Amsler, T. Gutsche, C. Hanhart, J. J. Hernández-Rey, et al. Review of particle physics. *Phys. Rev. D*, 110(3):030001, 2024. doi: 10.1103/PhysRevD.110.030001.

[194] CERN. Facts and figures about the lhc, . *CERN*, https://home.cern/science/physics/higgs-boson, Accessed 17/01/2025.

[195] CERN. Dark matter, . *CERN*, https://home.cern/science/physics/dark-matter, Accessed 17/01/2025.

[196] Kyu-Hyun Chae. Distinguishing dark matter, modified gravity, and modified inertia with the inner and outer parts of galactic rotation curves. *The Astrophysical Journal*, 941(1):55, December 2022. ISSN 1538-4357. doi: 10.3847/1538-4357/ac93fc.

[197] CERN. 09 december 1949, origins, . *CERN*, https://timeline.web.cern.ch/origins, Accessed 17/02/2025.

[198] CERN. Member states, . *CERN*, https://home.cern/about/who-we-are/our-governance/member-states, Accessed 17/02/2025.

[199] O. Aberle, I Béjar Alonso, O Brüning, P Fessia, L Rossi, et al. *High-Luminosity Large Hadron Collider (HL-LHC): Technical design report*. CERN Yellow Reports: Monographs. CERN, Geneva, 2020. doi: 10.23731/CYRM-2020-0010.

[200] CERN. Our history, . *CERN*, https://home.cern/about/who-we-are/our-history, Accessed 17/02/2025.

[201] Ewa Lopienska. The CERN accelerator complex, layout in 2022. Complexe des accélérateurs du CERN en janvier 2022. 2022.

[202] Oliver Brüning, Max Klein, Stephen Myers, and Lucio Rossi. 70 years at the high-energy frontier with the CERN accelerator complex. *Nature Rev. Phys.*, 6(10):628–637, 2024. doi: 10.1038/s42254-024-00758-5.

[203] CERN. Cern's accelerator complex, . *CERN*, https://home.cern/science/accelerators/accelerator-complex, Accessed 17/02/2025.

[204] CERN. Experiments, . *CERN*, https://home.cern/science/experiments, Accessed 19/02/2025.

[205] CERN. Facts and figures about the lhc, . *CERN*, https://home.cern/resources/faqs/facts-and-figures-about-lhc, Accessed 17/01/2025.

[206] Alex Kohls, Jens Vigen, and Micha Moskovic. A decade in lhc publications. *Cern Courier*, https://cerncourier.com/a/a-decade-in-lhc-publications/, Accessed 09/06/2025.

[207] CERN. Longer term lhc schedule, . *LHC Commissioning, CERN*, https://lhc-commissioning.web.cern.ch/schedule/LHC-long-term.htm, Accessed 17/06/2025.

[208] CERN. The large hadron collider, . *CERN*, https://home.cern/science/accelerators/large-hadron-collider, Accessed 18/02/2025.

[209] Werner Herr and B Muratori. Concept of luminosity. 2006. doi: 10.5170/CERN-2006-002.361.

[210] Jorg Wenninger. Operation and Configuration of the LHC in Run 2. 2019.

[211] CERN. Cern experiments observe particle consistent with long-sought higgs boson, . *CERN*, https://home.web.cern.ch/news/press-release/cern/cern-experiments-observe-particle-consistent-long-sought-higgs-boson, Accessed 19/02/2025.

[212] S. Chatrchyan, V. Khachatryan, A.M. Sirunyan, A. Tumasyan, W. Adam, et al. Combined results of searches for the standard model higgs boson in pp collisions at s=7 tev. *Physics Letters B*, 710(1):26–48, 2012. ISSN 0370-2693. doi: https://doi.org/10.1016/j.physletb.2012.02.064.

[213] Georges Aad, Tatevik Abajyan, Brad Abbott, Jalal Abdallah, Samah Abdel Khalek, et al. Combined search for the standard model higgs boson in pp collisions at $\sqrt{s} = 7$ tev with the atlas detector. *Phys. Rev. D*, 86:032003, 2012. doi: 10.1103/PhysRevD.86.032003.

[214] Addendum to the report on the physics at the HL-LHC, and perspectives for the HE-LHC: Collection of notes from ATLAS and CMS. Technical report, CERN, Geneva, 2019.

[215] G Aad, S Bentvelsen, G J Bobbink, K Bos, H Boterenbrood, et al. The ATLAS Experiment at the CERN Large Hadron Collider. *JINST*, 3:S08003, 2008. doi: 10.1088/1748-0221/3/08/S08003.

[216] CERN. Atlas, . *CERN*, https://home.cern/science/experiments/atlas, Accessed 19/02/2025.

[217] S Chatrchyan, G Hmayakyan, V Khachatryan, A M Sirunyan, R Adolphi, et al. The CMS experiment at the CERN LHC. The Compact Muon Solenoid experiment. *JINST*, 3:S08004, 2008. doi: 10.1088/1748-0221/3/08/S08004.

[218] A Augusto Alves, L M Andrade, F Barbosa-Ademarlaudo, I Bediaga, G Cernicchiaro, et al. The LHCb Detector at the LHC. *JINST*, 3:S08005, 2008. doi: 10.1088/1748-0221/3/08/S08005.

[219] K Aamodt, A Abrahantes Quintana, R Achenbach, S Acounis, D Adamova, et al. The ALICE experiment at the CERN LHC. A Large Ion Collider Experiment. *JINST*, 3:S08002, 2008. doi: 10.1088/1748-0221/3/08/S08002.

[220] CERN. Heavy ions and quark-gluon plasma, . *CERN*, https://home.cern/science/physics/heavy-ions-and-quark-gluon-plasma, Accessed 19/02/2025.

[221] CERN. Totem, . *CERN*, https://home.cern/science/experiments/totem, Accessed 19/02/2025.

[222] The TOTEM Collaboration, G Anelli, G Antchev, P Aspell, V Avati, et al. *Journal of Instrumentation*, 3(08):S08007, aug 2008. doi: 10.1088/1748-0221/3/08/S08007.

[223] O Adriani, L Bonechi, M Bongi, G Castellini, R D'Alessandro, et al. The LHCf detector at the CERN Large Hadron Collider. *JINST*, 3:S08006, 2008. doi: 10.1088/1748-0221/3/08/S08006.

[224] Pinfold J. et al. Moedal-mapp, an lhc dedicated detector search facility, 2023.

[225] Henso Abreu, Elham Amin Mansour, Claire Antel, Akitaka Ariga, Tomoko Ariga, et al. The faser detector. *Journal of Instrumentation*, 19(05):P05066, may 2024. doi: 10.1088/1748-0221/19/05/P05066.

[226] G. Acampora, C. Ahdida, R. Albanese, C. Albrecht, A. Alexandrov, et al. Snd@lhc: the scattering and neutrino detector at the lhc. *Journal of Instrumentation*, 19(05):P05067, may 2024. doi: 10.1088/1748-0221/19/05/P05067.

[227] Framework TDR for the LHCb Upgrade II: Opportunities in flavour physics, and beyond, in the HL-LHC era. Technical report, CERN, Geneva, 2021.

[228] LHCb SMOG Upgrade. Technical report, CERN, Geneva, 2019.

[229] LHCb Collaboration. LHCb Tracker Upgrade Technical Design Report. Technical report, 2014.

[230] Sara Cesare. The LHCb Upstream Tracker: Operations and Performance in Run3. 2024.

[231] Plamen Hopchev. SciFi: A large Scintillating Fibre Tracker for LHCb. Technical report, 2017. Presented at The Fifth Annual Conference on Large Hadron Collider Physics.

[232] LHCb Collaboration. LHCb PID Upgrade Technical Design Report. Technical report, 2013.

[233] S Okamura. Commissioning of the upgraded RICH system at the LHCb experiment. Commissioning of the upgraded RICH system at the LHCb experiment. *JINST*, 17(11):C11006, 2022. doi: 10.1088/1748-0221/17/11/C11006.

[234] Carlos Abellán Beteta, A. Alfonso Albero, Y. Amhis, S. Barsuk, C. Beigbeder-Beau, et al. Calibration and performance of the LHCb calorimeters in Run 1 and 2 at the LHC. Technical report, 2020.

[235] Ken Wyllie, Federico Alessio, Clara Gaspar, Richard Jacobsson, Renaud Le Gac, Niko Neufeld, and Rainer Schwemmer. Electronics Architecture of the LHCb Upgrade. Technical report, CERN, Geneva, 2013.

[236] R. Aaij, S. Benson, M. De Cian, A. Dziurda, C. Fitzpatrick, et al. A comprehensive real-time analysis model at the lhcb experiment. *Journal of Instrumentation*, 14(04):P04006–P04006, April 2019. ISSN 1748-0221. doi: 10.1088/1748-0221/14/04/p04006.

[237] LHCb Trigger and Online Upgrade Technical Design Report. Technical report, 2014.

[238] RTA and DPA dataflow diagrams for Run 1, Run 2, and the upgraded LHCb detector . 2020.

[239] Computing Model of the Upgrade LHCb experiment. Technical report, CERN, Geneva, 2018.

[240] G. Barrand, I. Belyaev, P. Binko, M. Cattaneo, R. Chytracek, et al. Gaudi — a software architecture and framework for building hep data processing applications. *Computer Physics Communications*, 140(1):45–55, 2001. ISSN 0010-4655. doi: https://doi.org/10.1016/S0010-4655(01)00254-5. CHEP2000.

[241] LHCb Collaboration. Moore application gitlab repository, . https://gitlab.cern.ch/lhcb/Moore., Accessed 15/08/2025.

[242] R. Aaij, J. Albrecht, M. Belous, P. Billoir, T. Boettcher, et al. Allen: A high-level trigger on gpus for lhcb. *Computing and Software for Big Science*, 4(1), April 2020. ISSN 2510-2044. doi: 10.1007/s41781-020-00039-7.

[243] Paul Andre Günther. LHCb's Forward Tracking algorithm for the Run 3 CPU-based online track reconstruction sequence. 2022.

[244] LHCb Upgrade GPU High Level Trigger Technical Design Report. Technical report, CERN, Geneva, 2020.

[245] M. Adinolfi, G. Aglieri Rinella, E. Albrecht, T. Bellunato, S. Benson, et al. Performance of the lhcb rich detector at the lhc. *The European Physical Journal C*, 73(5), May 2013. ISSN 1434-6052. doi: 10.1140/epjc/s10052-013-2431-9.

[246] L. Anderlini, F. Archilli, A. Cardini, V. Cogoni, M. Fontana, et al. Muon identification for lhcb run 3. *Journal of Instrumentation*, 15(12):T12005–T12005, December 2020. ISSN 1748-0221. doi: 10.1088/1748-0221/15/12/t12005.

[247] Computing Model of the Upgrade LHCb experiment. Technical report, CERN, Geneva, 2018.

[248] Johannes Albrecht, Frederic Henry Blanc, Matthew John Charles, Conor Fitzpatrick, Matthew David Needham, et al. Recommendations from Upgrade Ib Review. Technical report, CERN, Geneva, 2019.

[249] K. Akiba, M. Alexander, C. Bertella, A. Biolchini, A. Bitadze, et al. The lhcb velo upgrade module construction. *Journal of Instrumentation*, 19(06):P06023, jun 2024. doi: 10.1088/1748-0221/19/06/P06023.

[250] Victor Coco, Kazu Akiba, John Back, Claudia Bertella, Alexander Bitadze, et al. Velo Upgrade Module Nomenclature. Technical report, CERN, Geneva, 2019.

[251] The LHCb Collaboration. LHCb VELO Upgrade Technical Design Report. Technical report, 2013.

[252] Thomas Bird. Flavour studies with lhcb: b-meson mixing, lepton- flavour violation and the velo upgrade, March 2017.

[253] O Callot. Velo tracking for the High Level Trigger. Technical report, CERN, Geneva, 2003.

[254] O Callot. Online Pattern Recognition. Technical report, CERN, Geneva, 2004.

[255] D Hutchcroft. VELO Pattern Recognition. Technical report, CERN, Geneva, 2007.

[256] O Callot. FastVelo, a fast and efficient pattern recognition package for the Velo. Technical report, CERN, Geneva, 2011.

[257] T Bird, T Britton, O Callot, V Coco, P Collins, et al. VP Simulation and Track Reconstruction. Technical report, CERN, Geneva, 2013.

[258] Alexey Badalov, Daniel Campora, Gianmaria Collazuol, Marco Corvo, Stefano Gallorini, et al. GPGPU opportunities at the LHCb trigger. Technical report, CERN, Geneva, 2014.

[259] Alexey Pavlovich Badalov. *Coprocessor integration for real-time event processing in particle physics detectors*. PhD thesis, Universitat Ramon Llull. La Salle, 2016.

[260] Michel De Cian, Agnieszka Dziurda, Vladimir Gligorov, Christoph Hasse, Wouter Hulsbergen, et al. Status of HLT1 sequence and path towards 30 MHz. Technical report, CERN, Geneva, 2018.

[261] Daniel Hugo Cámpora Pérez. Optimization of high-throughput real-time processes in physics reconstruction. 2019.

[262] Agnieszka Dziurda, Maciej Giza, Vladimir V. Gligorov, Wouter Hulsbergen, Bogdan Kutsenko, et al. A parallel algorithm for fast reconstruction of primary vertices on heterogeneous architectures. *Eur. Phys. J. C*, 85(6):609, 2025. doi: 10.1140/epjc/s10052-025-14225-7.

[263] Federico Lazzari, Giovanni Bassi, Riccardo Cenci, Michael J Morello, and Giovanni Punzi. Real-time cluster finding for LHCb silicon pixel VELO detector using FPGA. *J. Phys.: Conf. Ser.*, 1525(1):012044, 2020. doi: 10.1088/1742-6596/1525/1/012044.

[264] Michel De Cian, Stephen Farry, Paul Seyfert, and Sascha Stahl. Fast neural-net based fake track rejection in the LHCb reconstruction. Technical report, CERN, Geneva, 2017.

[265] Roel Aaij, Lucio Anderlini, Sean Benson, Marco Cattaneo, Philippe Charpentier, et al. Selection and processing of calibration samples to measure the particle identification performance of the LHCb experiment in Run 2. *EPJ Tech. Instrum.*, 6(1):1, 2019. doi: 10.1140/epjti/s40485-019-0050-z.

[266] Tatiana Likhomanenko, Philip Ilten, Egor Khairullin, Alex Rogozhnikov, Andrey Ustyuzhanin, and Michael Williams. Lhcb topological trigger reoptimization. *Journal of Physics: Conference Series*, 664(8):082025, December 2015. ISSN 1742-6596. doi: 10.1088/1742-6596/664/8/082025.

[267] Ouail Kitouni, Niklas Nolte, and Mike Williams. Robust and provably monotonic networks. *Machine Learning: Science and Technology*, 4(3):035020, August 2023. ISSN 2632-2153. doi: 10.1088/2632-2153/aced80.

[268] Maciej Majewski. Analysis, and machine learning anomaly detection of the VELO-LHCb calibration. *J. Phys. : Conf. Ser.*, 1337(1):012006, 2019. doi: 10.1088/1742-6596/1337/1/012006.

[269] Pawel Kopciewicz, Kazu Akiba, Tomasz Szumlak, Sebastian Piotr Sitko, William Barter, et al. Simulation and Optimization Studies of the LHCb Beetle Readout ASIC and Machine Learning Approach for Pulse Shape Reconstruction. Technical Report 18, CERN, Geneva, 2021.

[270] Rinnert, Kurt and Cristoforetti, Marco. Deep learning approach to track reconstruction in the upgraded velo. *EPJ Web Conf.*, 214:06038, 2019. doi: 10.1051/epjconf/201921406038.

[271] Christoph Hasse. Alternative approaches in the event reconstruction of lhcb, January 2020.

[272] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013.

[273] Paul Almasan, José Suárez-Varela, Krzysztof Rusek, Pere Barlet-Ros, and Albert Cabellos-Aparicio. Deep reinforcement learning meets graph neural networks: Exploring a routing optimization use case. *Computer Communications*, 196:184–194, December 2022. ISSN 0140-3664. doi: 10.1016/j.comcom.2022.09.029.

[274] Khushmeen Kaur Brar, Bhawna Goyal, Ayush Dogra, Mohammed Ahmed Mustafa, Rana Majumdar, Ahmed Alkhayyat, and Vinay Kukreja. Image segmentation review: Theoretical background and recent advances. *Information Fusion*, 114:102608, 2025. ISSN 1566-2535. doi: https://doi.org/10.1016/j.inffus.2024.102608.

[275] Torbjörn Sjöstrand, Stephen Mrenna, and Peter Skands. A brief introduction to pythia 8.1. *Computer Physics Communications*, 178(11):852–867, June 2008. ISSN 0010-4655. doi: 10.1016/j.cpc.2008.01.036.

[276] I Belyaev, T Brambach, N H Brook, N Gauvin, G Corti, et al. Handling of the generation of primary events in Gauss, the LHCb simulation framework. Technical report, CERN, Geneva, 2010.

[277] David J. Lange. The evtgen particle decay simulation package. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 462(1):152–155, 2001. ISSN 0168-9002. doi: https://doi.org/10.1016/S0168-9002(01)00089-4. BEAUTY2000, Proceedings of the 7th Int. Conf. on B-Physics at Hadron Machines.

[278] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, et al. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2.

[279] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, et al. Pytorch: An imperative style, high-performance deep learning library, 2019.

[280] Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, et al. Deep graph library: A graph-centric, highly-performant package for graph neural networks, 2020.

[281] Mufei Li. Dgl faq, 2021. https://discuss.dgl.ai/t/frequently-asked-questions-faq/1681, Accessed 12/12/2024.

[282] PyTorch Foundation. Pytorch - bcewithlogitsloss., 2024. https://pytorch.org/docs/stable/generated/torch.nn.BCEWithLogitsLoss.html Accessed 23/09/2024.

[283] Google. Classification: Roc and auc, in machine learning crash course, 2024. https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc, Accessed 29/12/2024.

[284] DGL Contributors. Dgl documentation. https://docs.dgl.ai/, Accessed 12/12/2024.

[285] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V. Chawla. Heterogeneous graph neural network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, page 793–803, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362016. doi: 10.1145/3292500.3330961.

[286] K. Camilus and Govindan V K. A review on graph based segmentation. *International Journal of Image, Graphics and Signal Processing*, 4, 06 2012. doi: 10.5815/ijigsp.2012.05.01.

[287] Savannah Thais and Gage DeZoort. Instance segmentation gnns for one-shot conformal tracking at the lhc, 2021.

[288] Anthony Correia, Fotis I. Giasemis, Nabil Garroum, Vladimir Vava Gligorov, and Bertrand Granado. Graph neural network-based pipeline for track finding in the velo at lhcb, 2025.

[289] Anthony Correia and Fotis Giasemis. Etx4velo, 2023. *Cern Courier*, https://etx4velo.docs.cern.ch/.

Part V

Appendices

Appendix A

Declarations and Permissions

A.1 Pertaining to Part II

Accompanying declarations, pertinent to the Code of Practice on Assessment Appendix 7.2, relating to the use of material published in

> T. Ackernley, G. Casse, and M. Cristoforetti. Proton path reconstruction for proton computed tomography using neural networks. *Physics in Medicine & Biology, 66(7):075015, apr 2021.* doi: 10.1088/1361-6560/abf00f. [3]
>
>

A.1.1 Candidate Declaration

> This work was pursued as a collaborative effort, primarily with Dr. Cristoforetti, and we both contributed and collaborated on all aspects of said work, including writing of the paper text. I was heavily involved in producing the Monte Carlo simulations and developing the PPNN model, and Dr. Cristoforetti contributed extensively towards the analysis and discussion, particularly that covered in Sections 6.1.2 through 6.1.3 and production of Figures 6.2 through 6.6. Prof. Casse principally contributed to preparation and publication of the final manuscript.
>
> Thomas Ackernley

A.1.2 Co-author Declarations

I give my permission for you to include material from our jointly authored paper "Proton path reconstruction for proton computed tomography using neural networks" in your thesis.

22 September 2025, Marco Cristoforetti

This message to confirm that I am very much in favour for using any extract from the paper arxiv 2010.00427 (or doi:10.1088/1361-6560/abf00f), of which we are co-authors, for your thesis.

22 September 2025, Gianluigi Casse

A.2 Pertaining to Part III

Relating to the use of the model referred to here as the Hybrid Model.

I hereby grant Thomas Ackernley permission to use all material, code and private comunication I shared with him in his PhD research and also freely quote and reference it in his PhD thesis.

25 September 2025, Kurt Rinnert

Appendix B

VELO Module Positions

Each station within the VELO is numbered consecutively from 0 to 25, beginning with the negative z-direction. Similarly, every module is assigned a number $0-51$, with even and odd numbers corresponding to sides A and C respectively, and equally all sensor tiles $0-207$[250]. The position of each module in z, identified in the above manner and with corresponding station and sensors, is provided in Table B.1. Due to material thicknesses and tile arrangement, recorded position measurements are marginally displaced in z from those indicated here. Further details on nomenclature can be found in [250].

Station	C-side			A-side			Mean z
	Module	Sensors	z [mm]	Module	Sensors	z [mm]	[mm]
0	0	$0-3$	-287.50	1	$4-7$	-275.00	-281.25
1	2	$8-11$	-262.50	3	$12-15$	-250.00	-256.25
2	4	$16-19$	-237.50	5	$20-23$	-225.00	-231.25
3	6	$24-27$	-212.50	7	$28-31$	-200.00	-206.25
4	8	$32-35$	-137.50	9	$36-39$	-125.00	-131.25
5	10	$40-43$	-62.50	11	$44-47$	-50.00	-56.25
6	12	$48-51$	-37.50	13	$52-55$	-25.00	-31.25
7	14	$56-59$	-12.5	15	$60-63$	0.00	-6.25
8	16	$64-67$	12.50	17	$68-71$	25.00	18.75
9	18	$72-75$	37.50	19	$76-79$	50.00	43.75
10	20	$80-83$	62.50	21	$84-87$	75.00	68.75
11	22	$88-91$	87.50	23	$92-95$	100.00	93.75
12	24	$96-99$	112.50	25	$100-103$	125.00	118.75
13	26	$104-107$	137.50	27	$108-111$	150.00	143.75
14	28	$112-115$	162.50	29	$116-119$	175.00	168.75
15	30	$120-123$	187.50	31	$124-127$	200.00	193.75
16	32	$128-131$	212.50	33	$132-135$	225.00	218.75
17	34	$136-139$	237.50	35	$140-143$	250.00	243.75
18	36	$144-147$	262.50	37	$148-151$	275.00	268.75
19	38	$152-155$	312.50	39	$156-159$	325.00	318.75
20	40	$160-168$	387.50	41	$164-167$	400.00	396.75
21	42	$168-173$	487.50	43	$172-175$	500.00	493.75
22	44	$176-179$	587.50	45	$180-183$	600.00	593.75
23	46	$184-187$	637.50	47	$188-191$	650.00	643.75
24	48	$192-195$	687.50	49	$196-199$	700.00	693.75
25	50	$200-203$	737.50	51	$204-207$	750.00	743.75

Table B.1: Overview of the VELO module positions. Adapted from [250].

Appendix C

Results Tables for Chapter 13

Model	Accuracy		Balanced Accuracy		True Positive Rate		True Negative Rate		ROC AUC
		±		±		±		±	
Hybrid	0.9824	0.007318	0.9770	0.008593	0.9713	0.015024	0.9827	0.008651	0.9970
Basic	0.9719	0.008199	0.9753	0.007710	0.9790	0.011976	0.9716	0.010197	0.9970

Table C.1: Results corresponding to Figure 13.1(a).

Model	Accuracy		Balanced Accuracy		True Positive Rate		True Negative Rate		ROC AUC
		±		±		±		±	
Hybrid	0.9733	0.015544	0.9736	0.021810	0.9822	0.011681	0.9650	0.041860	0.9963
Basic	0.9694	0.017429	0.9757	0.021307	0.9914	0.007549	0.9601	0.041762	0.9973

Table C.2: Results corresponding to Figure 13.1(b).

Model	Efficiency		Clone Rate		Fake Rate	
		±		±		±
Hybrid	0.9296	0.037848	0.0158	0.015513	0.0020	0.004903
Basic	0.9467	0.033270	0.0145	0.014321	0.0244	0.029888

Table C.3: Results corresponding to Figure 13.1(c).

Model	Accuracy		Balanced Accuracy		True Positive Rate		True Negative Rate		ROC AUC
		±		±		±		±	
Hybrid 2 M.	0.9806	0.017446	0.9754	0.024846	0.9702	0.044390	0.9807	0.022122	0.9970
Hybrid 3 M.	0.9808	0.016409	0.9760	0.022083	0.9712	0.038231	0.9808	0.021804	0.9972
Hybrid 8 M.	0.9809	0.013947	0.9735	0.016370	0.9659	0.027991	0.9812	0.017656	0.9963
Hybrid 26 M.	0.9824	0.007318	0.9770	0.008593	0.9713	0.015024	0.9827	0.008651	0.9970
Sng. Net. 2 M.	0.9748	0.019052	0.9753	0.023300	0.9763	0.039146	0.9743	0.024878	0.9968
Sng. Net. 3 M.	0.9703	0.019301	0.9745	0.020734	0.9795	0.031573	0.9695	0.026236	0.9967
Sng. Net. 8 M.	0.9668	0.015694	0.9716	0.015009	0.9771	0.021468	0.9661	0.021136	0.9954
Sng. Net. 26 M.	0.9719	0.008199	0.9753	0.007710	0.9790	0.011976	0.9716	0.010197	0.9970
Sng. Net. 52 M.	0.9712	0.004791	0.9784	0.004436	0.9859	0.006364	0.9709	0.005903	0.9978
Sep. Net. 2 M.	0.9791	0.018216	0.9751	0.024606	0.9712	0.043162	0.9790	0.023476	0.9969
Sep. Net. 3 M.	0.9757	0.017594	0.9753	0.021241	0.9754	0.034786	0.9753	0.023615	0.9969
Sep. Net. 8 M.	0.9670	0.015391	0.9711	0.015156	0.9758	0.022209	0.9663	0.020551	0.9956

Table C.4: Results corresponding to Figure 13.2. The regular single network Basic model is abbreviated as Sng. Net, and the separate network Basic variant as Sep. Net.. The number of modules included is abbreviated as M..

Model	Accuracy		Balanced Accuracy		True Positive Rate		True Negative Rate		ROC AUC
		±		±		±		±	
Hybrid 2 M.	0.9806	0.017446	0.9754	0.024846	0.9702	0.044390	0.9807	0.022122	0.9970
Hybrid 3 M.	0.9808	0.016409	0.9760	0.022083	0.9712	0.038231	0.9808	0.021804	0.9972
Hybrid 8 M.	0.9809	0.013947	0.9735	0.016370	0.9659	0.027991	0.9812	0.017656	0.9963
Hybrid 26 M.	0.9824	0.007318	0.9770	0.008593	0.9713	0.015024	0.9827	0.008651	0.9970
Basic 2 M.	0.9748	0.019052	0.9753	0.023300	0.9763	0.039146	0.9743	0.024878	0.9968
Basic 3 M.	0.9703	0.019301	0.9745	0.020734	0.9795	0.031573	0.9695	0.026236	0.9967
Basic 8 M.	0.9668	0.015694	0.9716	0.015009	0.9771	0.021468	0.9661	0.021136	0.9954
Basic 26 M.	0.9719	0.008199	0.9753	0.007710	0.9790	0.011976	0.9716	0.010197	0.9970
Basic 52 M.	0.9712	0.004791	0.9784	0.004436	0.9859	0.006364	0.9709	0.005903	0.9978
w/o z 2 M.	0.9761	0.018987	0.9757	0.023354	0.9758	0.039450	0.9757	0.024687	0.9972
w/o z 3 M.	0.9725	0.018563	0.9749	0.020900	0.9781	0.032428	0.9718	0.025280	0.9971
w/o z 8 M.	0.9666	0.015324	0.9712	0.014771	0.9766	0.021556	0.9659	0.020439	0.9960
w/o z 26 M.	0.9654	0.008533	0.9700	0.008179	0.9751	0.012955	0.9650	0.010456	0.9958
w/o z 52 M.	0.9744	0.004339	0.9762	0.005121	0.9781	0.008686	0.9743	0.005426	0.9971
w/o z, sc. 2 M.	0.9729	0.019544	0.9745	0.023343	0.9768	0.038563	0.9723	0.025845	0.9969
w/o z, sc. 3 M.	0.9693	0.019173	0.9743	0.020146	0.9802	0.030193	0.9685	0.025861	0.9969
w/o z, sc. 8 M.	0.9660	0.015524	0.9713	0.014746	0.9772	0.021069	0.9653	0.020775	0.9959
w/o z, sc. 26 M.	0.9587	0.009087	0.9667	0.008492	0.9755	0.013046	0.9580	0.011300	0.9949
w/o z, sc. 52 M.	0.9673	0.004714	0.9740	0.004856	0.9811	0.007787	0.9670	0.005847	0.9962

Table C.5: Results corresponding to Figure 13.3. Sc. indicates the exclusion of scaling. The number of modules included is abbreviated as M..

Model	Accuracy		Balanced Accuracy		True Positive Rate		True Negative Rate		ROC AUC
		±		±		±		±	
Hybrid	0.9824	0.007318	0.9770	0.008593	0.9713	0.015024	0.9827	0.008651	0.9970
0.4 Mult.	0.9840	0.007226	0.9770	0.008520	0.9696	0.015166	0.9844	0.008429	0.9972
0.5 Mult.	0.9819	0.007343	0.9774	0.008185	0.9727	0.014270	0.9821	0.008723	0.9973
0.6 Mult.	0.9764	0.007884	0.9765	0.007854	0.9767	0.012627	0.9763	0.009722	0.9968
0.7 Mult.	0.9752	0.008394	0.9750	0.008312	0.9750	0.013554	0.9750	0.010379	0.9965
0.8 Mult.	0.9763	0.007850	0.9769	0.007774	0.9777	0.012428	0.9761	0.009692	0.9973
0.9 Mult.	0.9731	0.008069	0.9755	0.007807	0.9782	0.012267	0.9728	0.009969	0.9970
1.0 Mult.	0.9719	0.008199	0.9753	0.007710	0.9790	0.011976	0.9716	0.010197	0.9970
1.1 Mult.	0.9663	0.008472	0.9734	0.007637	0.9811	0.010992	0.9657	0.010910	0.9965
1.2 Mult.	0.9607	0.009072	0.9722	0.007408	0.9846	0.009732	0.9598	0.011470	0.9967
1.3 Mult.	0.9588	0.009317	0.9713	0.007703	0.9849	0.009475	0.9578	0.012038	0.9971
1.4 Mult.	0.9600	0.009276	0.9716	0.007545	0.9841	0.009791	0.9591	0.011609	0.9969
1.5 Mult.	0.9590	0.009260	0.9713	0.007507	0.9844	0.009694	0.9581	0.011629	0.9959
1.6 Mult.	0.9522	0.009726	0.9683	0.007655	0.9857	0.009370	0.9510	0.012211	0.9963

Table C.6: Results corresponding to Figure 13.4. The multiplier included is abbreviated as Mult..

Model	Accuracy		Balanced Accuracy		True Positive Rate		True Negative Rate		ROC AUC
		±		±		±		±	
Hybrid	0.9733	0.015544	0.9736	0.021810	0.9822	0.011681	0.9650	0.041860	0.9963
0.4 Mult.	0.9812	0.013896	0.9798	0.017358	0.9807	0.012205	0.9789	0.032361	0.9975
0.5 Mult.	0.9806	0.013776	0.9804	0.017363	0.9835	0.011282	0.9773	0.032710	0.9977
0.6 Mult.	0.9769	0.015597	0.9791	0.019592	0.9871	0.009796	0.9712	0.037839	0.9976
0.7 Mult.	0.9796	0.013880	0.9807	0.017972	0.9862	0.010242	0.9752	0.034273	0.9977
0.8 Mult.	0.9735	0.016924	0.9778	0.020726	0.9896	0.008438	0.9659	0.040433	0.9973
0.9 Mult.	0.9734	0.016928	0.9776	0.021105	0.9895	0.008587	0.9657	0.041235	0.9976
1.0 Mult.	0.9694	0.017429	0.9757	0.021307	0.9914	0.007549	0.9601	0.041762	0.9973
1.1 Mult.	0.9735	0.015739	0.9777	0.019767	0.9894	0.008473	0.9661	0.038466	0.9975
1.2 Mult.	0.9728	0.016500	0.9774	0.020540	0.9899	0.008408	0.9650	0.040068	0.9976
1.3 Mult.	0.9743	0.016083	0.9783	0.019959	0.9894	0.008534	0.9673	0.038821	0.9978
1.4 Mult.	0.9685	0.018385	0.9753	0.022246	0.9920	0.007265	0.9586	0.043751	0.9975
1.5 Mult.	0.9681	0.017558	0.9752	0.021375	0.9923	0.006994	0.9582	0.041984	0.9977
1.6 Mult.	0.9714	0.016402	0.9766	0.020162	0.9901	0.008109	0.9632	0.039307	0.9974

Table C.7: Results corresponding to Figure 13.5(a). The multiplier included is abbreviated as Mult..

Model	Accuracy		Balanced Accuracy		True Positive Rate		True Negative Rate		ROC AUC
		±		±		±		±	
Hybrid	0.9733	0.015544	0.9736	0.021810	0.9822	0.011681	0.9650	0.041860	0.9963
0.4 Mult.	0.9726	0.016665	0.9706	0.022189	0.9727	0.015084	0.9686	0.041900	0.9952
0.5 Mult.	0.9742	0.015782	0.9747	0.022248	0.9832	0.011183	0.9662	0.042888	0.9963
0.6 Mult.	0.9794	0.013914	0.9790	0.018306	0.9824	0.011922	0.9757	0.034572	0.9973
0.7 Mult.	0.9765	0.015489	0.9785	0.019791	0.9871	0.009479	0.9700	0.038201	0.9973
0.8 Mult.	0.9741	0.016048	0.9770	0.020534	0.9880	0.009168	0.9661	0.039893	0.9972
0.9 Mult.	0.9713	0.017479	0.9760	0.020998	0.9892	0.008315	0.9629	0.041060	0.9967
1.0 Mult.	0.9694	0.017429	0.9757	0.021307	0.9914	0.007549	0.9601	0.041762	0.9973
1.1 Mult.	0.9746	0.016188	0.9795	0.018997	0.9906	0.007810	0.9684	0.037031	0.9980
1.2 Mult.	0.9746	0.016115	0.9797	0.017745	0.9898	0.008120	0.9696	0.034374	0.9979
1.3 Mult.	0.9760	0.015571	0.9812	0.016925	0.9910	0.007511	0.9714	0.032859	0.9984
1.4 Mult.	0.9739	0.015905	0.9799	0.017820	0.9911	0.007524	0.9687	0.034680	0.9980
1.5 Mult.	0.9794	0.013998	0.9826	0.015729	0.9894	0.008472	0.9758	0.030112	0.9983
1.6 Mult.	0.9798	0.014222	0.9835	0.015333	0.9904	0.007848	0.9765	0.029503	0.9986

Table C.8: Results corresponding to Figure 13.5(b). The multiplier included is abbreviated as Mult..

Model	Efficiency		Clone Rate		Fake Rate	
		±		±		±
Hybrid	0.9296	0.037848	0.0158	0.015513	0.0020	0.004903
P.O. 0.4 Mult.	0.9416	0.033400	0.0162	0.015367	0.0042	0.007636
P.O. 0.5 Mult.	0.9436	0.033152	0.0149	0.014683	0.0047	0.008449
P.O. 0.6 Mult.	0.9445	0.033353	0.0146	0.014392	0.0084	0.012441
P.O. 0.7 Mult.	0.9442	0.033648	0.0138	0.014017	0.0085	0.012840
P.O. 0.8 Mult.	0.9465	0.032384	0.0147	0.014494	0.0062	0.010319
P.O. 0.9 Mult.	0.9486	0.031561	0.0152	0.014708	0.0124	0.016876
P.O. 1.0 Mult.	0.9467	0.033270	0.0145	0.014321	0.0244	0.029888
P.O. 1.1 Mult.	0.9435	0.034483	0.0141	0.014177	0.0193	0.024706
P.O. 1.2 Mult.	0.9459	0.033267	0.0143	0.014168	0.0169	0.021607
P.O. 1.3 Mult.	0.9415	0.035193	0.0144	0.014203	0.0250	0.030958
P.O. 1.4 Mult.	0.9414	0.035301	0.0146	0.014255	0.0234	0.029215
P.O. 1.5 Mult.	0.9388	0.035562	0.0158	0.014892	0.0185	0.023957
P.O. 1.6 Mult.	0.9427	0.034792	0.0155	0.014589	0.0322	0.038535
T.O. 0.4 Mult.	0.9386	0.034428	0.0159	0.015175	0.0033	0.006573
T.O. 0.5 Mult.	0.9377	0.034815	0.0152	0.014794	0.0034	0.006878
T.O. 0.6 Mult.	0.9434	0.033907	0.0147	0.014571	0.0082	0.012279
T.O. 0.7 Mult.	0.9466	0.032407	0.0152	0.014722	0.0066	0.010517
T.O. 0.8 Mult.	0.9430	0.034201	0.0146	0.014329	0.0129	0.017913
T.O. 0.9 Mult.	0.9441	0.033683	0.0142	0.014166	0.0098	0.014274
T.O. 1.0 Mult.	0.9467	0.033270	0.0145	0.014321	0.0244	0.029888
T.O. 1.1 Mult.	0.9450	0.033190	0.0142	0.014147	0.0094	0.013564
T.O. 1.2 Mult.	0.9460	0.033136	0.0141	0.014184	0.0117	0.016354
T.O. 1.3 Mult.	0.9473	0.032458	0.0139	0.014039	0.0134	0.018154
T.O. 1.4 Mult.	0.9464	0.033598	0.0140	0.014028	0.0222	0.027732
T.O. 1.5 Mult.	0.9479	0.032961	0.0138	0.013938	0.0229	0.028488
T.O. 1.6 Mult.	0.9452	0.033148	0.0146	0.014395	0.0136	0.018806

Table C.9: First part of results corresponding to Figure 13.6, continued in Table C.10. The multiplier included is abbreviated as Mult.. Inclusion of the multiplier in only the Pair Network, Triplet Network or both is abbreviated as P.O., T.O or B. respectively.

Model	Efficiency		Clone Rate		Fake Rate	
		±		±		±
B. 0.4 Mult.	0.9122	0.041471	0.0189	0.016790	0.0014	0.004455
B. 0.5 Mult.	0.9369	0.035183	0.0160	0.015288	0.0024	0.005399
B. 0.6 Mult.	0.9391	0.034395	0.0169	0.015626	0.0041	0.007492
B. 0.7 Mult.	0.9424	0.033912	0.0153	0.014788	0.0073	0.011432
B. 0.8 Mult.	0.9438	0.033518	0.0148	0.014585	0.0072	0.011293
B. 0.9 Mult.	0.9400	0.035021	0.0153	0.014670	0.0152	0.020891
B. 1.0 Mult.	0.9467	0.033270	0.0145	0.014321	0.0244	0.029888
B. 1.1 Mult.	0.9470	0.033017	0.0142	0.014221	0.0180	0.023409
B. 1.2 Mult.	0.9424	0.034895	0.0149	0.014507	0.0210	0.026383
B. 1.3 Mult.	0.9482	0.032778	0.0136	0.013847	0.0225	0.028687
B. 1.4 Mult.	0.9433	0.034567	0.0145	0.014194	0.0262	0.032037
B. 1.5 Mult.	0.9484	0.032313	0.0139	0.013996	0.0191	0.024595
B. 1.6 Mult.	0.9468	0.033134	0.0136	0.013876	0.0202	0.025710

Table C.10: Second part of results corresponding to Figure 13.6, continuing from Table C.9. The multiplier included is abbreviated as Mult.. Inclusion of the multiplier in only the Pair Network, Triplet Network or both is abbreviated as P.O., T.O or B. respectively.

Model	Accuracy		Balanced Accuracy		True Positive Rate		True Negative Rate		ROC AUC
		±		±		±		±	
Hybrid	0.9824	0.007318	0.9770	0.008593	0.9713	0.015024	0.9827	0.008651	0.9970
3 L. 16 N.	0.9635	0.009406	0.9720	0.007839	0.9812	0.010877	0.9628	0.011659	0.9967
3 L. 24 N.	0.9723	0.008360	0.9762	0.007560	0.9805	0.011238	0.9719	0.010457	0.9973
3 L. 32 N.	0.9750	0.008023	0.9767	0.007676	0.9786	0.012050	0.9748	0.009952	0.9973
3 L. 64 N.	0.9669	0.008681	0.9747	0.007493	0.9831	0.010324	0.9662	0.010979	0.9972
4 L. 16 N.	0.9755	0.007840	0.9766	0.007744	0.9777	0.012457	0.9754	0.009570	0.9971
4 L. 24 N.	0.9746	0.008114	0.9766	0.007706	0.9788	0.012111	0.9744	0.009991	0.9969
4 L. 32 N.	0.9736	0.008215	0.9770	0.007491	0.9808	0.011164	0.9733	0.010251	0.9975
4 L. 64 N.	0.9687	0.008523	0.9753	0.007512	0.9824	0.010590	0.9682	0.010858	0.9971
6 L. 16 N.	0.9734	0.008120	0.9750	0.007831	0.9768	0.012396	0.9732	0.010011	0.9965
6 L. 24 N.	0.9770	0.007782	0.9765	0.008022	0.9760	0.013021	0.9770	0.009695	0.9968
6 L. 32 N.	0.9751	0.007695	0.9761	0.007764	0.9772	0.012453	0.9750	0.009521	0.9969
6 L. 64 N.	0.9797	0.007610	0.9779	0.007805	0.9762	0.012959	0.9797	0.009131	0.9974
8 L. 16 N.	0.9748	0.007813	0.9765	0.007769	0.9784	0.012360	0.9746	0.009669	0.9960
8 L. 24 N.	0.9708	0.008499	0.9753	0.007699	0.9802	0.011479	0.9703	0.010593	0.9962
8 L. 32 N.	0.9717	0.008095	0.9764	0.007421	0.9815	0.010984	0.9713	0.010169	0.9975
8 L. 64 N.	0.9779	0.007800	0.9774	0.007912	0.9769	0.012900	0.9779	0.009525	0.9973
12 L. 16 N.	0.9777	0.007903	0.9773	0.007770	0.9770	0.012523	0.9776	0.009629	0.9973
12 L. 24 N.	0.9760	0.007832	0.9776	0.007523	0.9794	0.011656	0.9757	0.009785	0.9976
12 L. 32 N.	0.9717	0.008311	0.9757	0.007732	0.9799	0.011594	0.9713	0.010389	0.9971
12 L. 64 N.	0.9752	0.007930	0.9771	0.007644	0.9794	0.011947	0.9749	0.009858	0.9974
15 L. 16 N.	0.9713	0.008412	0.9762	0.007606	0.9815	0.011153	0.9709	0.010545	0.9974
15 L. 24 N.	0.9764	0.007882	0.9775	0.007541	0.9789	0.011923	0.9762	0.009625	0.9975
15 L. 32 N.	0.9771	0.007785	0.9775	0.007731	0.9781	0.012219	0.9770	0.009758	0.9974
15 L. 64 N.	0.9748	0.007948	0.9771	0.007578	0.9796	0.011776	0.9746	0.009815	0.9974

Table C.11: Results corresponding to Figure 13.7. The number of layers is abbreviated as L., and the number of neurons per layer N..

Model	Accuracy		Balanced Accuracy		True Positive Rate		True Negative Rate		ROC AUC
		±		±		±		±	
Hybrid	0.9824	0.007318	0.9770	0.008593	0.9713	0.015024	0.9827	0.008651	0.9970
Basic	0.9719	0.008199	0.9753	0.007710	0.9790	0.011976	0.9716	0.010197	0.9970
Interaction	0.9937	0.004877	0.9944	0.003952	0.9953	0.003958	0.9935	0.006998	0.9997

Table C.12: Results corresponding to Figure 13.8(a).

Model	Accuracy		Balanced Accuracy		True Positive Rate		True Negative Rate		ROC AUC
		±		±		±		±	
Hybrid	0.9733	0.015544	0.9736	0.021810	0.9822	0.011681	0.9650	0.041860	0.9963
Basic	0.9694	0.017429	0.9757	0.021307	0.9914	0.007549	0.9601	0.041762	0.9973
Interaction	0.9736	0.011162	0.9729	0.021078	0.9659	0.016437	0.9801	0.041293	0.9972
Basic - Int.	0.9863	0.010940	0.9691	0.014820	0.9426	0.024903	0.9957	0.018452	0.9987
Int. - Basic	0.9670	0.013778	0.9592	0.030291	0.9686	0.016481	0.9499	0.061481	0.9940

Table C.13: Results corresponding to Figure 13.8(b). Basic - Int. denotes using the Basic Network for the pair stage and Interaction Network for the triplet stage; and vice versa for Int. - Basic.

Model	Efficiency		Clone Rate		Fake Rate	
		±		±		±
Hybrid	0.9296	0.037848	0.0158	0.015513	0.0020	0.004903
Basic	0.9467	0.033270	0.0145	0.014321	0.0244	0.029888
Interaction	0.9437	0.033278	0.0107	0.012494	0.0000	0.000404
Basic - Int.	0.8932	0.045979	0.0235	0.019165	0.0001	0.001031
Int. - Basic	0.9429	0.034643	0.0209	0.017679	0.0001	0.001061

Table C.14: Results corresponding to Figure 13.8(c). Basic - Int. denotes using the Basic Network for the pair stage and Interaction Network for the triplet stage; and vice versa for Int. - Basic.

Model	Accuracy		Balanced Accuracy		True Positive Rate		True Negative Rate		ROC AUC
		±		±		±		±	
Hybrid	0.9824	0.007318	0.9770	0.008593	0.9713	0.015024	0.9827	0.008651	0.9970
Basic	0.9719	0.008199	0.9753	0.007710	0.9790	0.011976	0.9716	0.010197	0.9970
Sum	0.9937	0.004877	0.9944	0.003952	0.9953	0.003958	0.9935	0.006998	0.9997
Mean	0.9932	0.005201	0.9937	0.004386	0.9945	0.004646	0.9930	0.007701	0.9997
Max	0.9955	0.003783	0.9939	0.003897	0.9923	0.005498	0.9955	0.005617	0.9995
Min	0.9953	0.003958	0.9953	0.003567	0.9954	0.003817	0.9952	0.006004	0.9998

Table C.15: Results corresponding to Figure 13.9.

Model	Accuracy		Balanced Accuracy		True Positive Rate		True Negative Rate		ROC AUC
		±		±		±		±	
Hybrid	0.9824	0.007318	0.9770	0.008593	0.9713	0.015024	0.9827	0.008651	0.9970
Basic	0.9719	0.008199	0.9753	0.007710	0.9790	0.011976	0.9716	0.010197	0.9970
Latent	0.9937	0.004877	0.9944	0.003952	0.9953	0.003958	0.9935	0.006998	0.9997
Inputs	0.9929	0.005552	0.9941	0.004280	0.9956	0.003784	0.9926	0.008014	0.9997
Differences	0.9936	0.004657	0.9928	0.004317	0.9921	0.005724	0.9935	0.006770	0.9996
None	0.9950	0.004011	0.9950	0.003561	0.9951	0.004088	0.9948	0.005919	0.9998

Table C.16: Results corresponding to Figure 13.10.

Model	Accuracy		Balanced Accuracy		True Positive Rate		True Negative Rate		ROC AUC
		±		±		±		±	
Hybrid	0.9824	0.007318	0.9770	0.008593	0.9713	0.015024	0.9827	0.008651	0.9970
2 Itr.	0.9923	0.005762	0.9929	0.004208	0.9937	0.004759	0.9921	0.007226	0.9996
4 Itr.	0.9937	0.004877	0.9944	0.003952	0.9953	0.003958	0.9935	0.006998	0.9997
6 Itr.	0.9944	0.004069	0.9944	0.003420	0.9945	0.004208	0.9943	0.005232	0.9997
8 Itr.	0.9951	0.003760	0.9941	0.003619	0.9931	0.005175	0.9950	0.005011	0.9996
10 Itr.	0.9926	0.004659	0.9942	0.003753	0.9961	0.003619	0.9923	0.006716	0.9997
12 Itr.	0.9947	0.004434	0.9937	0.004028	0.9928	0.005489	0.9946	0.006226	0.9995

Table C.17: Results corresponding to Figure 13.11(a). Number of layers is abbreviated as Itr..

Model	Accuracy		Balanced Accuracy		True Positive Rate		True Negative Rate		ROC AUC
		±		±		±		±	
Hybrid	0.9733	0.015544	0.9736	0.021810	0.9822	0.011681	0.9650	0.041860	0.9963
2 Itr.	0.9756	0.027407	0.9445	0.084145	0.9793	0.044700	0.9101	0.172020	0.9992
4 Itr.	0.9736	0.011162	0.9729	0.021078	0.9659	0.016437	0.9801	0.041293	0.9972
6 Itr.	0.9737	0.012045	0.9722	0.024929	0.9673	0.016575	0.9773	0.049706	0.9974
8 Itr.	0.9730	0.011425	0.9706	0.026795	0.9697	0.014764	0.9717	0.053590	0.9965
10 Itr.	0.9718	0.012593	0.9668	0.025524	0.9700	0.015912	0.9637	0.051255	0.9957
12 Itr.	0.9590	0.015284	0.9561	0.028564	0.9541	0.020444	0.9584	0.057373	0.9931

Table C.18: Results corresponding to Figure 13.11(b). Number of layers is abbreviated as Itr..

Model	Efficiency		Clone Rate		Fake Rate	
		±		±		±
Hybrid	0.9296	0.037848	0.0158	0.015513	0.0020	0.004903
2 Itr.	0.9718	0.023867	0.0149	0.014112	0.0036	0.006667
4 Itr.	0.9437	0.033278	0.0107	0.012494	0.0000	0.000404
6 Itr.	0.9367	0.035712	0.0089	0.011310	0.0000	0.000439
8 Itr.	0.9404	0.033916	0.0094	0.011696	0.0000	0.000498
10 Itr.	0.9319	0.037500	0.0172	0.016101	0.0001	0.001614
12 Itr.	0.8846	0.048701	0.0146	0.014844	0.0000	0.000605

Table C.19: Results corresponding to Figure 13.11(c). Number of layers is abbreviated as Itr..

Model	Accuracy		Balanced Accuracy		True Positive Rate		True Negative Rate		ROC AUC
		±		±		±		±	
Hybrid	0.9824	0.007318	0.9770	0.008593	0.9713	0.015024	0.9827	0.008651	0.9970
2 L. 24 N.	0.9937	0.004877	0.9944	0.003952	0.9953	0.003958	0.9935	0.006998	0.9997
2 L. 32 N.	0.9922	0.005935	0.9910	0.004932	0.9900	0.006696	0.9920	0.008016	0.9994
2 L. 64 N.	0.9936	0.005550	0.9935	0.004256	0.9936	0.004765	0.9934	0.007501	0.9997
3 L. 24 N.	0.9946	0.004307	0.9944	0.003970	0.9944	0.004697	0.9944	0.006529	0.9998
3 L. 32 N.	0.9944	0.004537	0.9940	0.003996	0.9938	0.004928	0.9942	0.006608	0.9997
3 L. 64 N.	0.9945	0.004280	0.9950	0.003691	0.9957	0.003716	0.9943	0.006383	0.9998
4 L. 32 N.	0.9944	0.004620	0.9950	0.003751	0.9958	0.003773	0.9942	0.006637	0.9998

Table C.20: Results corresponding to Figure 13.12. The number of layers is abbreviated as L., and the number of neurons per layer as N..

Model	Accuracy		Balanced Accuracy		True Positive Rate		True Negative Rate		ROC AUC
		±		±		±		±	
Hybrid	0.9733	0.015544	0.9736	0.021810	0.9822	0.011681	0.9650	0.041860	0.9963
Basic	0.9694	0.017429	0.9757	0.021307	0.9914	0.007549	0.9601	0.041762	0.9973
Interaction	0.9736	0.011162	0.9729	0.021078	0.9659	0.016437	0.9801	0.041293	0.9972
2 L. 64 N.	0.9773	0.010267	0.9756	0.021322	0.9723	0.014221	0.9791	0.042116	0.9975
3 L. 64 N.	0.9779	0.010144	0.9753	0.022548	0.9746	0.013355	0.9762	0.044870	0.9976
3 L. 128 N.	0.9757	0.010402	0.9751	0.020369	0.9685	0.015026	0.9819	0.039910	0.9974

Table C.21: Results corresponding to Figure 13.13. The number of layers is abbreviated as L., and the number of neurons per layer as N..

Hybrid	0.9296	0.037848	0.0158	0.015513	0.0020	0.004903
Hybrid Filter	0.9292	0.038020	0.0106	0.012384	0.0020	0.004928
Basic	0.9467	0.033270	0.0145	0.014321	0.0244	0.029888
Basic Filter	0.9464	0.033386	0.0110	0.012409	0.0244	0.029935
Interaction	0.9437	0.033278	0.0107	0.012494	0.0000	0.000404
Int. Filter	0.9433	0.033503	0.0064	0.009611	0.0000	0.000448

Table C.22: Results corresponding to Figure 13.14. The Interaction Network is abbreviated as Int. in the final entry.

Appendix D

Reproduction of 'Proton path reconstruction for pCT using Neural Networks'

Reproduction of the Accepted Manuscript, as found at https://arxiv.org/abs/2010.00427.

Proton path reconstruction for pCT using Neural Networks

T. Ackernley[1,2], G. Casse[1,2] and M. Cristoforetti[2]

[1]Oliver Lodge, Department of Physics, University of Liverpool, Oxford Street, L69 7ZE Liverpool, United Kingdom

[2]Fondazione Bruno Kessler (FBK), Via Sommarive, 18, Povo, 38123, Trento, Italy

E-mail: mcristofo@fbk.eu

September 2020

Abstract. The Most Likely Path formalism (MLP) is widely established as the most statistically precise method for proton path reconstruction in proton computed tomography (pCT). However, while this method accounts for small-angle Multiple Coulomb Scattering (MCS) and energy loss, inelastic nuclear interactions play an influential role in a significant number of proton paths. By applying cuts based on energy and direction, tracks influenced by nuclear interactions are largely discarded from the MLP analysis. In this work we propose a new method to estimate the proton paths based on a Deep Neural Network (DNN). Through this approach, estimates of proton paths equivalent to MLP predictions have been achieved in the case where only MCS occurs, together with an increased accuracy when nuclear interactions are present. Moreover, our tests indicate that the DNN algorithm can be considerably faster than the MLP algorithm.

1. Introduction

When reviewing recent developments in cancer treatment, proton beam therapy has seen rapid growth as an external beam radiotherapy technique, being increasingly favoured over traditional x-ray treatment for several tumours. Unlike in regular radiation treatment, protons deposit most energy near the end of their path, a well-established effect known as the Bragg peak. By exploiting this property, protons are used to target tumours while subjecting their surroundings to little or no damage. Such treatment is well suited for tumours located near sensitive organs or in young patients for whom excess radiation exposure is a significant long term concern (Tian et al. (2018), Hu et al. (2018), Foote et al. (2012)). Its capacity for depositing large amount of energy in a small volume increases the precision of treatment but so too the need to precisely locate the proton beam spot.

Accurate calibration of proton ranges relies on a detailed knowledge of the Relative Stopping Power, or RSP, of any tissue a proton will pass through along its path. Inaccurate placement of Bragg peaks can not only result in under-dosage of the target

but also in significant exposure to the sensitive areas whose presence warranted proton therapy initially. Satisfactory resolution of RSP remains a substantial obstacle in unlocking the full potential of proton therapy. Current treatment planning systems rely on converting x-ray linear attenuation coefficient measurements, made in Hounsfield Units (HU), to RSP. Unfortunately, the non-unique relationship between HU and RSP introduces errors in the range of $2-5\%$ (Beaton et al. 2019).

Proton computed tomography, or pCT, has been suggested as an alternative to overcome this problem. For proton therapy planning pCT offers the advantage of measuring proton RSP directly, removing conversion uncertainties by using the same particle for both planning and treatment (Doolan et al. (2015)).

For a given proton i, the line integral of the RSP is related to the energy loss using

$$\mathrm{WEPL}_i \equiv \int_{\Gamma_i} \mathrm{RSP}(x)\mathrm{d}x \approx \int_{E_i^{out}}^{E_i^{in}} \frac{\mathrm{d}E}{S_{water}(E)}$$

where $\Gamma_i \subset \mathbb{R}^3$ is the proton path, RSP(x) is the stopping power relative to water at position $x \in R^3$, E_i^{in} and E_i^{out} are the entrance and exit proton energies, and $S_{water}(E)$ is the stopping power of water for energy E. This integral is the Water Equivalent Path Length (WEPL). Starting from this equation, the pCT reconstruction problem can be mapped to that of reconstructing each individual protons path, combined with the calculation of WEPL (through the right side of the equation), to recover the RSP map. It is therefore crucial that the reconstruction of the proton path will be as accurate as possible. Indeed, the better the determination of the proton trajectories, the better the RSP calculation will be.

Image reconstruction using protons poses an additional challenge over standard x-ray CT: during passage through matter protons experience significant deflections through Multiple Coulomb Scattering (MCS), and, more rarely, nuclear interactions, resulting in non-trivial curved paths. The probability of nuclear reactions compared to ionization interactions is less than 1% for 200 MeV protons. As a consequence, the influence of nuclear interactions of protons with atomic nuclei can be treated as correction to the electromagnetic processes (Fippel et al. (2004)). Accurate reconstruction of these paths determines the achievable imaging resolution in proton computed tomography (pCT) and thus the exact dose distribution in proton therapy. Unlike with x-ray CT, in which photon number attenuation along straight propagation lines is considered, the pCT reconstruction process requires proton paths to be individually estimated to account for the curved trajectories if an improved resolution is to be achieved (Johnson (2017)).

This requirement excludes direct reuse of many well-developed image reconstruction methods developed in x-ray CT (Johnson (2017), Bovik (2009)). Iterative algebraic methods, such as the algebraic reconstruction technique (ART), have been proposed as plausible pCT image reconstruction methods (Li et al. (2006), Johnson (2017)), but the computational cost of these algorithms is considerably high. More efficient techniques are direct reconstruction methods, often following on from x-ray CT methods, who's

development is an active area of research, as discussed in Khellaf et al. (2020).

At the core of these methods is the Most Likely Path (MLP) formalism for the reconstruction of the single proton trajectory. While scattering remains an inherently probabilistic process, precluding the exact prediction of any single track, MLP is well established as the most statistically precise method to account for MCS processes (Schulte et al. (2008), Williams (2004), Collins-Fekete et al. (2015)). Since its introduction in 1994 (Schneider & Pedroni (1994)), the MLP formalism as presented in Schulte et al. (2008) has undergone various refinements for use in different application scenarios (Collins-Fekete et al. (2015), Collins-Fekete, Volz, Portillo, Beaulieu & Seco (2017), Collins-Fekete, Bär, Volz, Bouchard, Beaulieu & Seco (2017), Krah et al. (2019), Brooke & Penfold (2020)).

In addition to the entry and exit positions of the beam, the MLP algorithm utilises the angle between the direction of travel and the perpendicular to the phantom surface to significantly improve the prediction (Schneider & Pedroni (1994)). These quantities can be measured by modern pCT scanners systems (Johnson (2017)). However, while the formulation of MLP accounts for small-angle multiple Coulomb scattering (MCS) and small energy loss, nuclear interactions play an influential role in a significant number of proton trajectories (Johnson (2017)). Recommended practice is therefore to reduce the events influenced by nuclear interactions or large angle MCS through a 3σ cut on both the difference in energy and the difference in the direction of travel angle between entry and exit (Schulte et al. (2008)). Unfortunately, this results in a reduction of the protons available for the pCT image reconstruction and in an increase of the time needed to compute the relative stopping power map for proton therapy treatment planning. The need to estimate proton paths on a one by one basis, coupled with the inability to use many well-established x-ray CT reconstruction methods, comes with a significant computational burden (Johnson (2017)). Various avenues of research into overcoming this problem have been explored, from optimizing the computer code for MLP evaluation (McAllister et al. (2009)), to alternative approaches approximating MLP through cubic splines (Collins-Fekete et al. (2015)) or polynomial approximations (Krah et al. (2019)).

It is in this context that we introduce a new and original approach for the estimation of the proton paths based on Machine Learning, through utilisation of a Deep Neural Network. The Proton Path Neural Network (PPNN) is capable of reaching the same performance as MLP when this last is applicable, and exceeding it on a large fraction of paths influenced by nuclear interactions. Moreover, our tests indicate that PPNN exhibits significantly shorter execution time than the MLP approach.

The paper is organised as it follows. An overview of the Monte Carlo simulations used and the relevant physics environment is given in Section 2.1. This is followed in Section 2.2 by a description of the existing MLP proton path reconstruction, before the introduction of PPNN in Section 2.3. Studies comparing the reconstruction capabilities of PPNN against MLP are presented in Section 3.1, with further analysis into the methods' behavioural differences and the characteristics of corresponding tracks

introduced in Section 3.2 and Section 3.3 respectively. Initial work investigating performance on an inhomogeneous phantom is reviewed in Section 3.4. Comparison of execution times is covered in Section 3.5. Finally, a discussion of these results is presented in Section 4.

2. Materials and methods

2.1. Monte Carlo Simulation

The Monte Carlo simulations presented were performed using GATE v9.0 (Jan et al. (2011)), a framework built upon the widely used Geant4 10.6 Monte Carlo simulation toolkit (Agostinelli et al. (2003)). Simulations incorporating only electromagnetic processes were performed using the *emstandard* physics list. The impact of nuclear interactions, among a full regime of physics processes, were modeled using the *QGSP_BIC* physics list. In the discussion of the results, the choice of physics environment is indicated for each simulation.

Our main model consists of a sheet of water centred on the origin of a standard x-y-z coordinate system with a side length of 20 cm in the z-axis direction and arbitrarily large extents in x and y. Monoenergetic protons initialised at 200MeV are simulated through the phantom, originating at the central point of the phantom's $z = -10\,\text{cm}$ face, such that their initial direction of travel are orientated inwards and perpendicular to the face and parallel to the positive z-axis direction. For convenience in the following we redefine our coordinate axis such that the initial point of any trajectory is located at the origin, with particles initialised at a depth of 0 cm and extending in range to a depth of 20 cm. This arrangement is illustrated in Figure 1.

Each data set produced initially contained 10^6 events; however, only trajectories which traversed the full phantom depth were retained, reducing the number of events ultimately used. Typically this led to data sets in excess of $800,000$ events. For the purposes of this study, trajectories themselves are quantified as a series of spatial coordinates evenly distributed at 0.1 cm intervals, including both phantom faces. A total of 201 coordinate points represent a complete path through the phantom, consisting of 603 variables. As the z depth coordinates are therefore a fixed set of values shared by all trajectories, for predicting a track only the x and y variables need be considered. Similarly, the initial and final points of each trajectory are known for each track and so likewise neglected. Thus a track prediction consists of two sets of 199 points each, for a total of 398 variables per track.

In addition, as a first check of the robustness of the PPNN approach in inhomogeneous media, the procedure as stated was repeated using a phantom comprising 2 cm of water, 7 cm of skull, 2 cm of cortical-bone, 7 cm of skull, and 2 cm of water. For the purposes of this simulation, cortical-bone was defined using material data found in Berger et al. (2016). Due to the increased stopping power, to ensure that a large fraction of impinging protons successfully traverse the phantom's full length, a beam energy of

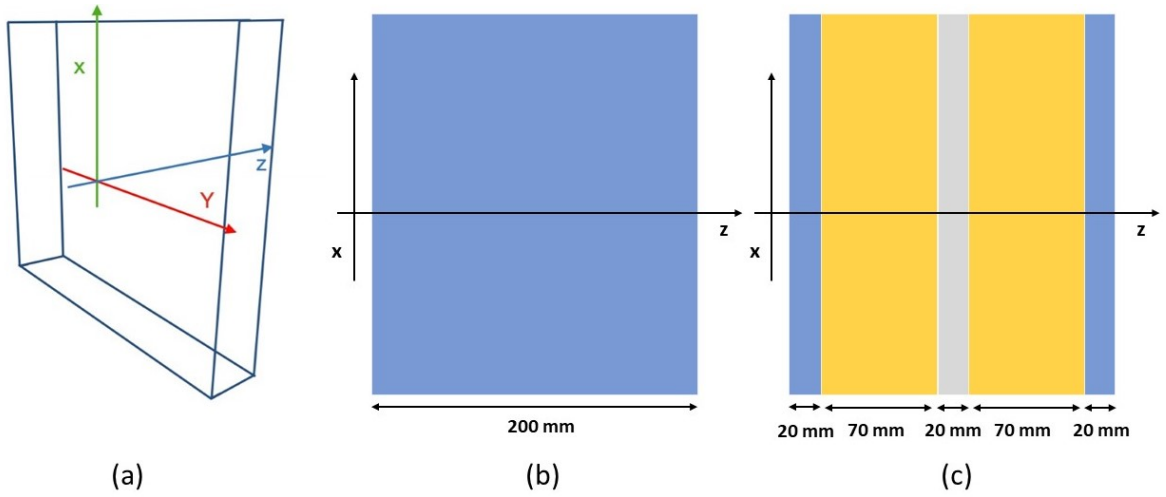


Figure 1. Illustration of the Monte Carlo geometry used in this study. 3D representation of the phantom space (a) and 2D projection on the $x-z$ plane for the water (b) and inhomogeneous phantom (c). Trajectories are only scored and monitored within the phantom volume itself. Note that for convenience we redefine our coordinate axis such that the initial point of each trajectory is located at the origin.

230 MeV was used. Additional simulations with an equivalently sized water phantom were carried out as before at this initial proton energy, as a baseline for comparison. All 230 MeV simulations were carried out under the *QGSP_BIC* physics list.

2.2. Most Likely Path

Given the coordinate system and the simulation framework described in Section 2.1, with the proton beam directed along the z direction, at any given depth along z a proton's path can be characterised by the two coordinates x and y and the two angles θ and ϕ relative to the z-axis. Proton scattering can be considered independent along the x and y axis and the MLP can be expressed independently for the two 2D parameter vectors $\mathbf{x} = (x, \theta)$ and $\mathbf{y} = (y, \phi)$.

Considering $\mathbf{x}$ for example, from Schulte et al. (2008) the MLP of protons in a homogeneous medium can be expressed, in a Gaussian approximation of the generalised Fermi-Eyeges theory of Multiple Coulomb Scattering (MCS), as

$$\mathbf{x}_{\mathrm{MLP}}(z) = (\Sigma_1^{-1} + R_1^T \Sigma_2^{-1} R_1)^{-1} (\Sigma_1^{-1} R_0 \, \mathbf{x}_{in} + R_1^T \Sigma_2^{-1} \, \mathbf{x}_{out}), \tag{1}$$

where $\mathbf{x}_{in}$ and $\mathbf{x}_{out}$ are the relevant entry and exit coordinates in the two 2D parameter vectors as mentioned above, R_0 and R_1 are the change of basis for small-angle rotation matrices

$$R_0 = \begin{pmatrix} 1 & z - z_{in} \\ 0 & 1 \end{pmatrix}, \quad R_1 = \begin{pmatrix} 1 & z_{out} - z \\ 0 & 1 \end{pmatrix}, \tag{2}$$

and Σ_1 and Σ_2 are covariance matrices

$$\Sigma_1 = \begin{pmatrix} \sigma^2_{t_1} & \sigma^2_{t_1\theta_1} \\ \sigma^2_{t_1\theta_1} & \sigma^2_{\theta_1} \end{pmatrix}, \quad \Sigma_2 = \begin{pmatrix} \sigma^2_{t_2} & \sigma^2_{t_2\theta_2} \\ \sigma^2_{t_2\theta_2} & \sigma^2_{\theta_2} \end{pmatrix}, \tag{3}$$

with components, called *scattering moments*, given for Σ_1 by the integrals

$$\sigma^2_{t_1} = E_0^2 \left(1 + 0.038 \ln \frac{z - z_{in}}{X_0}\right)^2 \int_{z_{in}}^{z} \frac{(z-u)^2}{\beta^2(u)p^2(u)} \frac{\mathrm{d}u}{X_0} \tag{4}$$

$$\sigma^2_{\theta_1} = E_0^2 \left(1 + 0.038 \ln \frac{z - z_{in}}{X_0}\right)^2 \int_{z_{in}}^{z} \frac{1}{\beta^2(u)p^2(u)} \frac{\mathrm{d}u}{X_0} \tag{5}$$

$$\sigma^2_{t_1\theta_1} = E_0^2 \left(1 + 0.038 \ln \frac{z - z_{in}}{X_0}\right)^2 \int_{z_{in}}^{z} \frac{(z-u)}{\beta^2(u)p^2(u)} \frac{\mathrm{d}u}{X_0}, \tag{6}$$

where u is the predicted proton path. The equivalent scattering moments for Σ_2 are found by replacing z_{in} with z and z with z_{out} in the equations above. $\mathbf{y}_{\mathrm{MLP}}(z)$ follows identically, with $\mathbf{x}_{in}$ and $\mathbf{x}_{out}$ replaced by $\mathbf{y}_{in}$ and $\mathbf{y}_{out}$ as necessary.

Assuming a homogeneous phantom composed of water, we use $X_0 = 36.1\,\mathrm{cm}$ for the radiation length of the material and $E_0 = 13.6$ MeV. The momentum velocity ratio $1/\beta^2(u)p^2(u)$ is approximated with a fifth-order polynomial following Schulte et al. (2008). This quantity is specific to the proton energy used; implementation for other energies requires its recalculation for accurate performance. For protons at 230 MeV this was calculated as outlined in Schulte et al. (2008). Monoenergetic protons initially at the required energy were incident on a simulated 20 cm deep water sample. The fifth-order polynomial was fitted to distribution of the mean value of $1/\beta^2(u)p^2(u)$ recorded at 5 mm intervals throughout.

2.3. Proton Path Neural Network

The Proton Path Neural Network (PPNN) is fully connected neural network based model designed to predict a proton trajectory in the form of a series of spacial points, as described in Section 2.1, using variables similar to those employed by MLP calculations. As with the MLP, trajectories along the x and y directions are reconstructed independently by separate instances of the same network. The input features of the network are quantities which can be recorded by a modern pCT scanning apparatus; $\Delta x = (x_{out} - x_{in})$ and $\Delta\theta = (\theta_{out} - \theta_{in})$ in the x direction and equivalently $\Delta y = (y_{out} - y_{in})$, $\Delta\phi = (\phi_{out} - \phi_{in})$ along y. This data is passed through 4 fully connected (or dense) layers of 24, 48, 96 and 199 nodes respectively. This type of layers are the most simple between the many developed in the context of Deep Neural Network: the output of the layer is a vector $\mathbf{y}$ obtained by

$$\mathbf{y} = \sigma(\mathbf{W} \cdot \mathbf{x} + \mathbf{b})$$

where $\mathbf{W}$ and $\mathbf{b}$ are called respectively weigths and bias and correspond to the parameters of the layer that will be fixed during training of the network; $\mathbf{x}$ is the input vector and σ is the activation function introducing non linear effects in the network

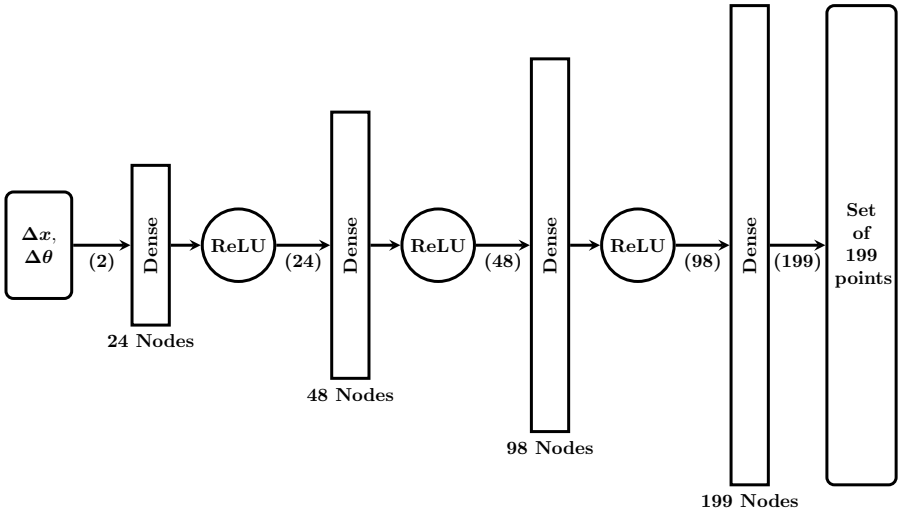


Figure 2. PPNN architecture. The Proton Path Neural Network PPNN consists of four fully connected layers with 24, 48, 96, 199 nodes and a Relu activation function after each of the first three layers. The current number of variables present at various points is additionally indicated in brackets.

behaviour. As activation function we employed the Rectified linear unit (ReLU) after each of the first 3 layers ($\mathrm{ReLU}\,(x) = \max(0, x)$). A representation of the network architecture is presented in Figure 2.

Training and validation of the network was performed using more than 1,600,000 trajectories (800,000 along each direction) generated as described in Section 2.1 using the $QGSP\_BIC$ physics list. 80% of the tracks are used for the training and the remaining 20% reserved for validation. Optimization of the network weights is performed using the `Adam` algorithm (Kingma & Ba (2014)) with a learning rate fixed at 10^{-5}. For the loss, the Mean Squared Error (MSE) is used,

$$\mathrm{MSE} = \frac{1}{M}\sum_{m}^{M}\frac{1}{N}\sum_{n}^{N}(u_{mn} - \hat{u}_{mn})^2, \tag{7}$$

where M is the number of samples, $N = 199$ is the number of points in each proton path, u again the predicted path and $\hat{u}$ the true trajectory. The (Square) Root of the Mean Squared Error (RMSE) is commonly adopted in literature evaluating the performance of the MLP reconstruction procedure. At a batch size of 32 samples per batch, one epoch (one cycle through the full training dataset) running on Tesla K80 GPU requires approximately 80 seconds on a Standard NC6 Microsoft Azure machine. For an introduction on Deep Neural Network we suggest looking at the free material available at https://d2l.ai/.

The loss history can be seen in Figure 3, in which after around 400 epochs the loss flattens both for the train and validation datasets with the ratio between the two histories almost constant; suggesting that the network is not overfitting to the examples present in the training dataset. Ultimately the model was trained for 1000 epochs.

In addition, a second instance of the PPNN was trained with a 230 MeV proton

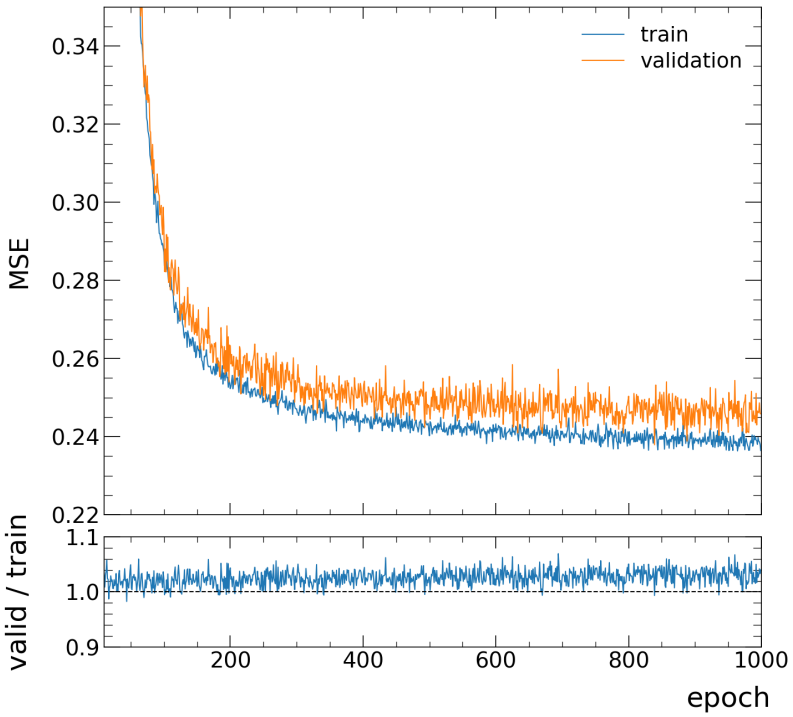


Figure 3. Loss history during network training at each epoch, for both the training and validation.

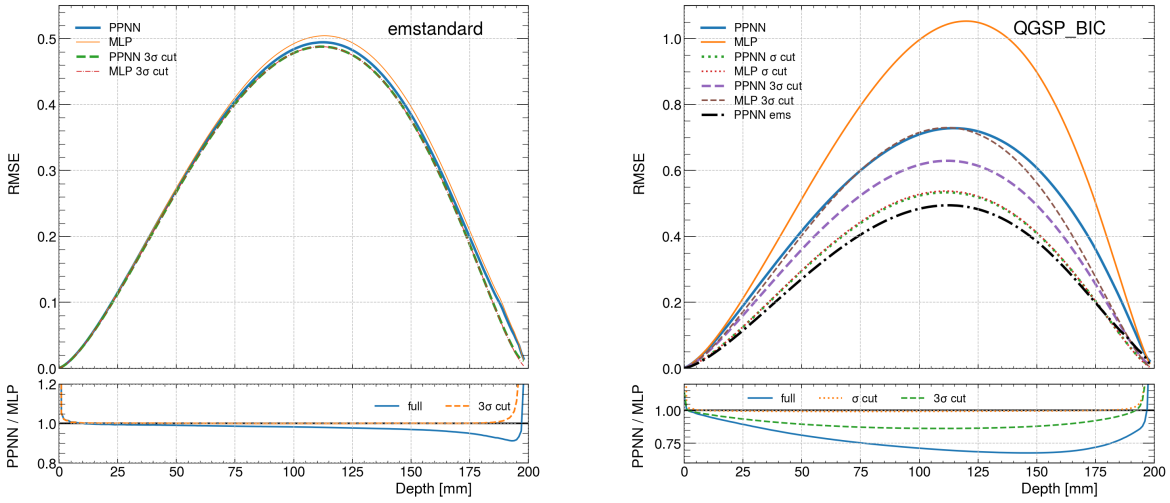
dataset in excess of $1,400,000$ events, using the same methodology and a pure water phantom. This instance is used when reconstructing datasets with protons at that energy.

3. Results

To principally test the performance of PPNN two entirely new datasets of 800,000 protons each were generated: the first with only electromagnetic interactions (*emstandard* physics list), the other with all the physical processes including nuclear interactions (*QGSP_BIC* physics list). These data sets are generated independently from that used during the PPNN training procedure to avoid any possible source of overfitting.

3.1. Root Mean Squared Error

Figure 4-(a) shows the RMSE for estimates of the paths using PPNN or MLP on the *emstandard* dataset. Even without the 3σ cuts suggested in Schulte et al. (2008) we can see that the difference between the two predictions is quite small. This difference disappears (the two lines corresponding to the MLP and PPNN case are barely

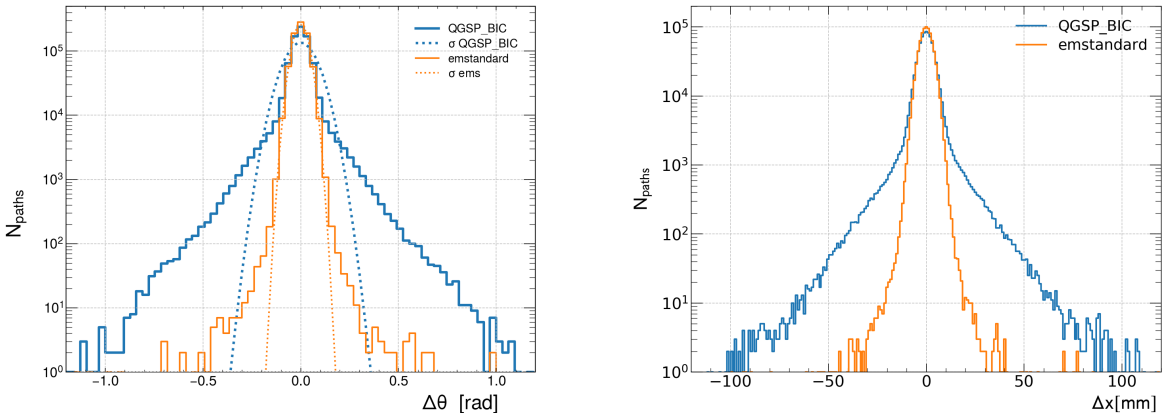


(**a**) (**b**)

Figure 4. Root Mean Squared Error obtained with MLP and PPNN using the (a) *emstandard* and (b) *QGSP_BIC* datasets. Solid lines are the performance on the full dataset while dotted and dashed incorporate 1σ, and 3σ cuts, performed on the energy and difference in the direction of travel angle between entering and exiting the phantom, respectively. The dashed-dotted line in (b) is the same solid PPNN result in (a) added here to have a clear picture of the increasing of the errors when including nuclear interactions.

distinguishable) upon applying said 3σ cut to the angles and energy; under which here only $\sim 1\%$ of the paths are omitted. This result clearly shows that the PPNN prediction is fully consistent with the MLP approach, indicating that the approximations inherent to the method are valid. This is crucial because anything different would represent a serious flaw in the PPNN reconstruction method.

Moreover, the difference in the PPNN prediction error with or without the cut is practically negligible, suggesting that our method can be applied to reconstruct trajectories where processes other than MCS are present. This is more evident in Figure 4-(b) where the RMSE is evaluated for the *QGSP_BIC* dataset. When nuclear interactions are included the error significantly increases, but to a far lesser extent for PPNN than for MLP. Only with a 1σ cut do the performances of the two methods become comparable. Unfortunately, such a huge cut entails the loss of $\sim 24\%$ of the tracks. Comparing the full interaction dataset result with that of the pure electromagnetic result, we see that with the typical 3σ cut applied to both cases the RMSE of PPNN is about 26% larger for the full interaction that for the pure MCS dataset. For the 2σ cut the discrepancy in performance decreases to around 20%, which corresponds to a fraction of discarded tracks of $\sim 8\%$ from the *QGSP_BIC* dataset.



(**a**) (**b**)

Figure 5. (a) Distribution of $\Delta\theta = (\theta_{out} - \theta_{in})$ angle for the two test datasets (solid lines) overlaid with the associated Gaussian using the σ values obtained from a fit of the *emstandard* data and the *QSPG_BIC* data (dotted lines). (b) Distribution of $\Delta x = (x_{out} - x_{in})$. In both plots it is evident that an exponential rather than a Gaussian decay provides a better fit with respect to the number of paths for the *QSPG_BIC* dataset.

3.2. Error as a function of deviations

To understand the origin of this difference in performance between the two methods, Figure 5-(a) illustrates the distribution of $\Delta\theta = (\theta_{out} - \theta_{in})$ for both *QSPG_BIC* and *emsstandard* datasets. The σ cut is applied assuming a Gaussian distribution of the signal, but from the figure a difference between the two distributions clearly emerges. For the full physics simulation the Gaussian approximation, as employed in the MLP, clearly fails to describe the distribution. While the cuts based on a Gaussian fit are acceptable in the *emstandard* case, they exhibit a large discrepancy with data when the full range of physics processes are included. In Figure 5-(b) we see a similar result for the distribution of lateral displacement $\Delta x = (x_{out} - x_{in})$, with the Gaussian shape of the *emstandard* distribution supplanted by an exponential decrease in the *QSPG_BIC* distribution.

Given this observation and having verified that the PPNN approach has the same perfomances as MLP in the context of pure electromagnetic interaction, where MLP is designed to work, from now on we will consider only the results obtained using the *QSPG_BIC* physics dataset as a much more realistic representation of clinical pCT scenario.

As the distributions of Figure 5 clearly show the limits of the MLP formulation, it is interesting therefore to consider how the error increases as a function of the two variables $\Delta\theta$ and Δx. This is presented in Figure 6. Here the proton paths are collected into bins of 0.1 rad and 1 mm for $\Delta\theta$ and Δx respectively, with the RMSE computed in the corresponding direction. The figure compares the error (right axis) and the number of trajectories (left axis) to show the differences in performance. Note the logarithmic

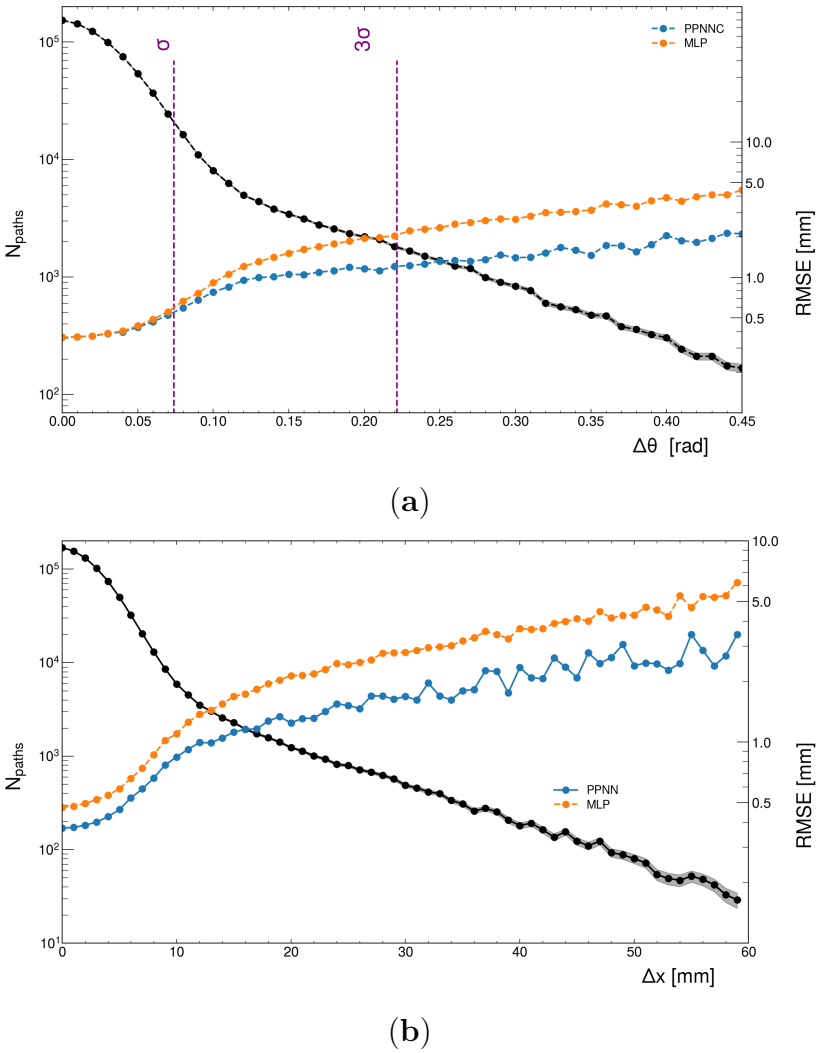


Figure 6. (a) RMSE (right vertical axis, coloured lines) and number of paths (left vertical axis, black lines) as a function of $\Delta\theta$ for PPNN and MLP evaluated on the *QSPG_BIC* dataset. The shaded black area represent the statistical error. Vertical lines refer to the position of the 1 and 3 σ cut. (b) Same as (a) but as a function of Δx. The difference in performance between the two methods emerges immediately.

scale on both right and left y axis. From Figure 6-(a) we see that, as expected from the RMSE plot, the two lines for PPNN and MLP begin to separate at around 1σ cut at $\Delta\theta \simeq 0.075$ rad. For 35% of the tracks $\Delta\theta$ is larger than 0.075, implying that the PPNN method improves on the MLP reconstruction for an important fraction of proton paths. Notice that the same analysis must be done for the ϕ angle which would remove an analogous number of paths, resulting in a final cut of almost 50% of the tracks. Figure 6-(b) shows the reconstructed paths distribution broken down in term of final displacement, Δx. Again the performance of PPNN is consistently better across the full span of the plot, with trajectories at large angle deviations resolved with improved precision.

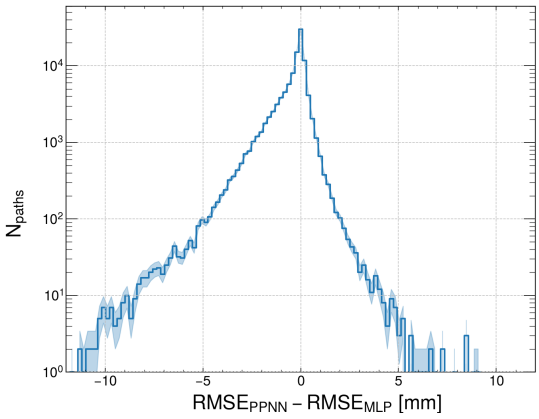


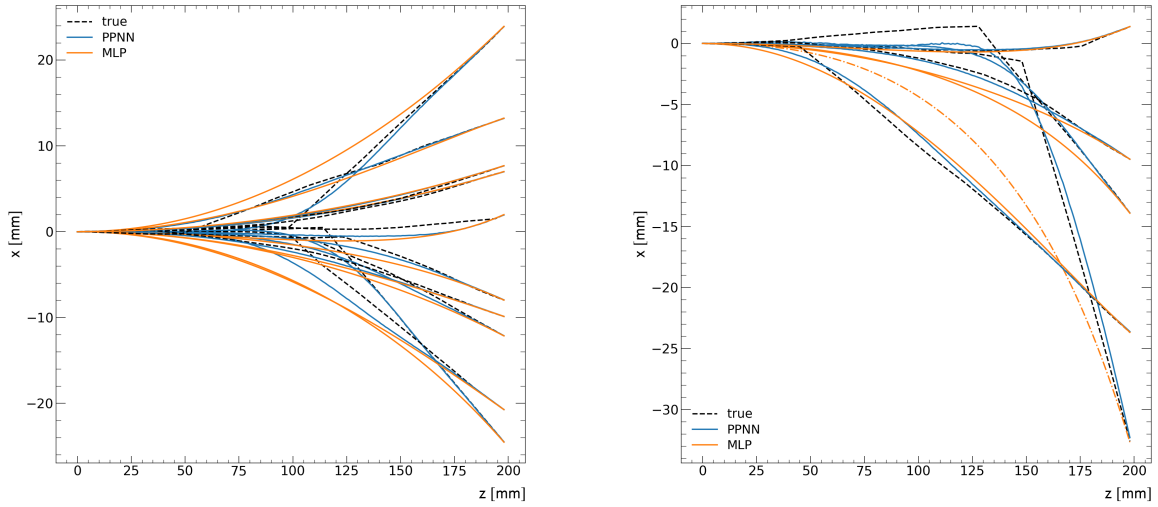
Figure 7. Distribution of the difference between the RMSE of PPNN and MLP for the *QSPG_BIC* dataset. The shaded area correspond to the statistical error.

3.3. Different trajectories for different errors

To gain an insight into the tracks with the largest difference in reconstruction performance, let us begin by considering only tracks outside the 1σ cut in θ. In Figure 7 we present the distributions of the difference between the RMSE for PPNN and MLP for tracks outside the aforementioned cut. Negative values of the difference correspond to tracks in which PPNN had the smallest error, while the positive side of the axis corresponds to the inverse. In the first instance we can see that the profile is exponential, while in the second the decay is noticeably faster; confirming that at large deviations of the angle θ, PPNN shows a notably superior performance.

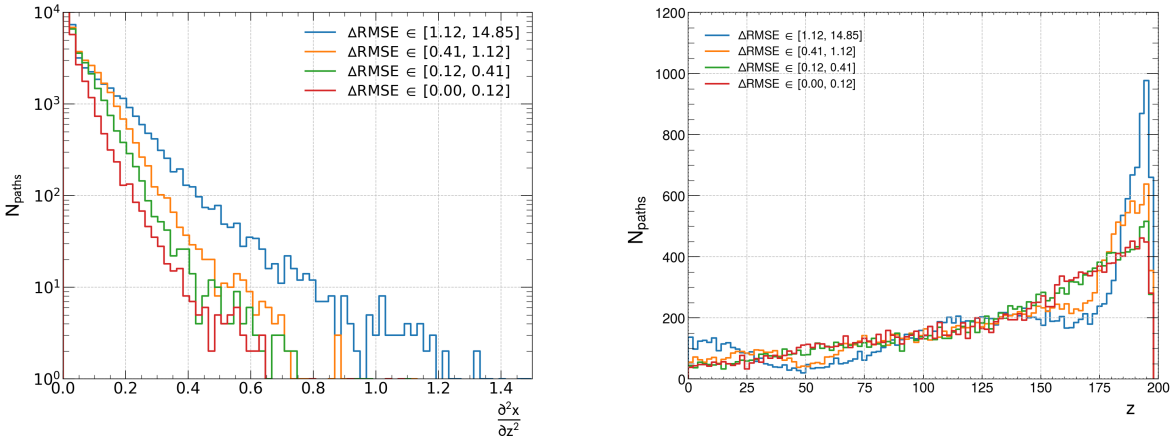
Focusing in on only the behaviour when PPNN outperforms MLP, let us consider only the set of events on the negative side of histogram. Dividing into 10 quantiles split by $\Delta RMSE$, in Figure 8-(a) we illustrate a selection of randomly chosen tracks, one from each quantile. As expected, for larger deviations from straight paths PPNN can better follow the simulated curve in the majority of such cases, growing more notable for larger $\Delta RMSE$. For Figure 8-(b) the same dataset is divided into quartiles, with the last bin, containing tracks with the largest error difference, further divided into two subgroups. As with Figure 8-(a) we chose a random track from each of the five groups. Both figures further support that PPNN improved performance is due at-least in part to a better capability to reproduce the particle path in the presence of nuclear interaction, which causes greater changes in the direction of the track.

To further analyse this characteristic, Figure 9-(a) shows the distribution of the second derivative of the x component of the tracks, with respect to the z direction, again for track in which PPNN outperforms MLP, broken down into quartiles. Large values of this quantity are connected with significant direction change, such as those observed in Figure 8. The four lines correspond to the four quartiles of the blue histogram in Figure 7, as introduced in Figure 8-(b). Where PPNN exhibits the better performance, we see that the difference between the tracks reconstructed with PPNN and MLP grows



(**a**) (**b**)

Figure 8. Examples of tracks for which the PPNN outperform MLP. (a) Tracks are selected at random from inside each of 10 quantiles, using the data of Figure 7. (b) Same as (a), but in which tracks are extracted from quartile groups; with the last quartile, which corresponds to tracks with the largest discrepancies between the two methods, divided into two.

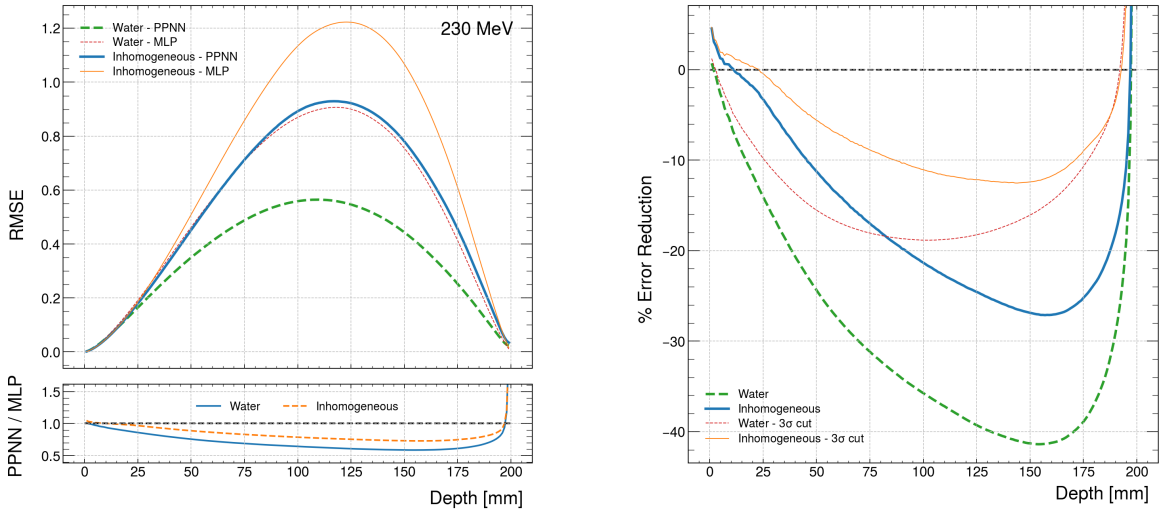


(**a**) (**b**)

Figure 9. (a) Distributions of the second derivative of the tracks in the x direction with respect to the z coordinate. Lines indicate the four quartiles of the distribution of $\Delta RMSE < 0$. (b) Distribution of the position along the z axis for the maximum of the second derivative for each path.

with increasing values of $\frac{\partial^2 x}{\partial z^2}$: the more a trajectory differs from pure MCS scattering, the more the PPNN improves over MLP.

Figure 9-(b) shows the distribution of $\max(\frac{\partial^2 x}{\partial z^2})$ as a function of z. The distribution for the last quartile, corresponding to the largest discrepancies between the two methods, has a notably different behaviour compared to the other three lines. It exhibits



(**a**) (**b**)

Figure 10. (a) Root Mean Squared Error obtained with MLP and PPNN on a water and an inhomogeneous slab phantom irradiated at 230 MeV. (b) Percentage reduction in RMSE with respect to depth by PPNN over MLP. All studies were performed under the *QGSP_BIC* physics environment.

significantly more events occurring at small and large z values. An example of these events can be seen in Figure 8-(b) where we have a strong deflection at $z \approx 190$ mm. We see that MLP struggles to reproduce this event while the neural network can provide a superior result.

3.4. Inhomogeneous slab phantom

In this section, we present the results obtained using PPNN in the reconstruction of proton trajectory traversing the slab phantom described in 2.1 and represented schematically in Figure 1-(c). Due to the inhomogeneous phantom's increased stopping power, a proton energy of 230 MeV was chosen to ensure a significant fraction of simulated events traversed the full phantom depth. This ensured datasets in excess of $1,400,000$ trajectories ($700,000$ along each direction) for 10^6 simulated particles. Both PPNN and MLP methods were re-trained (re-calibrated for MLP) to the new energy scheme, as described in Sections 2.3 and 2.2. For this purpose, we consider a simulation with 230 MeV protons through a water phantom analogous to the one used in the 200 MeV case.

The RMSE error for both phantoms, using either PPNN or MLP, is shown in Figure 10-(a). This compares the water and inhomogeneous systems, without cuts and using the *QGSP_BIC* physics environment. For the water phantom both PPNN and MLP behave similarly to the corresponding 200 MeV case. This is an important check that the higher energy implementations of the two methods are functioning correctly.

Focusing on the reconstruction error for the inhomogeneous case, we similarly

observe that with PPNN the error is consistently reduced. Interestingly the error on the new phantom using PPNN is comparable with that obtained with MLP in the pure water simulation.

The improvement obtained with PPNN is more pronounced when examining the percentage reduction of RMSE by PPNN over MLP, as shown in Figure 10-(b). A reduction in the error of the order of 25% can be seen around 150mm, while on average the improvement is in excess of 10% over MLP across a significant portion of the depth. Introducing the familiar 3σ cuts decreases the error reduction in both the water and inhomogeneous cases, along with the difference in improvement between them.

3.5. Execution time comparison

For this comparison of the execution time of the two algorithms, the highly optimized version of MLP presented in McAllister (2009) is used, in which 90% of the MLP is precalculated and the number of operation required is minimized. We ported the code in python using the vectorization capabilities of the NumPy (numpy.org) library to parallelize the execution on the number of protons. PPNN is written in python using the PyTorch (pytorch.org) framework.

Both codes were executed on the CPU of a Standard NC6 Microsoft Azure machine. Running the two algorithms on all the 1,600,000 trajectories of the test dataset in unique batch combinations and repeating the procedure 10 times we obtain an almost constant execution time of 0.47 ± 0.01 sec for PPNN and 7.11 ± 0.08 sec for MLP. Within the validity of this test, the PPNN method is sixteen times faster than the optimized MLP.

4. Discussion

Although MLP represents a powerful method of estimating proton path in pCT applications, it suffers from different limitations. The approach is designed specifically to account only for effects on the proton path connected with MCS and energy loss. This is reflected by the strategy of discarding protons trajectories with large deviation from straight paths to reduce the error. Moreover, simulation in a realistic scenario of high fluence (hundreds of millions of protons) and small spacing for the MLP (fraction of millimetre) can require more than one hour; time mostly spent reconstructing the proton (paths Khellaf et al. (2020)).

In the interests of alleviating these two problems we propose an alternative method, based on Deep Learning Neural Network, to estimate the proton trajectory for pCT. The results presented in the previous section suggests that within the PPNN approach, these two problems can be relieved to some degree. Figure 4 and Figure 7 show that using PPNN a good approximation of the path can be obtained for a much larger number of protons than using MLP. This is important because in principle fewer protons are needed to reach the same reconstruction quality, lowering both the dose and the computation time. Consolidating this claim is one of the aims of our future developments.

The ability of the network to reconstruct tracks outside the validity of the MLP approach is intrinsically tied to the nature of deep learning. Neural networks learn "blindly" from examples; parsing though the training dataset, by means of the back-propagation procedure for the minimisation of the loss function, the network adapts its weights to the characteristics of the events it experiences, including those that show large $\Delta\theta$ and/or Δx. While such underlying processes maybe challenging to formulate into mathematical models, there are sufficient patterns for the network to refine its prediction processes. Without an assumed structure to reproduce, it is not bound to solely replicating the form of a given physical model. A tentative explanation of what the network learns may be inferred from Figure 9 and the analysis of the second derivative of x w.r.t. z. The network displays significant improvement over MLP where the second derivative is large, especially near the end of the trajectories.

The study of inhomogeneous systems is only started here, and it certainly warrants a much more in-depth investigation into more realistic configurations of the phantom. The phantom considered is certainly extreme; large volumes of a high-density material such as those in the slab phantom will rarely be encountered in clinical practice, and in this sense we do not expect the gain to be so large in a realistic situation. Nevertheless, it is encouraging that notably better results are obtained with PPNN with respect to MLP, with a reductions of the RMSE of the order of 20%. This is a more significant improvement compared to the work presented in (Brooke & Penfold 2020) with a similar phantom, where the maximum enhancement is about 5% for simulation with the same beam energy.

Regarding execution speed, it is true that the time spent for reconstruction is only one of the various aspects for evaluating a pCT system for clinical routine. Moreover, our work is relevant only in the context of reconstruction methods based on the evaluation of the proton path. Nevertheless, because these methods are seen as the most promising for applicability in the clinical context and the MLP execution speed is by order of magnitudes the slowest part of the algorithm (Khellaf et al. (2020)), the substantial improvement shown by PPNN compared with the optimized MLP can be regarded as an important feature.

5. Conclusions

MLP is the principal method adopted in pCT for the reconstruction of single proton paths through the body. In this paper we have demonstrated that using Deep Learning Neural Network it is possible to recreate the same performance of MLP in the regime in which MLP is applicable and achieve a better performance outside its region of validity. Using PPNN would also permit discarding fewer protons in the pCT procedure. Moreover, an execution time test of the two algorithms indicates that PPNN can be substantially faster in performing the reconstruction. In the future we plan to move forward in the development of the method towards a full reconstruction procedure applicable to more realistic phantoms.

6. Acknowledgments

We would like to acknowledge Simon Rit for the useful discussion and clarification of the MLP method in the early phase of the experiments and development of the PPNN method.

Agostinelli, S., Allison, J., Amako, K. a., Apostolakis, J., Araujo, H., Arce, P., Asai, M., Axen, D., Banerjee, S., Barrand, G. . et al. (2003). Geant4—a simulation toolkit, *Nuclear instruments and methods in physics research section A: Accelerators, Spectrometers, Detectors and Associated Equipment* **506**(3): 250–303.

Beaton, L., Bandula, S., Gaze, M. N. & Sharma, R. A. (2019). How rapid advances in imaging are defining the future of precision radiation oncology, *British journal of cancer* **120**(8): 779–790.

Berger, M. J., Inokuti, M., Andersen, H. H., Bichsel, H., Powers, D., Seltzer, S. . M., Thwaites, D. . & Watt, D. E. (2016). Report 49, *Journal of the International Commission on Radiation Units and Measurements* **os25**(2): NP–NP.
URL: `https://doi.org/10.1093/jicru/os25.2.Report49`

Bovik, A. C. (2009). *The essential guide to image processing*, Academic Press.

Brooke, M. D. & Penfold, S. N. (2020). An inhomogeneous most likely path formalism for proton computed tomography, *Physica Medica* **70**: 184–195.

Collins-Fekete, C.-A., Bär, E., Volz, L., Bouchard, H., Beaulieu, L. & Seco, J. (2017). Extension of the fermi–eyges most-likely path in heterogeneous medium with prior knowledge information, *Physics in Medicine & Biology* **62**(24): 9207.

Collins-Fekete, C.-A. C., Doolan, P., Dias, M. F., Beaulieu, L. & Seco, J. (2015). Developing a phenomenological model of the proton trajectory within a heterogeneous medium required for proton imaging, *Physics in Medicine & Biology* **60**(13): 5071.

Collins-Fekete, C.-A., Volz, L., Portillo, S. K., Beaulieu, L. & Seco, J. (2017). A theoretical framework to predict the most likely ion path in particle imaging, *Physics in Medicine & Biology* **62**(5): 1777.

Doolan, P., Testa, M., Sharp, G., Bentefour, E., Royle, G. & Lu, H. (2015). Patient-specific stopping power calibration for proton therapy planning based on single-detector proton radiography, *Physics in Medicine & Biology* **60**(5): 1901.

Fippel, M., Soukup, M. et al. (2004). A monte carlo dose calculation algorithm for proton therapy, *Medical Physics* **31**.

Foote, R. L., Stafford, S. L., Petersen, I. A., Pulido, J. S., Clarke, M. J., Schild, S. E., Garces, Y. I., Olivier, K. R., Miller, R. C., Haddock, M. G. et al. (2012). The clinical case for proton beam therapy, *Radiation Oncology* **7**(1): 1–10.

Hu, M., Jiang, L., Cui, X., Zhang, J. & Yu, J. (2018). Proton beam therapy for cancer in the era of precision medicine, *Journal of hematology & oncology* **11**(1): 136.

Jan, S., Benoit, D., Becheva, E., Carlier, T., Cassol, F., Descourt, P., Frisson, T., Grevillot, L., Guigues, L., Maigne, L. et al. (2011). Gate v6: a major enhancement of the gate simulation platform enabling modelling of ct and radiotherapy, *Physics in Medicine & Biology* **56**(4): 881.

Johnson, R. P. (2017). Review of medical radiography and tomography with proton beams, *Reports on Progress in Physics* **81**(1): 016701.

Khellaf, F., Krah, N., Létang, J. M., Collins-Fekete, C.-A. & Rit, S. (2020). A comparison of direct reconstruction algorithms in proton computed tomography, *Physics in Medicine & Biology* **65**(10): 105010.

Kingma, D. P. & Ba, J. (2014). Adam: A method for stochastic optimization, *3rd International Conference on Learning Representations (ICLR) 2015*.

Krah, N., Létang, J.-M. & Rit, S. (2019). Polynomial modelling of proton trajectories in homogeneous media for fast most likely path estimation and trajectory simulation, *Physics in Medicine &*

Biology **64**(19): 195014.

Li, T., Liang, Z., Singanallur, J. V., Satogata, T. J., Williams, D. C. & Schulte, R. W. (2006). Reconstruction for proton computed tomography by tracing proton trajectories: A monte carlo study, *Medical physics* **33**(3): 699–706.

McAllister, S. A. (2009). *Efficient proton computed tomography image reconstruction using general purpose graphics processing units*, PhD thesis, California State University, San Bernardino.

McAllister, S., Schubert, K., Schulte, R. & Penfold, S. (2009). General purpose graphics processing unit speedup of integral relative electron density calculation for proton computed tomography, *2009 IEEE Nuclear Science Symposium Conference Record (NSS/MIC)*, IEEE, pp. 4085–4087.

Schneider, U. & Pedroni, E. (1994). Multiple coulomb scattering and spatial resolution in proton radiography, *Medical physics* **21**(11): 1657–1663.

Schulte, R., Penfold, S., Tafas, J. & Schubert, K. (2008). A maximum likelihood proton path formalism for application in proton computed tomography, *Medical physics* **35**(11): 4849–4856.

Tian, X., Liu, K., Hou, Y., Cheng, J. & Zhang, J. (2018). The evolution of proton beam therapy: Current and future status, *Molecular and clinical oncology* **8**(1): 15–21.

Williams, D. (2004). The most likely path of an energetic charged particle through a uniform medium, *Physics in Medicine & Biology* **49**(13): 2899.