

# Developing a Hybrid Machine Learning Model for VELO Upgrade Track Reconstruction



A thesis submitted in accordance with the requirements of the  
University of Liverpool for the degree of Doctor of Philosophy

by

**Phillip Marshall**

Supervisors: Themis Bowcock, Simon Maskell

Oliver Lodge Laboratory

Department of Physics

University of Liverpool

March 2022



## **Abstract**

High energy physics experiments have been processing large quantities of data for decades. The growth of high volume data collection in industry has led to significant technical development in big data and machine learning due to their enormous commercial potential. This knowledge can be harnessed by the next generation of high energy physics experiments, which require sophisticated real-time, high throughput data processing pipelines to exploit the physics potential of huge datasets. The ambitious goals of the upgrade LHCb experiment require a shift to reconstructing all particle collisions in real-time. The corresponding increase in data rate will test the capability of conventional processing algorithms, such as the methods of track reconstruction. Machine learning methods are potential candidates for future tracking algorithms. They can learn from data without prior physics knowledge and are capable of incredible speed through parallelisation on specialist hardware such as FPGAs.

In this thesis I focus on the VELO detector, which surrounds the proton collision point at the LHCb experiment. Tracks reconstructed in the VELO are a vital component in reducing data throughput by the rejection of unwanted collision events. After experimenting with machine learning methods and toy tracking models, I built on work to develop a hybrid VELO tracking algorithm incorporating a neural network to join pairs of hits into track seeds. The algorithm matches baseline performance target, and is more robust to small random detector misalignment than the conventional tracking algorithm.

## **Acknowledgements**

I would like to thank those whose knowledge, guidance and support has made this thesis possible. Thank you to my supervisors Themis Bowcock and Simon Maskell, for their invaluable mentorship. I am extremely grateful for the expertise of Kurt Rinnert, David Hutchcroft, Karlis Dreimanis of the University of Liverpool, as well as Marco Cristoforretti of FBK and Jan Buytaert of CERN. For accepting with enthusiasm my placement request, David Smith from OnTrac Ltd. I am eternally thankful for my wellbeing advisor Ashley, who has listened through high and low. I extend thanks to my peers and friends; especially Tom and Lauren, James and Julia, and everyone from the LIVDAT CDT. Also to everyone who made moving to new places so enjoyable; Caleb and Christian, and Francesco who taught me to ski. To the Geneva Wizards and UFO Trento Ultimate Frisbee teams for making me feel so welcome far from home. I must finally thank my friends and family who have been with me the whole way; especially Henry, Poppy, Wojciech and Maggie.

### **Declaration**

This thesis is the result of my own work, except where a specific reference to the work of others is made. This thesis has not been submitted for any other qualification to this or any other university.



---

# Contents

---

<b>Acronyms</b>	<b>vi</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Track Reconstruction . . . . .	2
1.2 Triggering . . . . .	3
1.3 Data Challenges . . . . .	3
1.4 Machine Learning and Big Data . . . . .	5
<b>2 The LHCb Experiment</b>	<b>7</b>
2.1 The Large Hadron Collider . . . . .	7
2.1.1 CERN Accelerator Complex . . . . .	8
2.1.2 Future of the LHC . . . . .	10
2.1.3 LHC Computing Program . . . . .	10
2.2 Introduction to LHCb and its Upgrade . . . . .	11
2.3 Physics at the LHC and LHCb . . . . .	13
2.4 Tracking System . . . . .	15
2.4.1 Vertex Locator . . . . .	15
2.4.2 Upstream Tracker . . . . .	16
2.4.3 Magnet . . . . .	17
2.4.4 SciFi . . . . .	18
2.5 Particle Identification . . . . .	19
2.5.1 Ring Imaging Cherenkov Detectors . . . . .	19
2.5.2 Calorimeters . . . . .	20

2.5.3	Muon System . . . . .	21
2.6	Online system . . . . .	22
2.6.1	Timing and Fast Control . . . . .	22
2.6.2	Event Builder . . . . .	23
2.6.3	Event Filter Farm . . . . .	23
2.7	Data Storage . . . . .	24
2.8	LHCb Performance . . . . .	24
2.8.1	Detector Performance and Physics Discoveries . . . . .	24
<b>3</b>	<b>VELO Detector</b>	<b>27</b>
3.1	VELO Upgrade Design . . . . .	27
3.1.1	Modules . . . . .	29
3.1.2	VELOPix readout . . . . .	30
3.1.3	Electronics and DAQ . . . . .	31
3.2	VELO Upgrade Requirements . . . . .	32
3.2.1	Physics . . . . .	32
3.2.2	Radiation . . . . .	32
3.2.3	Material . . . . .	33
3.2.4	Cooling . . . . .	34
3.3	VELO front-end hybrid visual inspection . . . . .	34
<b>4</b>	<b>LHCb Software and Computing</b>	<b>37</b>
4.1	The LHCb Trigger . . . . .	37
4.1.1	Physics Requirements . . . . .	37
4.1.2	Inclusive and Exclusive Selections . . . . .	38
4.1.3	Run 1 and 2 Trigger . . . . .	39
4.1.4	Turbo Stream . . . . .	39
4.1.5	Run 3 Trigger . . . . .	41
4.1.6	LHCb Trigger with GPUs . . . . .	43
4.2	Track Reconstruction . . . . .	45
4.2.1	Track Types . . . . .	45
4.2.2	Clustering . . . . .	46
4.2.3	Conventional VELO Pattern Recognition . . . . .	46
4.2.4	Vertex Reconstruction . . . . .	47
4.2.5	UT and Forward Tracking . . . . .	48
4.2.6	Detector Alignment and Calibration . . . . .	49
<b>5</b>	<b>Machine Learning</b>	<b>51</b>

5.1	History of Machine Learning . . . . .	52
5.2	Why use Machine Learning in High Energy Physics? . . . . .	53
5.3	Types of Machine Learning Algorithm . . . . .	55
5.3.1	Supervised Learning . . . . .	55
5.3.2	Unsupervised Learning . . . . .	56
5.4	Applications of Machine Learning . . . . .	57
5.4.1	Current use in High Energy Physics . . . . .	57
5.4.2	Future Applications in High Energy Physics . . . . .	59
5.5	Why Choose a Neural Network for VELO Tracking? . . . . .	60
5.6	Neural Networks and How They Work . . . . .	61
5.6.1	Mathematics of Neural Networks . . . . .	62
5.7	Six Month Industry Placement at OnTrac . . . . .	66
<b>6</b>	<b>A New Hybrid Model for Tracking</b>	<b>69</b>
6.1	The Tracking Problem . . . . .	70
6.2	Velo Tracking Requirements . . . . .	70
6.2.1	Efficiency . . . . .	70
6.2.2	Timing . . . . .	71
6.3	Current Challenges Facing Tracking Systems . . . . .	71
6.4	Initial Machine Learning Tracking Investigations . . . . .	72
6.4.1	Joining Closest Points and Building Tracks . . . . .	73
6.4.2	Predicting Hits in Adjacent Modules . . . . .	75
6.4.3	Unsupervised Learning . . . . .	77
6.5	Hybrid Tracking Method and Results . . . . .	79
6.5.1	Initial Development of the Hybrid Model . . . . .	79
6.5.2	Overview of My Revised Hybrid Model . . . . .	80
6.5.3	Find Doublet Candidates with a Neural Network . . . . .	81
6.5.4	Data Sample and Network Training Details . . . . .	85
6.5.5	Track Reconstruction . . . . .	86
6.5.6	Clone Reduction . . . . .	86
6.5.7	Efficiency Criteria . . . . .	87
6.5.8	Reconstruction Efficiency . . . . .	88
6.5.9	Module Misalignment Study . . . . .	89
6.5.10	Closest target hit vs neural network . . . . .	91
6.5.11	Inference Time Investigation . . . . .	91
6.6	Possible Improvements . . . . .	93
6.6.1	Graph Neural Networks . . . . .	93

---

6.6.2	Connecting Doublets . . . . .	94
6.6.3	Retraining the Model . . . . .	94
6.7	Summary . . . . .	96
<b>7</b>	<b>Expanding the Model</b>	<b>99</b>
7.1	Shortcomings of the Hybrid Model . . . . .	99
7.2	Possible Improvements . . . . .	100
7.2.1	End to End Machine Learning . . . . .	100
7.2.2	Multiple Neural Networks . . . . .	101
7.3	Hardware Acceleration . . . . .	101
	<b>Conclusion</b>	<b>103</b>
	<b>Bibliography</b>	<b>105</b>
	<b>List of Figures</b>	<b>119</b>
	<b>List of Tables</b>	<b>121</b>
	<b>Appendix</b>	<b>123</b>

---

# Acronyms

---

- ALICE** A Large Ion Collider Experiment. 3, 8
- ASIC** Application Specific Integrated Circuit. 29–31, 34, 60, 100, 102, 120
- ATLAS** A Toroidal LHC ApparatuS. 3, 4, 8
- CERN** European Organisation for Nuclear Research. 1, 2, 7, 8, 10, 11, 27, 34, 119
- CMS** Compact Muon Solenoid. 3, 4, 8, 58, 59, 120
- CNN** Convolutional Neural Network. 59, 75, 76, 96
- CP** Charge-Parity. 14, 15, 25, 37
- CPU** Central Processing Unit. 4, 43, 44, 46, 69, 91, 92, 96, 101, 102
- DBSCAN** Density-Based Spatial Clustering of Applications with Noise. 77
- ECAL** Electromagnetic Calorimeter. 12, 20, 21
- EFF** Event Filter Farm. 22–24, 42, 44
- FPGA** Field-Programmable Gate Array. i, 23, 46, 60, 84, 100, 102
- GNN** Graph Neural Network. 93
- GPU** Graphics Processing Unit. 1, 3, 6, 23, 42–44, 69, 84, 91, 92, 96, 102

- HCAL** Hadronic Calorimeter. 12, 21
- HL-LHC** High Luminosity LHC. 3–6, 10, 71, 104
- HLT** High Level Trigger. 3, 12, 23, 24, 27, 39, 41–44, 46, 70, 71
- IP** Impact Parameter. 32, 39
- LHC** Large Hadron Collider. 2, 3, 7–13, 15, 16, 23, 27, 28, 37, 39, 41, 103
- LHCb** Large Hadron Collider Beauty. i, 1–5, 7–9, 11, 12, 14–19, 24, 25, 27, 32, 37–41, 43, 45, 47, 58, 59, 69–71, 81, 87, 88, 96, 100–104, 119, 120
- LINAC** Linear Accelerator. 8, 10
- LIV.DAT CDT** Liverpool Big Data Science Centre for Doctoral Training. 66, 103
- PID** Particle Identification. 15, 19
- PS** Proton Synchrotron. 10
- PSB** Proton Synchrotron Booster. 10
- PV** Primary Vertex. 25, 47, 48
- ReLU** Rectified Linear Unit. 61, 63, 65, 83, 84
- RICH** Ring Imaging Cherenkov. 11, 19
- RTA** Real Time Analysis. 6
- SciFi** Scintillating Fibre Tracker. 11, 12, 15, 17, 18, 20, 37, 45, 48, 49, 70
- SM** Standard Model. 8
- SPS** Super Proton Synchrotron. 10
- SWP** Safe Work Planning. 67, 68
- TFC** Timing and Fast Control. 22
- TPC** Time Projection Chamber. 75, 96
- UT** Upstream Tracker. 11, 12, 15–17, 19, 21, 37, 45, 48, 49, 70

**VELO** Vertex Locator. i, 1, 11, 12, 15–17, 19, 27–32, 34, 37, 39, 41, 42, 44–46, 48, 49, 57, 60, 69–73, 75–77, 79, 80, 85–88, 93, 96, 101–103, 119, 120, 122

**WLCG** Worldwide LHC Computing Grid. 11, 24

# CHAPTER 1

---

## Introduction

---

**H**IGH energy physics experiments collect and process vast quantities of data at ever increasing rates. Soon, the current methods of analysis will not keep up with data rates from new experiments, such as future upgrades to the LHCb experiment at CERN. One of the most time consuming steps in the data processing chain is reconstructing the trajectory of particles from raw detector data, a process known as tracking. Any reduction in time taken to perform this step can lead to large savings in computation, thereby increasing the quantity of data that can be processed, and increasing the potential to discover new physics. The aim of this thesis is to investigate the potential of a machine learning based tracking algorithm for the LHCb Vertex Locator (VELO). Machine learning models are a candidate for future tracking algorithms because they can be simple to tune, having fewer parameters than conventional algorithms, and because they can take advantage of hardware acceleration, using GPUs for example, for very fast inference.

In this chapter I will outline the main themes of this thesis. Firstly an introduction to the concepts and history of track reconstruction in Sec. 1.1, and experiment triggers in Sec. 1.2. This is followed by an overview of the data challenges that will be faced by the next generation of high energy physics experiments in Sec. 1.3. In Sec. 1.4 I will introduce machine learning and big data, and how ideas from these fields can be used in high energy physics.



## 1.1 Track Reconstruction

In order to understand high energy particle interactions, like those produced at the Large Hadron Collider (LHC), it is vitally important to know the position and momenta of particles produced in collision events, these are known as kinematic parameters [1]. In the early days of high energy physics, this could only be done by the manual inspection of bubble chamber images seen in Fig. 1.1; however due to the complexity and rate of collisions in modern experiments, computers are used instead to perform the tracking process automatically in the blink of an eye. Even so, tracking can make up a large proportion of an experiments processing time, for example, almost 50% of the first stage of data processing at LHCb [2]. Tracking is conventionally split into two components, track finding and track fitting. Track finding identifies possible track candidates, then a fit is performed to identify the best track candidates and estimate some kinematic parameters. For curved tracks of particles moving in magnetic fields, momentum can be estimated from how sharp the track curves. The path of particle tracks can be traced backwards towards their origin to determine the position of collisions and decays, the primary and secondary vertices respectively. The latter being especially important for heavy flavour physics analysis. Conventional tracking methods generally use variants of a Kalman filter, which are algorithms that recursively build tracks from detector measurements [3]. Track fitting usually involves calculating a  $\chi^2$  for each track candidate, which determines whether it is compatible with a single particle track. Track fitting reduces the number of ghost/fake tracks, tracks built of hits from different particles. In many cases, to save time, it is possible to do simple fit in real-time, then a full fit later when time allows.

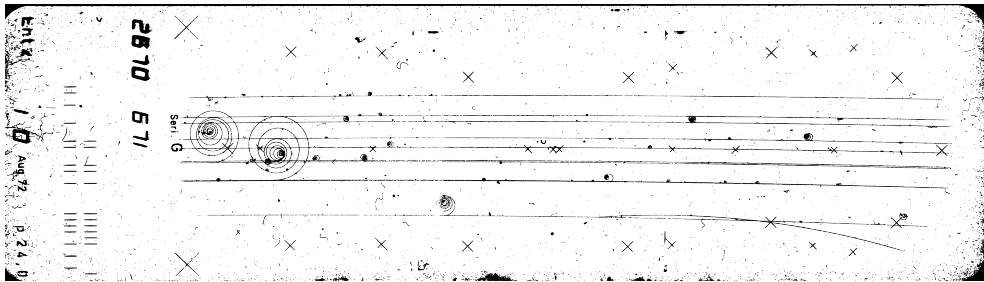


Figure 1.1: A film photo of tracks in a bubble chamber exposed to 24 GeV/c protons from the CERN Proton Synchrotron in 1972 [4]. The direction of spiral tracks indicates the charge of a particle and the tighter the curvature of a path, the lower the particle momentum.

## 1.2 Triggering

In general, only a small number of collisions at high energy physics experiments produce particles and processes interesting to physicist. A trigger is a system that determines which collision events contain these interesting physics processes, and which do not [5]. As collision rates in experiments increase, and as physicists search for rarer and rarer processes, the need for faster and more effective triggers becomes more important. In the days of bubble chamber experiments the trigger system consisted of “trained personnel” [4], who would inspect each individual image for potentially interesting physics, and discard events that did not. These people, known as “scanners” would also note the features of each image, such as which particle types were present. The only use of computers was for estimating parameters from images such as particle momentum. This is an example of an *offline* trigger, meaning the images are analysed after the fact and do not influence the running of the experiment. 50 years later for the LHCb upgrade, the new High Level Trigger (HLT) is purely software based and will run on GPUs. It will process up to 40 million events per second in real-time, ingesting terabytes of data per second from multiple tracking detectors and calorimeters. The LHCb HLT is an *online* trigger, it runs in real-time with the experiment and decides which events need further processing and saving to storage. An online trigger also allows slower parts of an experiment to keep up, by requiring them to only process a subset of events. In general, experiments use an online trigger to select data efficiently in the moment, and then an offline system to further analyse events when computing resources are available.

## 1.3 Data Challenges

As experiments process more complex collisions at an ever increasing rate, data analysis requirements will continue to grow. For example, LHCb has collected approximately one exabyte ( $1 \times 10^{18}$  bytes) of data during Runs 1 & 2 of the LHC between 2011 and 2018 [7]. The vast quantities of data that future particle physics experiments will collect and process will require significant updates to software and hardware to keep pace. In Run 3 of the LHC, starting in 2022, LHCb will process 40 times as many collisions per second, and the current trigger system will not keep up. Figure 1.2 shows how the number of physics processes selected by the trigger, known as the yield, fails to keep increasing as luminosity grows. The High Luminosity LHC (HL-LHC) upgrade will commence data taking in Run 4, creating a step change in the quantity of data recorded by all four major LHC experiments (ALICE, ATLAS, CMS,

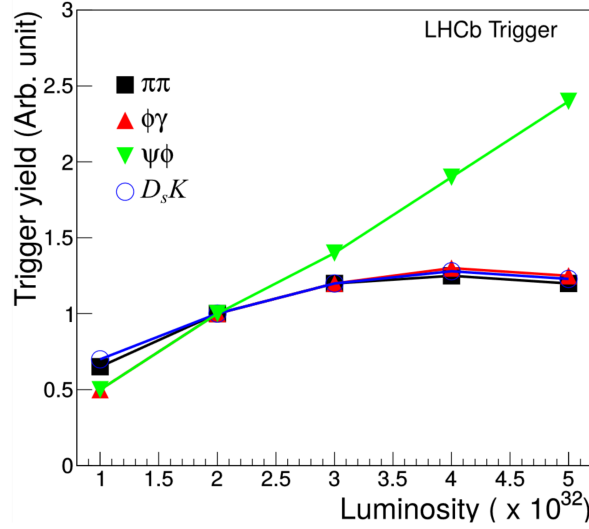


Figure 1.2: Yield of the LHCb Run 2 trigger at upgrade luminosity saturates [6]. To unlock physics performance at higher luminosity, a trigger upgrade is required. The system was designed for a luminosity of  $2 \times 10^{32}$ , was run at  $4 \times 10^{32}$  in Run 2 and is increasing to  $5 \times 10^{32}$  in Run 3 - way beyond the design limits.

LHCb). ATLAS and CMS will record 5-10 times as many events, with the average number of collisions per event (pile-up) increasing from 60 to 200, leading to a combined data rate likely to exceed  $200 \text{ Tbit s}^{-1}$  [8]. The HL-LHC will produce  $300 \text{ fb}^{-1}$  every year. Physics performance in the HL-LHC era will be determined by the power of computing resources, Fig. 1.3 shows the huge increases in CPU and disk space requirements of the CMS experiment in the HL-LHC era. Effort must now focus on how this data will be processed in the future, such as embracing scalable *big data* technologies and methods, such as machine learning, and cloud processing and storage [9].

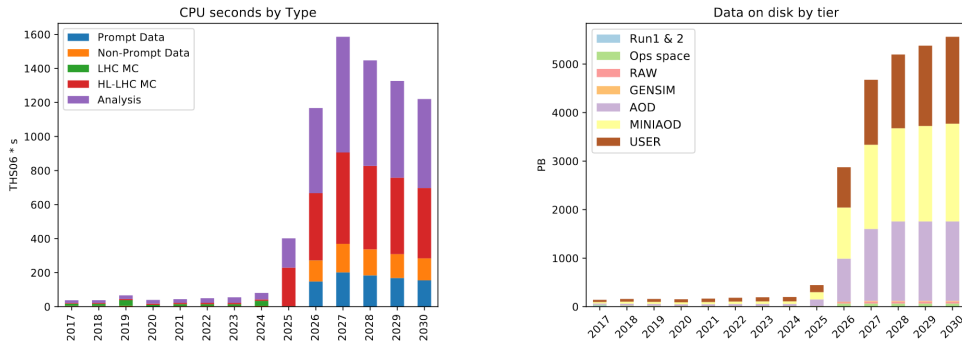


Figure 1.3: 2017 estimate of the huge increase in CPU and disk space requirements of CMS for the HL-LHC era [10]

## 1.4 Machine Learning and Big Data

Machine learning allows computers to learn from data, to achieve insights that humans could not, and to do it faster than any conventional algorithm. Machine learning is already used for many applications in high energy physics and its use is only expected to grow. Boosted Decision Trees<sup>1</sup> are used to distinguish between signal and background events better than manual cuts done by hand [11]. Where detectors produce images as outputs, such as in neutrino experiments, convolutional neural networks are used to determine which particles are present in events, as seen in Fig. 1.4. Work is being done to more deeply embed machine learning models into high-energy physics data pipelines - using tools from the world of big data and open source projects - to increase the ease at which machine learning can be incorporated into future experiments [8].

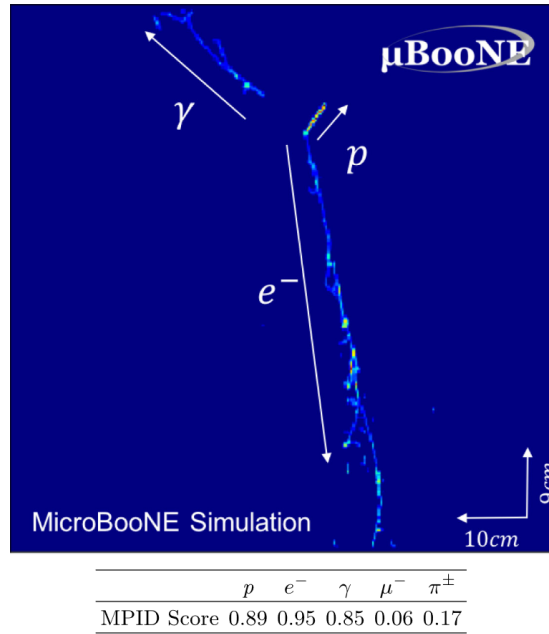


Figure 1.4: An example of tracks seen by the MicroBooNE neutrino detector [12]. A convolutional neural network can use this image to estimate the probability (MPID score) that either a photon, electron, proton,  $\mu^-$  or  $\pi^\pm$  are present in an event without prior physics knowledge, just by training with examples.

The data rate of new and future physics experiments will be very high, as seen in the LHCb upgrade and HL-LHC. Conventional trigger systems identify a small fraction of interesting events for future offline analysis [11]. However, the rate of interesting events is becoming so large that the analysis of physics interactions must begin at the moment they occur, as it is uneconomical and impractical to store this

<sup>1</sup>As simple machine learning model.

quantity of data. This new paradigm of data processing is called Real Time Analysis (RTA).

Tracking is one of the most intensive processing tasks in particle physics experiments, performing the vital role of reducing dimensionality from millions of sensor channels into higher-level track objects. Increasing hit densities of events in future experiments such as the HL-LHC will test the speed and performance of current tracking algorithms, which do not scale well with more complex events with ever increasing combinations of hits [13]. Machine learning algorithms such as neural networks are known for their fast inference times, which could enable real-time analysis of the huge datasets expected in upcoming experiments. Machine learning algorithms can also take advantage of the latest trends in hardware development, such as the move towards parallel processing on GPUs, which will only prove more important as single thread performance stalls.

In some cases ‘deep learning’ with low-level features can be very effective. Applying physics knowledge to construct high-level features for physics analysis may not always be optimal. Reducing human intervention and applying machine learning deeper into the analysis chain may provide new insights, identifying physics processes that leave only subtle hints in detectors [14].

---

# The LHCb Experiment

---

**T**HIS chapter contains an overview of the Large Hadron Collider (LHC) and the LHCb experiment located at CERN in Geneva, Switzerland. In Sec. 2.3 I will state the physics theory of LHCb and the goals for future physics discoveries. Long-shutdown 2 is currently underway at the LHC, providing an opportunity for experiments to upgrade equipment and techniques before the start of Run 3, scheduled to begin in 2022 [15]. I will describe the sub-detectors of LHCb - responsible for tracking (Sec. 2.4) and particle identification (Sec. 2.5) - as well as important components such as the online system (Sec. 2.6) and data storage (Sec. 2.7), and how they are being upgraded for Run 3. I will also provide an overview of the important physics results of the experiment, and detector performance in Runs 1 & 2 in Sec. 2.8.

## 2.1 The Large Hadron Collider

The Large Hadron Collider (LHC) is the worlds largest and most powerful particle accelerator, located at the European Organisation for Nuclear Research (CERN) near Geneva, Switzerland. Buried 100 m beneath the Franco-Swiss countryside, the 27 km long circular tunnel houses a two-ring-superconducting-hadron collider capable of accelerating bunches of protons up to a maximum centre of mass energy of  $\sqrt{s} = 14 \text{ TeV}$  [16]. The tunnel was originally completed in 1988 to house the Large Electron-Positron (LEP) Collider, which was shutdown in November 2000 to allow installation

of the LHC. The general layout of the LHC is shown in Fig. 2.1, a combination of linear accelerators and rings of increasing diameter are used to accelerate protons before injection into the main ring.

There are four large experiments placed around the LHC ring: ALICE, ATLAS, CMS and LHCb. ATLAS [17] and CMS [18] are enormous general-purpose detectors, built to study the Standard Model (SM) of particle physics and jointly announced the discovery of the Higgs Boson in 2012 [19][20]. Having two similar but independent general-purpose experiments allows results to be better verified. The ALICE experiment [21] is a detector optimised for studying heavy-ion collisions that studies quantum chromodynamics (QCD) by investigating quark-gluon plasma. ALICE takes data during proton-proton operation, but also during special Pb-Pb runs. LHCb [22] is an experiment focused on heavy flavour physics by searching for rare charge-parity symmetry violating decays involving charm and beauty quarks. More details of the LHCb detector can be found in Sec. 2.2

After commissioning in 2010, the LHC began operation in 2011 with a centre of mass energy of 7 TeV, later rising to 8 TeV [23]. Run 1 lasted two years before the first long shutdown between 2013 and 2014. Long shutdowns give time for experiments, and the LHC itself, to upgrade and replace components. A new design of magnet interconnects, amongst many other upgrades to the LHC, enabled an increase of centre of mass energy to 13 TeV in Run 2. During Run 2, instantaneous luminosity - a measure of the number of collisions - was increased to  $2 \times 10^{34} \text{ cm}^{-2} \text{ s}^{-1}$ , twice what the LHC was designed for [24]. Long shutdown 2 began in 2019, with Run 3 originally scheduled to start in 2021, however due to delays caused by the covid-19 pandemic, the expected start date is now 2022.

### 2.1.1 CERN Accelerator Complex

A complex of linear and circular accelerators boost protons up to extremely high energies, this chain involves sending protons on a journey through CERN's historical circular accelerators of increasing diameter before reaching the main ring of the LHC. The proton source is a simple bottle of hydrogen gas, an electric field is used to strip the hydrogen atoms of their electrons to yield protons. Linear Accelerator (LINAC) 4, the first accelerator in the chain, accelerates the protons to an energy of 160 MeV. The beam is then injected into the Proton Synchrotron Booster (PSB), which accelerates the protons to 2 GeV, followed by the Proton Synchrotron (PS), which pushes the beam to 25 GeV. Protons are then sent to the Super Proton Synchrotron (SPS), where they are accelerated to 450 GeV. The protons are finally transferred to the two main beam pipes of the LHC. The beam in one pipe circulates clockwise while the beam

in the other pipe circulates anticlockwise, increasing in energy until they reach their maximum [25]. Protons are steered around the ring by superconducting 8 T magnets, cooled to 1.9 K by super-fluid liquid Helium.

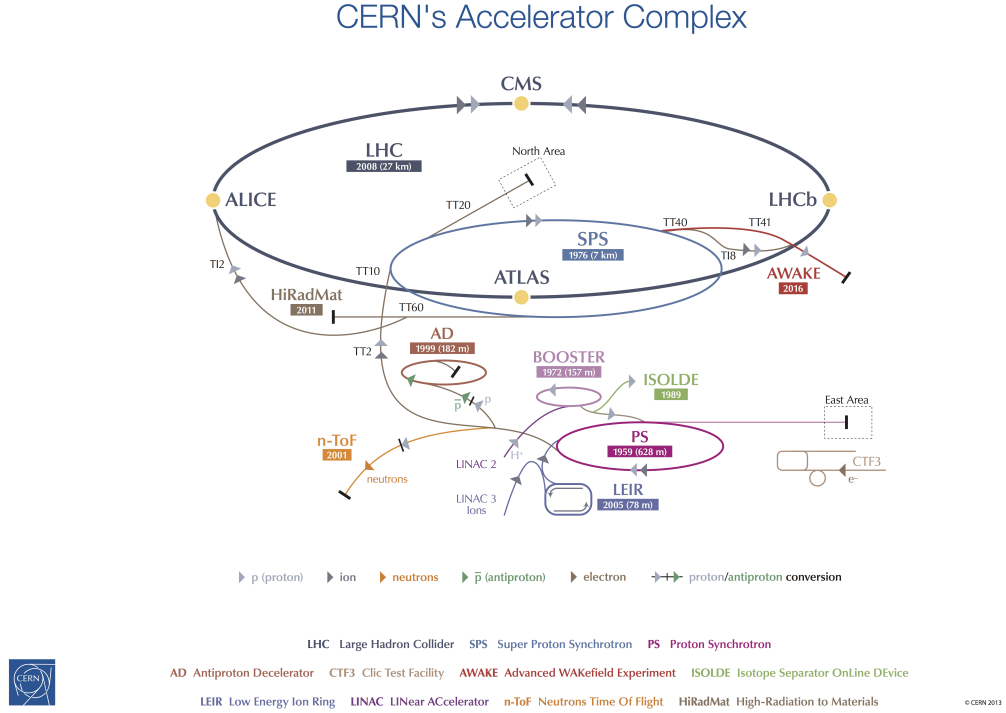


Figure 2.1: Diagram of Large Hadron Collider (LHC) facility[26].

The LHC ring is formed of eight straight and eight curved sections. Experiments and beam systems can be placed at any of the straight sections of the ring, these locations are called *points* [27]. The two general purpose experiments, ATLAS and CMS, are located at opposite points, 1 and 5. Points 3 and 7 contains the collimation system, which keeps the beams focused correctly. Point 4 houses an RF system for each beam, which uses very large electric fields in resonant superconducting cavities to accelerate protons to maximum energy. A beam dump is located at point 6 - the energy in each beam is equivalent to a train travelling at 100mph and this could cause serious damage to components if not controlled. The beam dump is used in emergencies and at the end of each fill to remove the beams. It uses kicker magnets to divert each beam into a 700m long tunnel to be absorbed. Points 2 and 8 house the ALICE and LHCb experiments respectively, as well as the beam injection facilities.



### 2.1.2 Future of the LHC

The main focus of upgrades to the LHC in long shutdown 2 are to the beam injection system, in preparation for High Luminosity LHC (HL-LHC) operation in Run 4 [28]. LINAC 2 will be replaced by LINAC 4, meaning that the energy of particles delivered to Proton Synchrotron Booster (PSB) has increased from 50 MeV to 160 MeV. Upgrades to magnets in the PSB will enable the energy of protons accelerated to be raised to 2 GeV. Various other upgrades will increase beam intensity and reduce impedance in the PS and SPS, including a new SPS beam dump, power supply system and beam diagnostics.

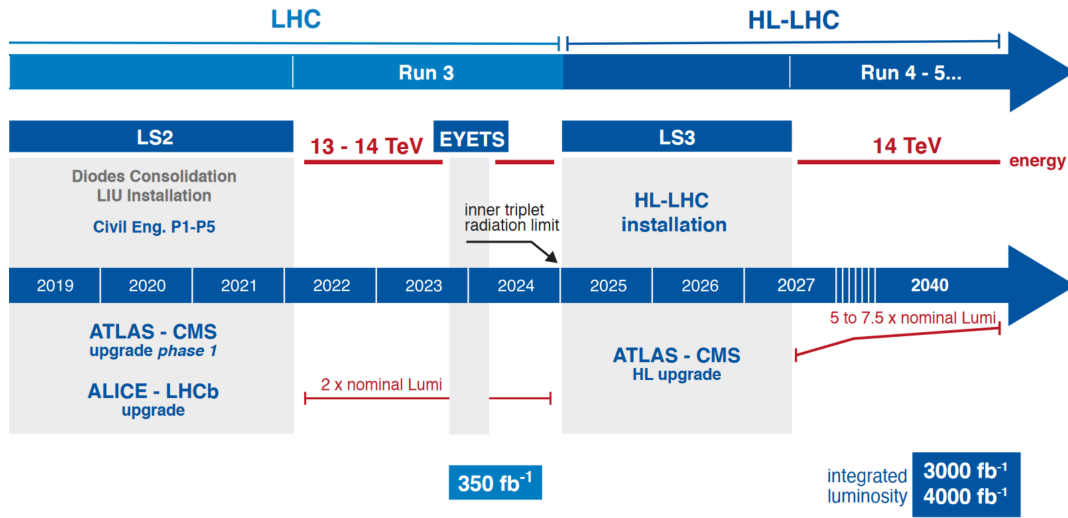


Figure 2.2: Timeline of the LHC from Long Shutdown 2 to the HL-LHC. Showing how centre of mass energy and luminosity will change over time [29].

The HL-LHC era will begin in Run 4 in 2025. In order to reduce statistical uncertainties and maintain the LHC as the world's most powerful particle accelerator, instantaneous luminosity will be increased to five times the original LHC design value, with 10 times more data collected over this era compared to the three previous runs [30]. Upgrades are required to all accelerator components, including the development of new, more powerful, superconducting magnets.

### 2.1.3 LHC Computing Program

To support the physics goals of the LHC, as well as other experiments based at CERN, a computing system - just as impressive as any detector - works behind the scenes to connect the site with itself and the world.

The world wide web was invented at CERN in 1989 by British scientist Tim Berners-Lee as a way to share information with institutions across the globe. In 1991, 80% of

the internet capacity in Europe for international traffic was installed in the CERN Data Centre. Today the Worldwide LHC Computing Grid (WLCG), aka *the grid*, is a global system of computing resources for storage, distribution and analysis that gives 12 000 physicists in 42 countries access to data generated by the LHC. Global data transfer rates on the grid exceeded 60 GB/s at the end of Run 2, a number that will only increase in the future [31].

The CERN Data Centre processes on average one petabyte per day and LHC experiments produce about 90 petabytes per year, with an additional 25 petabytes of data produced per year for data from other (non-LHC) experiments at CERN [32]. Connectivity across the CERN site is provided by an internal network of 50 000 km of optical fibre [33].

Three main systems form the backbone of CERN computing, the File Transfer Service (FTS), EOS disk storage, and tape storage. EOS<sup>1</sup> is the main disk storage system for experiment data and was developed at CERN, with a current capacity of 350 PB [34]. Data from experiments flows into EOS, before transfer to users for analysis or to permanent storage. This movement of data is managed by the FTS, which also distributes files across the grid. Long term archival data storage uses magnetic tapes. These securely hold all of CERN's physics data, about 330 PB from Runs 1 and 2, the equivalent of over 2000 years of 24/7 HD video [35].

## 2.2 Introduction to LHCb and its Upgrade

The LHCb experiment is a single-arm forward magnetic spectrometer with a polar angle coverage from 15 mrad to 300 (250) mrad in the horizontal (vertical) plane, corresponding to a pseudorapidity range of  $2 < \eta < 5$  in the forward direction [37]. Pseudorapidity is defined as  $\eta = -\ln \tan \frac{\theta}{2}$  and represents the angle in relation to the beam line. The shape of the detector is specialised for b physics, at the high energy of LHC proton collisions,  $b\bar{b}$  pairs are produced predominantly at shallow angles relative to the beam axis, in both forward and backward directions. The arrangement of sub-detectors is shown in Fig. 2.3.

Protons move along the  $z$ -axis and collide inside the Vertex Locator (VELO) at the far edge of the detector. Particles are tracked through the VELO and Upstream Tracker (UT) before passing through the magnet which bends the trajectories of charged particles. After the magnet is the final tracking station, the Scintillating Fibre Tracker (SciFi). Particle identification is achieved with Ring Imaging Cherenkov (RICH) detectors, one placed each side of the magnet. Two calorimeters measure

---

<sup>1</sup>EOS Open Storage

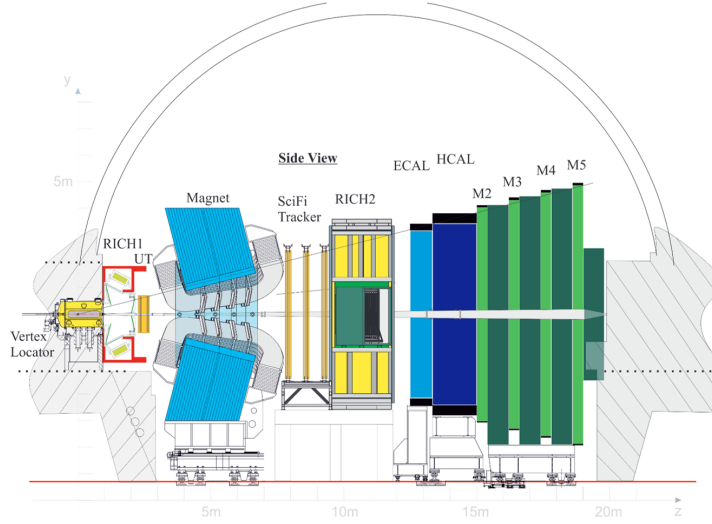


Figure 2.3: A diagram of the upgrade LHCb detector. The VELO is located on the far left of the diagram surrounding the proton-proton collision point. It is followed downstream by the UT and SciFi tracking detectors, sandwiching the magnet. Particle identification is provided by RICH 1 % 2. The further downstream detectors are the calorimeters and muon chambers [36].

the energy of particles. Electrons and photons in the Electromagnetic Calorimeter (ECAL), and hadrons in the Hadronic Calorimeter (HCAL). The final sub-detector is the muon system, formed of gas filled chambers. The detection of muons is important as they are signatures of many physics processes interesting to LHCb, such as CP sensitive B decays like  $B_s^0 \rightarrow \mu^- \mu^+$ .

Run 3 of the LHC commences after long-shutdown 2 in 2022. A significant upgrade to the LHCb detector, comprising a complete replacement of almost all sub-detectors, will enable measurements with greater precision and smaller statistical uncertainties. New front-end electronics will enable full detector readout at the 40 MHz bunch crossing rate of the LHC, feeding into a fully software High Level Trigger (HLT) [36]. As opposed to operation in Runs 1 & 2, when a simple hardware trigger reduced the collision rate to 1 MHz before the software trigger stage. This change will increase trigger efficiency by a factor of two in many important physics channels [38]. For a detailed description of the trigger upgrade see Sec. 4.1. Instantaneous luminosity will increase by a factor of five to  $2 \times 10^{33} \text{ cm}^{-2} \text{ s}^{-1}$ , pile-up will increase to  $\mu = 7.6$ , and a total of  $50 \text{ fb}^{-1}$  of integrated luminosity will be collected by 2030. Combining the two-times increase in trigger efficiency and five-times increase in luminosity, the expected signal yield will be a factor ten greater.

## 2.3 Physics at the LHC and LHCb

The LHC was built to study physics of the standard model and beyond [16]. The standard model is a theory that provides our best description of elementary particles and how they interact [39]. The standard model comprises six quarks, three flavours of lepton - electron, muon and tau - with their associated neutrinos. The forces are explained by the exchange of Gauge bosons; photons which carry the electromagnetic force, W and Z which carry the weak force, and gluons that carry the strong force. Finally is the Higgs Particle, discovered at the LHC in 2012, revealing the origin of particle masses, and filled the last gap in the standard model. The LHC has also provided high precision measurements of standard model parameters, such as the best measurement of the W mass [40] in 2021.

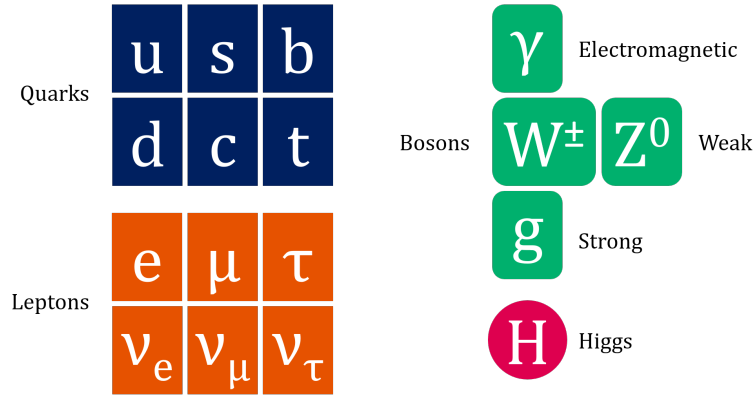


Figure 2.4: Components of the standard model of particle physics. There are six quarks: up, down, strange, charm, bottom and top. Leptons are arranged in three generations: electron, muon and tau; and their corresponding neutrinos. Gauge bosons mediate the fundamental forces. Photons carry the electromagnetic force, W and Z bosons carry the weak force and gluons mediate the strong force. The Higgs particle is a manifestation of the Higgs field, the interaction of particles with this field gives them mass.

There are many phenomena that the standard model cannot explain, but which the LHC could provide answers [41]. In order to do this we must create conditions not seen since moments after the big bang, with energy high enough to see deeper into the inner workings of the universe. Cosmological phenomena such as the rotational speed of galaxies point to the existence of dark matter, matter that only interacts gravitationally. Could a dark matter candidate could be found at the energy scales of the LHC? The discovery of super-symmetric particles could help unify the fundamental forces, could they be at the LHC?

One of the greatest mysteries of the universe is that it exists at all. The standard model predicts the equal production of matter and anti-matter at the beginning of

time. These would completely annihilate each other leaving the universe empty. In reality, the evidence of our matter filled universe means that there must have been more matter than anti-matter. This could be explained by CP symmetry violation, which exists in the standard model, but not at a level that can explain the asymmetry [22].

CP is the product of charge-conjugation and parity symmetries, which are individually violated by weak interactions, but together are generally conserved. Weak interactions involving quarks changing flavour and family, mediated by the W boson, are the most fruitful sector of physics for studying CP violation [42].

CP violation was first observed in 1964 in the decays of neutral kaons at the Brookhaven National Laboratory [43], whilst Belle (Japan) and BaBar (USA) were dedicated B experiments before LHCb. These experiments found results consistent with the standard model [42]. An upgraded experiment, Belle II, has been operating since 2019.

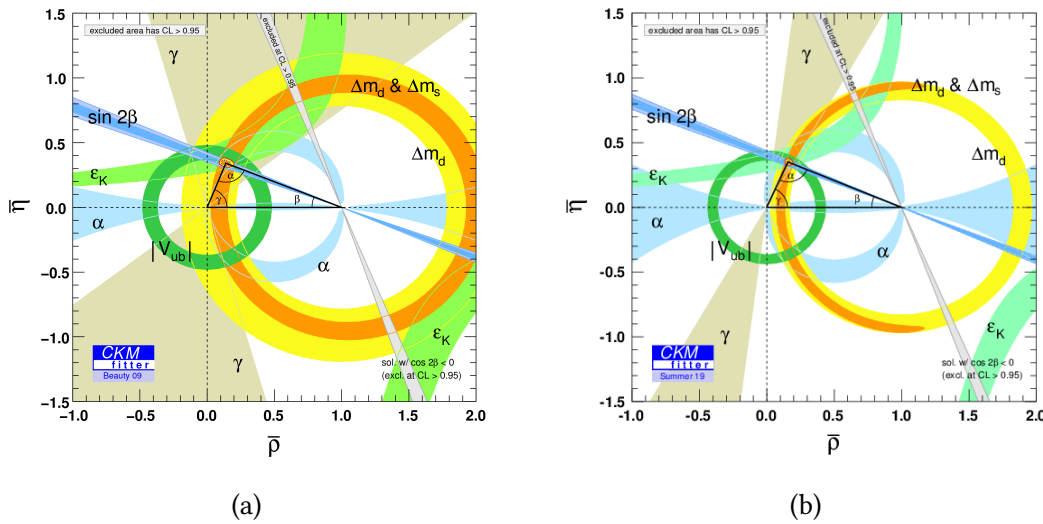


Figure 2.5: Three angles of the unitary triangle are fundamental constants in standard model. Describes flavour changing weak interactions of quarks. a) 2009 best fit before LHC. b) 2019 best fit with significantly reduced uncertainties [44].

The CKM matrix contains elements describing strength of flavour-changing weak interactions, i.e. the probability that quarks will change flavour. These CKM parameters can be represented as the side length and angles of a triangle, shown in Fig. 2.5. The goal of LHCb is to measure these parameters to check if they do or do not agree with the standard model predictions. Decays of beauty hadrons, especially B mesons, are studied because they leave distinctive signatures in detectors which can be searched for, and because these decays are sensitive to physics beyond the standard model [42].

The hardware and software of LHCb are specially designed to provide evidence of CP violation beyond the standard model, through more precise measurements of standard model parameters.  $b\bar{b}$  pairs are produced in high energy proton collisions in the LHC. They fly at a shallow angle to the beam, both in the same direction. Therefore by designing LHCb as a forward region detector, it can fully captures approximately a quarter of b-hadron decays products. B mesons have a long decay time, so can fly for on average, 1 cm in the detector, before decaying at a secondary vertex. LHCb must have precise tracking and vertex reconstruction capability close to the collision point to detect these displaced secondary vertices. A Particle Identification (PID) system is needed to categorise products of collision events, especially kaons and pions, which are in the final states of many decays important to physics analysis at LHCb. For similar reasons, an advanced muon detection system is fitted.

As an example of the high quality results that LHCb is capable of producing, a recent analysis of Run 2 data measured the oscillation frequency between  $B_s^0$  and  $\bar{B}_s^0$  to be  $\Delta m_s = 17.7683 \pm 0.0083$  ps [45]. This represents a two times improvement in precision over the previous best measurement.

## 2.4 Tracking System

The tracking system of LHCb consists of three sub-detectors. The Vertex Locator (VELO), closest to the interaction point, the Upstream Tracker (UT) just before the magnet, and the Scintillating Fibre Tracker (SciFi) downstream of the magnet. Tracks are reconstructed in each detector, then joined together if possible. The most useful tracks are those which can be reconstructed in all three sub-detectors, known as long tracks. The bending of charged particle trajectories can be measured by having tracking detectors upstream and downstream of the magnet. This gives high precision estimates of the particle momentum and mass [36]. Before describing the details of the tracking system it is helpful to define the spatial coordinate system used in LHCb. The  $z$ -axis runs parallel to the beamline, the  $y$ -axis is vertical, while the  $x$ -axis is horizontal.

### 2.4.1 Vertex Locator

The Vertex Locator (VELO) is the sub-detector of LHCb that closely surrounds the proton-proton collision point, an image of a detector half and a diagram of sensor planes is shown in Fig. 2.6. The VELO provides precise vertex positioning and track reconstruction and is an important component in the trigger system [37]. Displaced secondary vertices are a key signature of heavy flavour decays and the VELO is

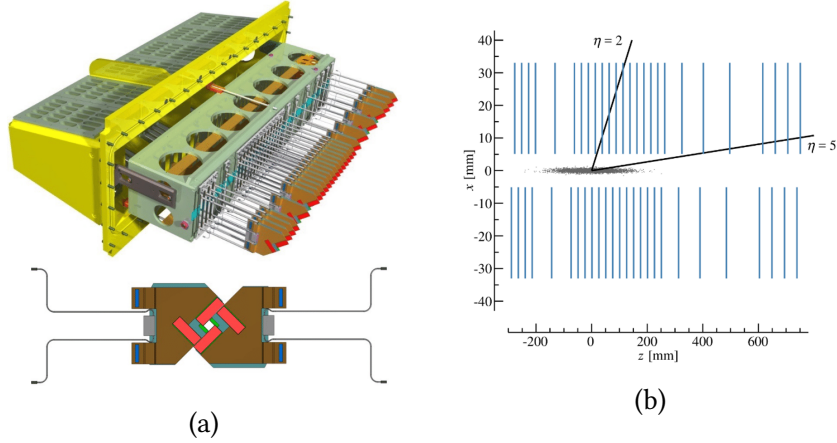


Figure 2.6: (a) Illustration of one half of the VELO detector (upper) and view of how two modules come together around the beam (lower) [46]. (b) A view of the VELO in the  $xz$  plane, with the LHCb pseudorapidity acceptance labelled, and with a shaded area where collisions occur. Pseudorapidity,  $\eta$ , measures the angle  $\theta$  to the beam and is defined by  $\eta \equiv -\ln [\tan \theta/2]$  [37].

designed to accurately determine their position. The design is optimised to detect charged particle tracks in the forward and backward region within a pseudorapidity acceptance of  $2 < \eta < 5$ . The detector consists of 52 modules holding silicon pixel sensors, the closest pixels to the beam are just 5.1 mm away, requiring a high level of radiation hardness. During the start and end of LHC fills, the sensors are retracted to limit damage from the unaligned beam. A detailed description of the VELO is given in Chap. 3.

### 2.4.2 Upstream Tracker

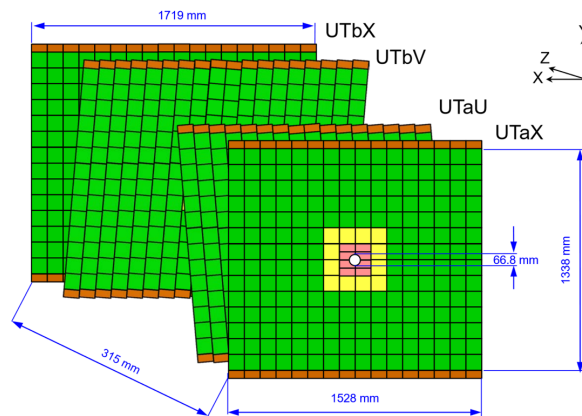


Figure 2.7: Diagram of Upstream Tracker (UT) [36].

The Upstream Tracker (UT) is the second stage of the tracking system. It provides

additional tracking information in the large 7 m gap between the VELO and the SciFi tracker, and is located just upstream of the magnet. Hits in the UT greatly reduce the number of fake tracks due to incorrectly joining VELO and SciFi track segments, and improves momentum resolution by  $\sim 25\%$ .

The UT is important for measuring downstream tracks created by decays that occur beyond the VELO, such as  $K_S^0 \rightarrow \pi^+\pi^-$  and  $\Lambda \rightarrow p\pi^-$ . The UT provides the only measurements of trajectories before the magnet in these situations. It is possible to use fringe magnetic fields between the VELO and the UT to estimate momentum of upstream tracks - tracks that are ejected from the detector acceptance by the magnet.

The UT is a four layer 250  $\mu\text{m}$  thick silicon strip detector. The strips are aligned vertically, and the middle two layers are tilted at  $\pm 5^\circ$  to measure the  $y$ -coordinate of each hit. The staves in each layer have a slight overlap in the  $x$  and  $y$  planes so there are no gaps in acceptance, and a circular hole in the centre allows the beam pipe to pass through. Because of the higher track density closer to the beam, shorter and thinner strip sensors are used in the centre part of each layer to reduce occupancy and increase resolution. Figure 2.7 shows a diagram of the detector. In the yellow shaded area, strips are of nominal length, but with half the nominal pitch, 95  $\mu\text{m}$ . In the pink shaded area, sensors have half nominal pitch and half nominal length, 5 cm. The sensors must be kept at  $-5^\circ\text{C}$  to limit the leakage current of sensors, this is done with evaporative CO2 cooling.

### 2.4.3 Magnet

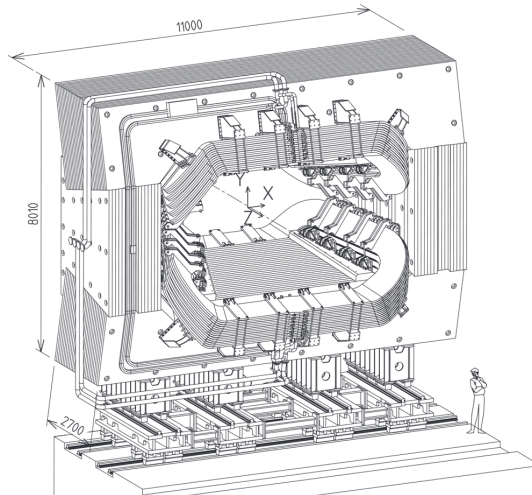


Figure 2.8: Diagram of LHCb magnet with a human for scale [47].

The tracks of charged particles curve in a magnetic field. Particles with higher momenta curve less sharply, therefore we can estimate momentum by measuring



the particle trajectory before and after passing through a strong magnetic field. The LHCb magnet is a dipole that bends in the horizontal  $x$ -plane [48]. Momentum resolution requirements of  $< 1\%$  demand an integrated field strength of 4 Tm over its 10 m length, but with minimal fringe field strength outside the magnet. The magnet consists of a steel yoke and inner aluminium coils together measuring 11 m wide and 8 m tall, weighing 1600 tons and drawing 4.2 MW of power. The coils are constructed of annealed aluminium-99.7, each coil having a  $50 \times 50 \text{ mm}^2$  cross section, with a 25 mm diameter bore for water cooling. The coils form a conical shape sculpted to the contour of the experiment acceptance.

Precision asymmetry analyses measure the difference between particles and their anti-particles. These results can be contaminated by defects in the detectors, such as misalignment of the muon stations. Therefore, the polarity of the magnet is flipped every few weeks to cancel out this effect. Particles with opposite charge but that are otherwise identical should follow the same trajectory when the polarity is flipped. When the magnetic field is along the positive (negative)  $y$ -axis it is known as up (down) polarity at LHCb. The magnet is one of the only parts of the experiment that will remain for Run 3.

#### 2.4.4 SciFi

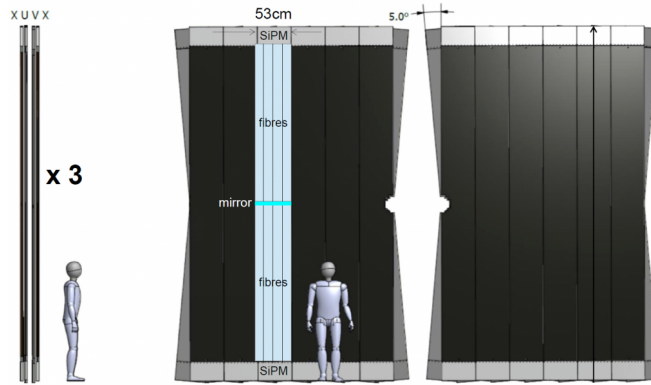


Figure 2.9: Diagram of Scintillating Fibre tracker (SciFi) [49].

The Scintillating Fibre tracker provides tracking information immediately downstream of the magnet, which is used to calculate the momentum of charged particles by measuring the bending of particle trajectories. The SciFi replaces the gas-filled straw tube outer detector from Runs 1 & 2. The sub-detector is formed of three stations, with having four detection layers. Each station is separated by approximately 70 cm in the  $z$ -axis, with a gap of 20 mm between layers. The middle two layers in each station are tilted at  $\pm 5^\circ$  stereo angle with respect to the  $y$ -axis, whilst the first and last

layer have no tilt. Tilting of these layers enables the vertical y-coordinate of a hit to be determined. The SciFi provides  $100\text{ }\mu\text{m}$  spatial resolution of hits in the bending (x) direction. In each layer there are 12 modules, each module is a fibre mat consisting of six layers of densely packed blue-emitting scintillating plastic fibres with a diameter of  $250\text{ }\mu\text{m}$  [50]. When a charged particle interacts with a fibre, photons are produced which travel up the fibre to be detected by a silicon photomultiplier (SiPM) at the end of each mat.

## 2.5 Particle Identification

### 2.5.1 Ring Imaging Cherenkov Detectors

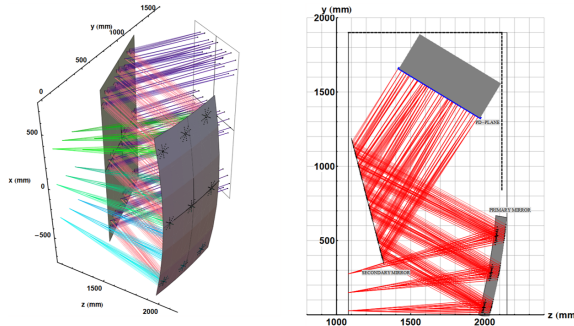


Figure 2.10: Diagram of RICH detector [51].

The Ring Imaging Cherenkov (RICH) system of LHCb provides Particle Identification (PID) of charged hadrons over the momentum range  $1.5\text{--}100\text{ GeV } c^{-1}$ . The RICH detectors utilise the Cherenkov effect, whereby a particle travelling in a medium at faster than the speed of light in that medium will emit a cone of light. The light is detected by Multianode photomultipliers (MaPMTs). Particle momentum can be estimated by measuring the angle of the cone of light, as they are strongly correlated. Three different media (radiators) are used to cover the full momentum range, silica aerogel and C<sub>4</sub>F<sub>10</sub> gas in RICH 1 and CF<sub>4</sub> gas in RICH 2 [51].

#### RICH 1

RICH 1 is located downstream of the magnet, between the VELO and UT. It will undergo significant upgrades for Run 3, with mirrors adjusted to reduce peak occupancy. In order to cover the full acceptance whilst keeping a small physical size, RICH 1 is located close to the collision point. A spherical mirror, then a plane mirror is used to focus the Cherenkov light onto photo-detectors. These are located away from the beam so that it can be shielded from the magnetic field.

## RICH 2

RICH 2 is positioned upstream of the magnet, between the SciFi tracker and the calorimeters. Except for new readout electronics and photo-detectors, it will remain unchanged for Run 3. The angular acceptance of RICH 2 is 120 mrad horizontally and 100 mrad vertically. A similar optical setup of spherical and plane mirrors are used, as in RICH 1.

### 2.5.2 Calorimeters

Calorimeters are used to measure the energy of particles produced in high-energy collisions, as well as for particle identification. Calorimeters work by utilising the phenomenon of showering, which occurs when a particle is stopped in a bulk material. There are two different types of showers, which require two different detectors to measure. The electromagnetic calorimeter measures showers produced by particles that interact via the electromagnetic force, such as electrons, positrons and photons. The hadronic calorimeter measures the energy of particles that interact via the strong nuclear force such as protons, pions and kaons.

#### Electromagnetic Calorimeter

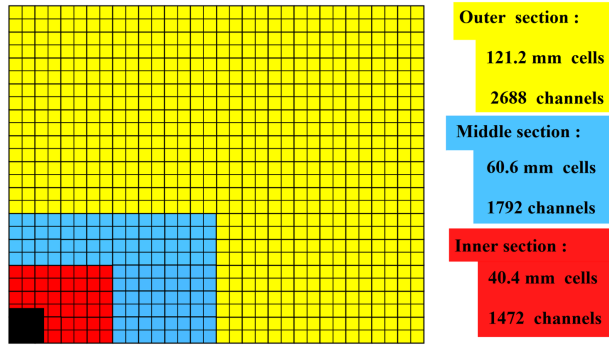


Figure 2.11: Layout of sensors in ECAL, one quarter of face shown [52].

The Electromagnetic Calorimeter (ECAL) weighs 102 tons and is built from 66 2 mm layers of Lead sheets interspersed with 4 mm thick polystyrene scintillator plates. Wavelength shifting fibres are used for light collection, with light detected with photo-detectors. For optimal resolution of showers from high-energy proton collisions, the ECAL must be quite thick,  $25\chi_0$  (interaction lengths) was chosen [52]. The bulk of this detector will remain for Run 3, however the pre-shower (PS) and scintillating pad detector (SPD) will be removed and the front-end electronics will

be upgraded to cope with triggerless readout. Like the UT, three regions of sensor granularity are used due to higher occupancy near the beam line.

### Hadronic Calorimeter

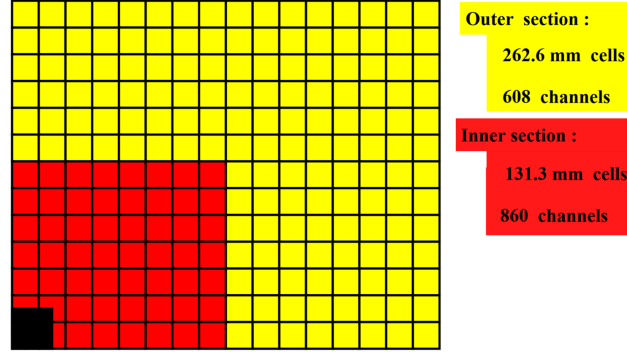


Figure 2.12: Layout of sensors in HCAL, one quarter of face shown [52].

The Hadronic Calorimeter (HCAL) is 8.4 m high, 6.8 m wide and 1.65 m deep and weighs in at 490 tons. It can be split in two halves to be moved apart for access. The bulk is formed of steel and polystyrene scintillating plates. Two sizes of sensor are used for the inner and outer section. Much like the ECAL, only the front-end electronics of the HCAL will be replaced for Run 3. Also as part of the upgrade, the PMT gain will be reduced by a factor five in order to increase their lifespan. To compensate, gain from the front-end electronics will be increased by the same factor.

### 2.5.3 Muon System

Muons are present in the final states of many CP sensitive B decays, for example  $B_s^0 \rightarrow \mu^- \mu^+$  [53]. The Muon system a vital component of online and offline reconstruction and can achieve 20% transverse momentum ( $p_t$ ) resolution. During Runs 1 & 2 data from the Muon system was one of a small number of features fed into the first stage of the trigger system.

The detector originally consisted of four stations, interleaved with shields to attenuate hadrons, electrons and photons. For Run 3, the first station (M1) will be removed, leaving three stations downstream of the calorimeters. Each station has four regions of sensor granularity, with multi-wire proportional chamber (MWPC) sensors used for the inner regions and resistive plate chambers (RPV) for the outer regions. Each station has two redundant sensor systems, to ensure high hit efficiency.

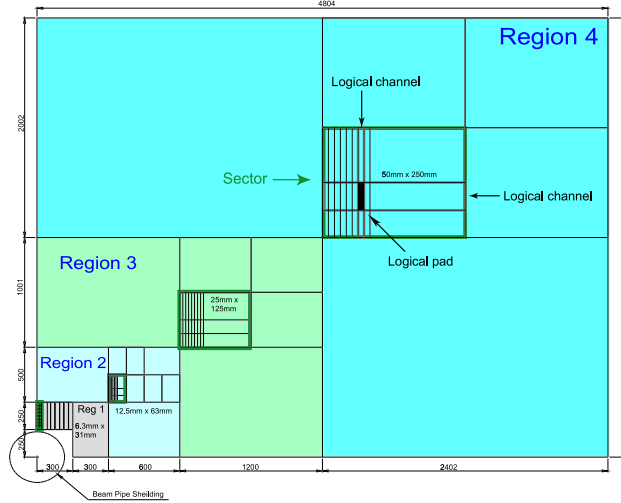


Figure 2.13: Diagram of the face of one station of the muon detector. Details of each sector are shown [53].

## 2.6 Online system

As important as any sub-detector is the online system, the computing infrastructure backbone of the experiment. It is responsible for data acquisition, experiment control and analysis [54].

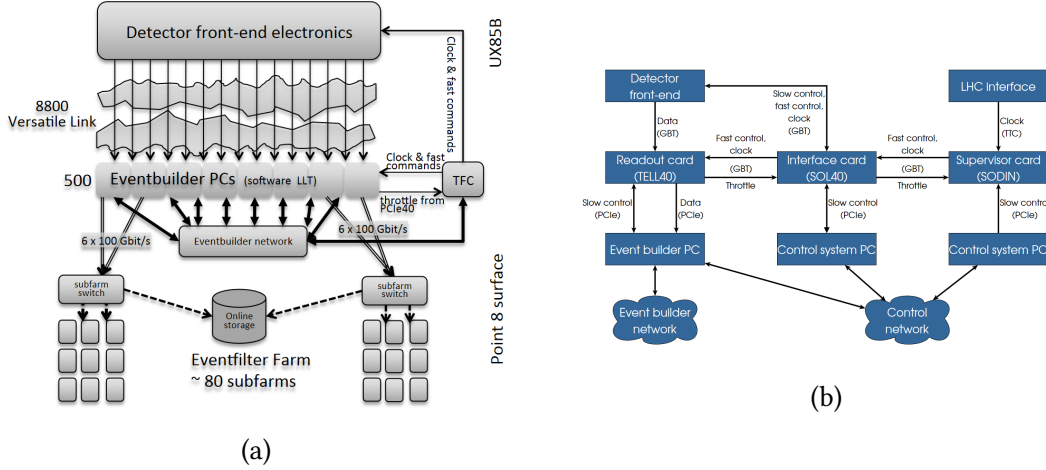


Figure 2.14: (a) Upgraded LHCb readout system [54]. (b) Timing and Fast Control system [55]

### 2.6.1 Timing and Fast Control

The Timing and Fast Control (TFC) system is the brain of the experiment. It is responsible for managing the transfer of event data from the detector front-end to the Event builder and EFF for processing, and keeping the experiment in sync, by sending

clock and trigger commands. For Run 3 all front-end electronics of each sub-detector will be replaced to cope with triggerless readout at 40 MHz. Zero-suppression detector data is transferred over almost 10 000 optical links from the detector to TELL40 readout boards. Each optical link is capable of  $4.48 \text{ GB s}^{-1}$  using a custom protocol known as GBT (GigaBit Transceiver) developed at the LHC for use in high radiation environments [56]. The readout card sends event fragments to event builder PCs for assembly into full events. Supervisor boards, called SODIN, manage the readout system, distributing the clock and regulating the data flow from the detector front-end to the readout boards. Between the readout and supervisor boards are interface boards (SOL40) which distribute signals from supervisor boards and interface with the experiment control system (ECS).

### 2.6.2 Event Builder

The Event Builder combines event fragments from the detector front-end readout into single computing nodes so that each event can be processed in the HLT1 and sent to the EFF for further processing. Event data arrives at a combined  $40 \text{ Tbit s}^{-1}$  from the front-end electronics onto a PCIe40 FPGA hosted by each Event Builder PC. Data is pushed from this card onto the memory of the Event Builder PC in the form of a multi-event fragment packet (MEP) to reduce packet rate, each MEP contains data from approximately 1000 events. All MEPs with data from the same bunch crossing are combined on a single PC. For each MEP, one PC is designated as the *event builder PC* and all other PCs send their MEPs to this PC. The Event Builder network will comprise 170 PCs, each hosting three PCIe cards. GPUs will be incorporated into Event Builder PCs to run HLT1, reducing the data rate into the Event Filter Farm [57].

### 2.6.3 Event Filter Farm

The Event Filter Farm (EFF) in Run 3 will run the second stage of the High Level Trigger to fully reconstruct event data. It is formed of PCs in a containerised data centre located on-site at Point 8 which will also house the Event Builder PCs. The EFF is responsible for reducing the data rate from the detector to a level appropriate for storage. In order to maintain high utilisation of computing assets during periods when the LHC is not running, the computing power of the Event Filter Farm can be used for offline data processing and simulation. Monitoring and configuration is managed by the Experiment Control System (ECS), which provides an interface between the experiment and operators in the control room.



Figure 2.15: The new LHCb containerised data centre under construction at point 8. [58]

## 2.7 Data Storage

The bulk of LHCb data storage is handled by the Worldwide LHC Computing Grid (WLCG) in the form of disk and tape. With at least two copies of collected data stored as backup. Estimates for storage requirements by the end of Run 3 are 159 PB on disk and 345 PB on tape [38]. On site, in the Event Filter Farm a disk buffer is needed between the two stages of the High Level Trigger. Assuming 170 event builder PCs, each with a 8 GB disk buffer, the buffer would store approximately 0.45 s of HLT1 output [57].

## 2.8 LHCb Performance

### 2.8.1 Detector Performance and Physics Discoveries

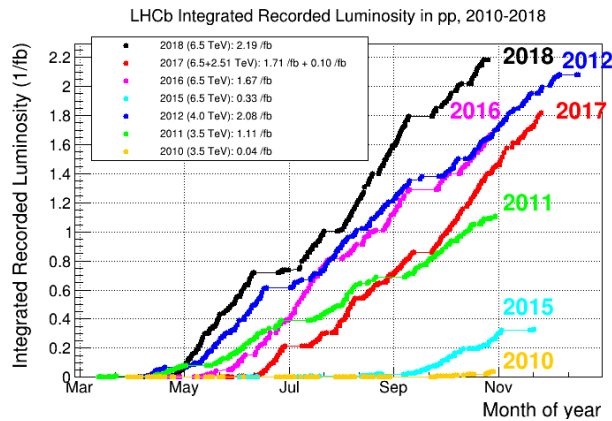


Figure 2.16: Integrated luminosity recorded each year of Runs 1 and 2 [59].

An integrated luminosity of  $9 \text{ fb}^{-1}$  was collected in Runs 1 & 2. With  $10^{12}$  heavy flavour decays collected in Run 1 at a centre of mass energy of  $\sqrt{s} = 7 \text{ TeV}$ , at the time the world's largest sample of charm and beauty decays [60]. The collision energy was increased to a maximum  $\sqrt{s} = 13 \text{ TeV}$  in Run 2, accompanied by an increase in instantaneous luminosity also. To increase Primary Vertex precision, pileup was kept to about  $\mu = 1$ , meaning one collision per bunch crossing. LHCb is aiming to record  $50 \text{ fb}^{-1}$  of collision data in Runs 3 & 4. In its first eight years of data taking, the LHCb experiment has produced many important results in the fields of CP violation, precise measurements and new particle searches. Results so far hint of new physics, but as yet not enough significance has been recorded. Below is a brief summary of a variety of results from the first eight years of LHCb operation.

Confirmation of CP violation in charm decays by measuring the difference in decay rates of  $D^0$  and  $\bar{D}^0$  to  $K^+ K^-$  and  $\pi^+ \pi^-$  pairs. The difference in asymmetries was measured to be  $\Delta A_{CP} = -0.154\% \pm 0.029\%$ , with a significance of  $5.3\sigma$  [61].

Lepton universality was studied by measuring the ratio  $R_k$ , for decays involving  $b \rightarrow s$ , which is heavily suppressed in standard model [62]. The value of  $R_k$  was measured to be consistent with the standard model to  $2.5\sigma$ . Another results consistent with the standard model prediction was the observation of the  $B_s^0$  to  $\mu \mu$  decay [63]. Measuring the CP violating phase  $\phi_0$  one of main goals of the experiment, it was found to be consistent with standard model [64].

LHCb has discovered many new particles in its first phase of operation. Almost 30 new hadrons were discovered in Runs 1 and 2 including tetra and penta-quarks (particles containing four or five quarks) [65]. Penta-quarks were found from the decay  $\Lambda_b \rightarrow J/\psi p$  [66]. A new Charmonium particle was discovered, a meson consisting of charm and anti-charm quark [67].

LHCb results have also increased the gap between experiment and theory. A measurement of the branching fraction of  $B_s^0 \rightarrow \phi \mu^+ \mu^-$  increased the tension with the standard model to  $3.6\sigma$  [68], a further hint of possible new physics.





---

# VELO Detector

---

**I**N this chapter I will describe in detail the Vertex Locator (VELO), which provides precision tracking and vertex finding for the LHCb experiment [37].

Unique to LHC experiments, and as outlined in Sec. 2.2, the VELO provides tracking measurements immediately around the proton-proton collision point. This is key for finding displaced secondary vertices associated with B and C decays that the physics programme of LHCb focuses on [69]. In Sec. 3.2 I will describe the material and physics requirements that motivate the design of the detector described in Sec. 3.1. Accurate VELO tracking are necessary for the successful physics program, since VELO tracks are a vital component of the new High Level Trigger (HLT) for Run 3, and because tracks are built starting from VELO tracks. Finally in Sec. 3.3 I will provide details of the service task I undertook whilst on Long Term Attachment at CERN to visually inspect VELO front-end hybrids. I describe the issues that were identified and the solutions that were found.

### 3.1 VELO Upgrade Design

The VELO detector surrounds the proton-proton collision point at LHCb. It measures approximately one metre long, with sensors 5 cm square in cross section spread along its length. Sensors are located such that at least four measurements are made of tracks in the experiment pseudorapidity acceptance of  $2 < \eta < 5$ . During the start of

each LHC fill, the sensors are moved apart 3 cm to avoid radiation damage from the unfocused beam [70].

The sensors in each half are surrounded by the RF foil. This foil provides a physical barrier between the primary LHC vacuum and the secondary VELO vacuum, and protects front-end electronics from electric currents induced by the beam. Enclosing the modules in a secondary vacuum prevents molecules outgassing from the modules affecting the LHC vacuum. To reduce multiple scattering the material budget of the detector must be as small as possible, therefore the RF foil is just 250  $\mu\text{m}$  thick. The foil has a complex shape to fit around the sensors and is machined from a solid block of Aluminium, then etched to final thickness.

Figure 3.1 shows a diagram of the VELO module positions in the  $xz$  plane. The collision point is off-centre in the upstream ( $z$ ) direction so that most tracking measurements are collected downstream, towards the experiment. It is also important to measure upstream (backwards) tracks to improve primary vertex reconstruction [70].

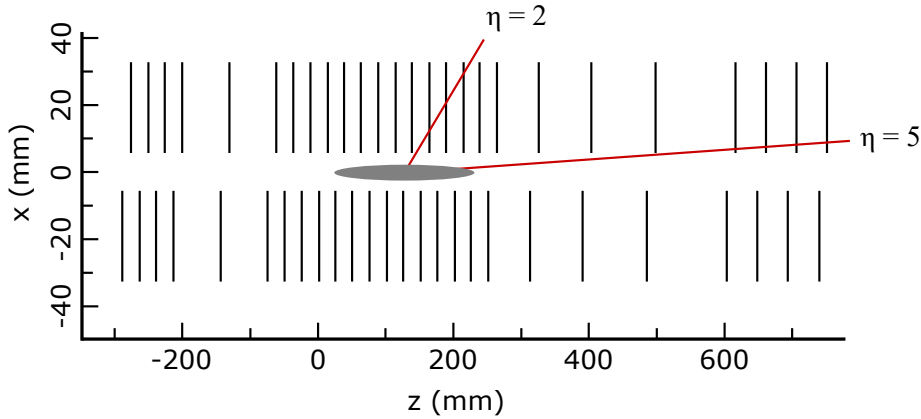


Figure 3.1: View in the  $xz$  plane view showing VELO module positions in black and pseudorapidity acceptance in red. The grey area is the approximate collision region

The overall design and size of the brand new VELO detector for Run 3 is very similar to that used in Runs 1 & 2 but with several important changes. The 52 modules that carry the sensors have had their  $z$ -positions tweaked to make sure most tracks remain in acceptance, despite the change in sensor shape from circular to square. The edge of the sensors sit very close to the beam, this distance has been reduced from 8.2 mm to 5.1 mm in the new detector.

The design of the entire sensing and readout system has been changed for the upgrade, silicon strip sensors have been changed to silicon pixels, and the new readout and data acquisition system can handle real-time data taking required for Run 3 [37].



for these components is stringent. They must be extremely radiation hard, a maximum hadron flux of  $8 \times 10^{15} \text{ n}_{\text{eq}}/\text{cm}^2$  for the sensor tip closest to the beamline. There are significant challenges to cool the sensors and readout ASICs. Sensors are bump bonded to a readout ASIC, which is glued to the substrate. The substrate contains micro-channels through which liquid  $\text{CO}_2$  flows at  $-40^\circ\text{C}$  to cool the sensors, which must be maintained at  $-20^\circ\text{C}$  to avoid thermal runaway. Sensors are wire bonded to front-end hybrids, from which cables for data and power are connected.

The whole module assembly can move in the  $x$ -axis, this is so that during beam fills the sensors are protected from high radiation fluences caused by a misaligned beam. Sensors protrude 5 mm from the substrate at the edge closest to beam, this reduces the material volume in this area to improve vertex finding precision, however this also makes the most active part of the sensor more difficult to cool. There is a two pixel overlap between sensors on opposite sides of the module to make sure steeply angled tracks are reconstructed. Modules for the upgrade detector are constructed at both Manchester (UK) and Nikhef (Amsterdam, NL), they are shipped for final assembly at Liverpool (UK) before being installed at CERN [73].

### 3.1.2 VELOPix readout

When particles pass through sensor pixels they deposit a small amount of charge, this is called a *hit*. Hits from the pixel sensors are readout by VeloPix ASICs which are bump bonded to the sensors to provide electrical and mechanical connection. To minimise data flow VELOPix uses a zero-suppression binary readout, meaning that only pixels which register hits are readout. Figure 3.3 shows how the data rate of each pixel varies. Each hit contains a row and column number for the cluster, with a cluster size (number of pixels). Because 55% of hits will deposit energy in more than one pixel, due to the small pixel size and steep track angles, pixels are combined into  $2 \times 4$  groups called *super-pixels*. This minimises redundant data transfer because only one time stamp is needed for each super-pixel, rather than for every pixel, which leads to a 30% lower bandwidth. The time stamp identifies the bunch crossing number but cannot be used for tracking. Adding timing information to the track reconstruction process is known as *4D tracking*, which may be possible after future upgrades in Run 4 [74].

The move to a triggerless readout for Run 3 means that the VELO front end electronics must be capable of read out at 40 MHz. Each ASIC must cope with data rates of 15.1 Gbit/s, with a total VELO output data rate of 2.85 Tbit/s. Zero-suppression readout means that the hottest ASICs have a considerably higher data rate from those further from the beam.

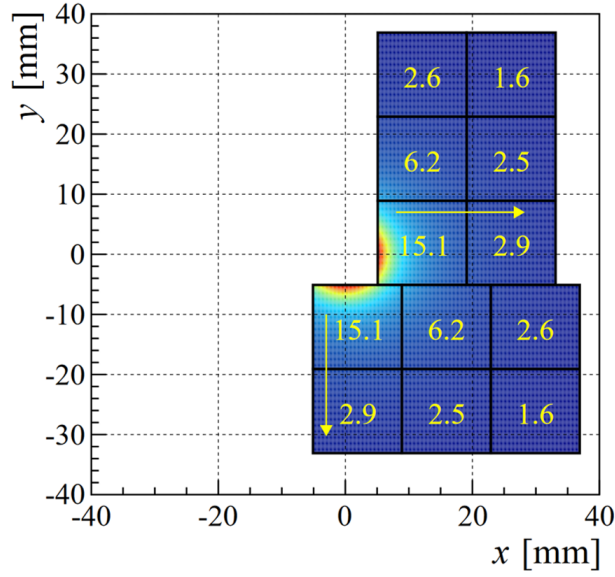


Figure 3.3: Data rate per ASIC in  $\text{Gbs}^{-1}$  for most active module, the arrows denote readout direction. The colour represent the data rate in each pixel, this image shows that the very inner pixels produce far more data than those further from the collision point. [37].

### 3.1.3 Electronics and DAQ

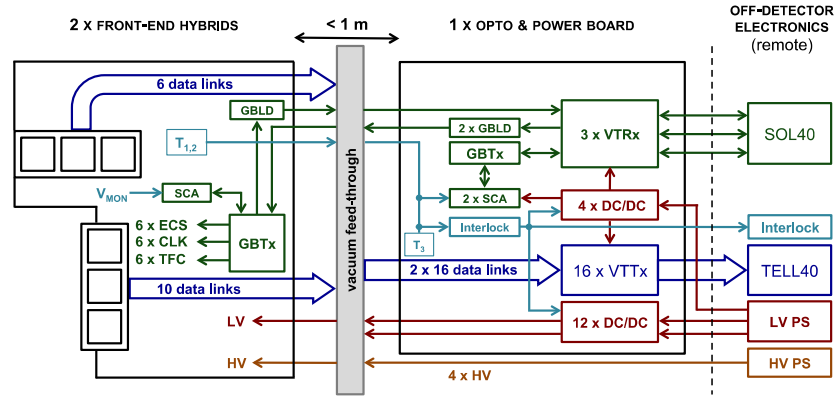


Figure 3.4: Schematic of VELO electronics. Each module has a front-end hybrid on each of its two faces. [37]

The design of the VELO electrical architecture is motivated by the need to locate as little as possible inside the secondary vacuum tank [37]. This is because of cooling challenges, the high radiation dose, material budget in the detector acceptance and the difficulty of access for maintenance. No optical devices are used inside the secondary vacuum tank, instead, low-mass electrical cables are used to send data and control signals. The DC/DC converters that power the front-end ASICs are also placed outside the vacuum tank. Figure 3.4 shows a diagram of the VELO electronics.

Data flows from the front-end electronics, through the vacuum feed through into an Opto-Power board (OPB). This board provides the interface between electrical and optical signals, which are sent to the surface. The OPB also converts DC/DC power for supply low-voltage to the front-end electronics. High-voltage power for biasing the sensors is supplied via a separate vacuum feedthrough.

## 3.2 VELO Upgrade Requirements

### 3.2.1 Physics

In Run 3, crucial trigger information will be provided by the VELO that was not available before. B meson decays can be identified by long decay times, meaning particles travel a significant distance within the VELO before decaying. This distance is called the Impact Parameter (IP). The VELO must be able to identify large impact parameter decays in real time. The first measurements must be as close as possible to the primary vertex, because impact parameter resolution improves approximately proportionally to the radius of the first hit from the beam. To facilitate accurate tracking, the module positions are optimised so that 99% of particle tracks within the angular acceptance of LHCb, that originate from near the interaction region, leave hits in at least four sensor layers. Figure 3.5 shows results of simulations indicating the improved impact parameter resolution and tracking efficiency possible with the new detector. Equation 3.1 was used to calculate the z-distance between central stations, where  $\theta_{max}$  corresponds to  $\eta = 2$  and  $R_{in}$  and  $R_{out}$  are the inner and outer radius of the sensors [37].

$$\Delta z = \frac{R_{out} - R_{in}}{4 \tan \theta_{max}} \quad (3.1)$$

### 3.2.2 Radiation

The radiation environment that the VELO inhabits is harsh and non-uniform, with a factor 40 difference in fluence between pixels closest and furthest from the beam. Figure 3.6 shows how fluence changes with radius from the beam. A distinctive feature of the design are the movable modules. The innermost pixels are only 5.1 mm from the beam which could lead to radiation damage from beam injection, during which the cross sectional area of the beam is increased. To reduce the risk of sensor damage, each half of the VELO is moved out 3 cm until the beam is stabilised and data taking

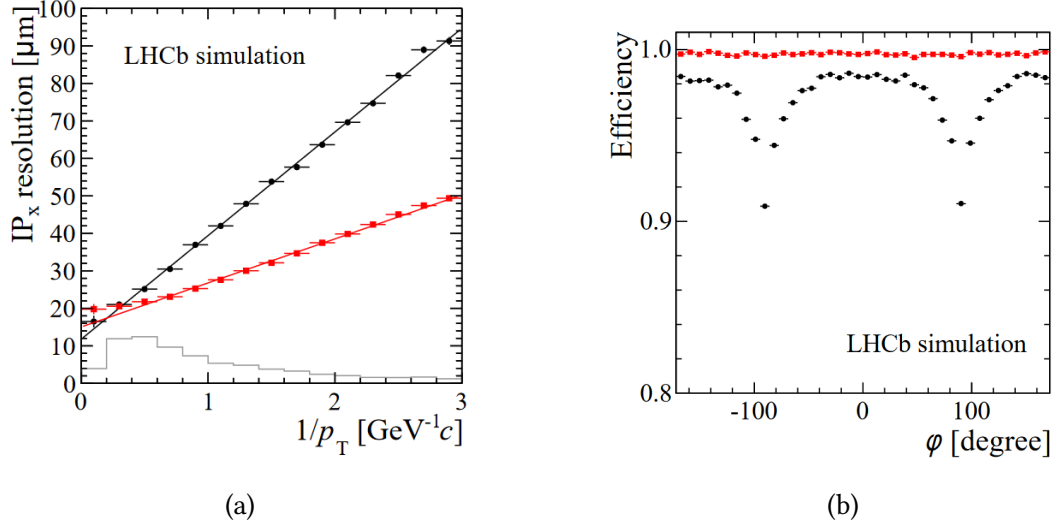


Figure 3.5: Simulation results from the VELO upgrade TDR showing improvement between original VELO (black) and upgrade VELO (red). a) IP resolution and b) efficiency with respect to phi [37].

can begin. Radiation can reduce signal by trapping charges, which can be mitigated by a high bias-voltage which creates a large electric field in the sensors. The sensors are also very thin, thin sensors are less affected by irradiation and reduce the material budget.

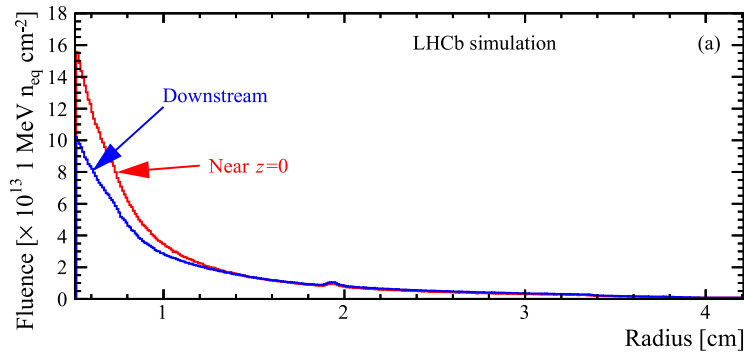


Figure 3.6: Radiation dose as a function of radius from beam for two sensors, one near the interaction point, one far away [37]. This demonstrates how unequal the radiation dose is for pixels closest to the beam

### 3.2.3 Material

In order to minimise the effects of multiple scattering, the material budget in the acceptance region must be kept as low as possible. Scattering reduces tracking efficiency and impact parameter resolution which reduces data quality. A minimal material bud-



get also reduces the impacts of radiation damage to exposed parts [73]. The choice of micro-channel cooled silicon substrate is partly due to it having a very low material budget, at only  $400\text{ }\mu\text{m}$  thick, with the sensors only  $200\text{ }\mu\text{m}$  thick.

The effects of material are quantified by the fraction of the radiation length,  $x/X_0$ . Simulation shows that the upgrade VELO has a similar overall material budget to the old VELO,  $\sim 21.3\%$  to  $20\%$  respectively, however the material budget of the upgrade VELO before the first hit within the detector acceptance is 2.7 times less than the old VELO. Approximately 53% of the upgrade budget comes from the RF-foil.

### 3.2.4 Cooling

Sensors must be maintained at  $-20\text{ }^\circ\text{C}$  to reduce the chance of thermal runaway due to radiation damage. The original VELO was cooled by  $\text{CO}_2$  passing through cooling blocks at the base of the module substrate, the upgrade VELO still uses  $\text{CO}_2$ , but it is instead passed through micro-channels in the substrate itself at  $-30\text{ }^\circ\text{C}$ . Thereby directly reaching the areas that need cooling the most [46]. Sensors are bump-bonded to VELOPix ASICs, which are then glued directly to the substrate. This enables heat to easily transfer from the ASIC to substrate.

## 3.3 VELO front-end hybrid visual inspection

As a service task whilst on LTA<sup>1</sup> at CERN in 2019 I was involved with VELO module production, namely assisting Jan Buytaert<sup>2</sup> visually inspecting front-end hybrids before they could be incorporated into modules. The front-end hybrid provides the connection between the sensors and the outside world. The sensors are bump-bonded to VeloPix ASICs, which are in turn wire-bonded to pads on the top edge of the hybrid. Connector sockets are provided to enable attachment of low-voltage cables and data tapes. High voltage for biasing the sensors is delivered in a separate cable directly to the sensor. It is important that all VELO module components are produced at a high standard because any issue during operation is very difficult to fix, due to the VELO being encased in a vacuum chamber and being exposed to radiation. Figure 3.7 shows images of stereo microscope used for the visual inspection process, the front-end hybrid itself and the view through the microscope at the solder joints of a connector socket.

There were a number of issues with the first batches of front-end hybrids delivered to CERN. The main areas of concern were damage to the wire-bonding pads, bad

---

<sup>1</sup>Long term attachment

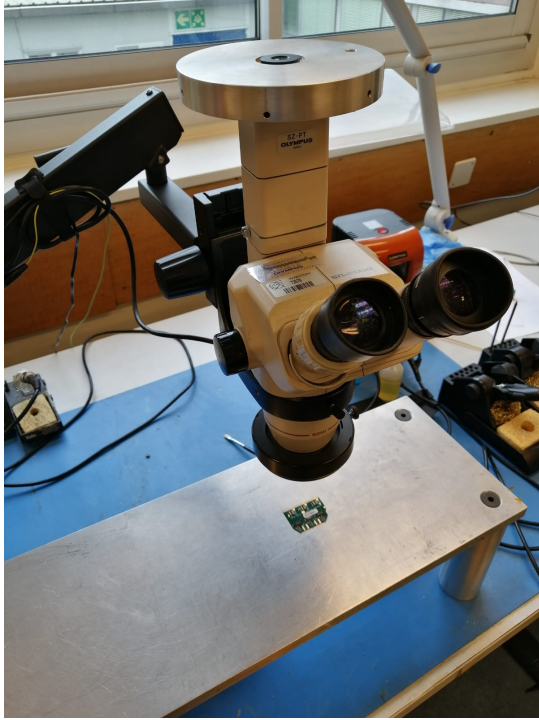
<sup>2</sup>CERN

Test	PM		PM & JB	
	Pass	Pass %	Pass	Pass %
Flatness	46	97.9%	60	96.8%
Traces	38	80.9%	51	82.3%
Bond Pads	26	55.3%	34	54.8%
Connectors	44	93.6%	59	95.2%
<b>Grade A</b>	<b>20</b>	<b>42.6%</b>	<b>27</b>	<b>43.5%</b>

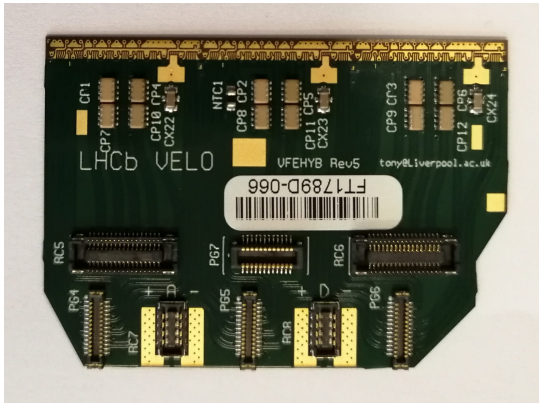
Table 3.1: Table of results for the visual inspection of 62 front-end hybrids at CERN. Showing the number and percentage of hybrids that passed each test of the inspection. Less than half of hybrid reached grade A status. Inspections were performed by Phillip Marshall (PM) or Jan Buytaert (JB).

soldering of connector sockets to the hybrid (Figure 3.7c), and damage to the traces (internal wires that connect the components of the hybrid). The solder issues were sometimes caused by the hybrids not laying flat, meaning the connectors could not sit properly on the surface. Table 3.1 shows the results of the visual inspection. Less than half the hybrids tested reach Grade A standard, the the rest were graded F. The most common cause of failure was damage to bond pads, seen on 45% of hybrids tested.

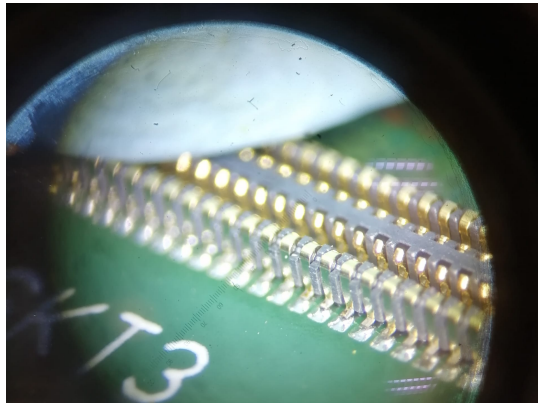
After these issues were reported, the causes were quickly discovered. The bond-pad issues were caused by small pieces of solder landing on the bond-pads whilst components were soldered to the hybrid, this was solved by protecting the bond-pads with Kapton tape whilst soldering. The bad soldering of connector sockets was caused by the warping of the hybrid itself, the long flat connectors would not sit properly on the bent hybrid making it difficult to solder. There were also some connections that were simply poorly soldered, these could be re-done at CERN. Hybrids with damage to traces could not be salvaged and would not be used in module production. After feedback from CERN further batches of hybrids were of a much higher standard and module production could commence.



(a)



(b)



(c)

Figure 3.7: a) Stereo microscope setup, demonstrating the size of each hybrid. b) Photograph of front-end hybrid. The bond pads - which connect to the VeloPix ASICs - are the gold areas along the top edge of the board. The solder joints of all sockets (PGx and RCx) and capacitors (CPx) were inspected. c) Connector solder joint through microscope.

---

# LHCb Software and Computing

---

As important as the physical detector, the software of LHCb processes and manages the enormous flow of data starting at the detector readout and ending with physics analysis. Many collision events produced at the LHC do not contain interesting physics processes, and due to limited computing and storage facilities it does not make sense to record all events. Therefore a trigger is required to reject those events which do warrant further analysis. In this chapter I will describe the trigger system of LHCb in Sec. 4.1, how it has developed over time and how it has been changed for Run 3 in Sec. 4.1.5. In Sec. 4.2 I will introduce the existing LHCb tracking pipeline - including clustering of hits, vertex reconstruction and tracking algorithms in the VELO, UT and SciFi - as well as describing particle identification and detector alignment.

## 4.1 The LHCb Trigger

### 4.1.1 Physics Requirements

Not all proton collisions in the LHC generate products important to physicists at a specialised heavy-flavour experiment such as LHCb.  $b\bar{b}$  pairs are produced in only 1% of collisions, and out of these, only 15% contain a B-meson with decay products that are all produced within the acceptance of the detector [75]. Then considering that the branching fraction of decays used to study CP violation are typically less

than  $10^{-3}$ , it is clear that a significant process of background elimination is required. The raw data rate from the upgraded LHCb detector in Run 3 will be of order  $\text{TB s}^{-1}$ . Storing this quantity of data for processing at a later date would be impractical and expensive. A trigger is used to identify and begin processing of events interesting to the physics programme of the experiment, and importantly rejecting background events. The trigger must detect important events with the highest efficiency, whilst also rejecting as many background events as possible. Figure 4.1 shows the data flow of the upgraded LHCb experiment. The trigger system reduces the data rate from  $5 \text{ TB s}^{-1}$  from the detector readout to  $10 \text{ GB s}^{-1}$  into offline processing and storage.

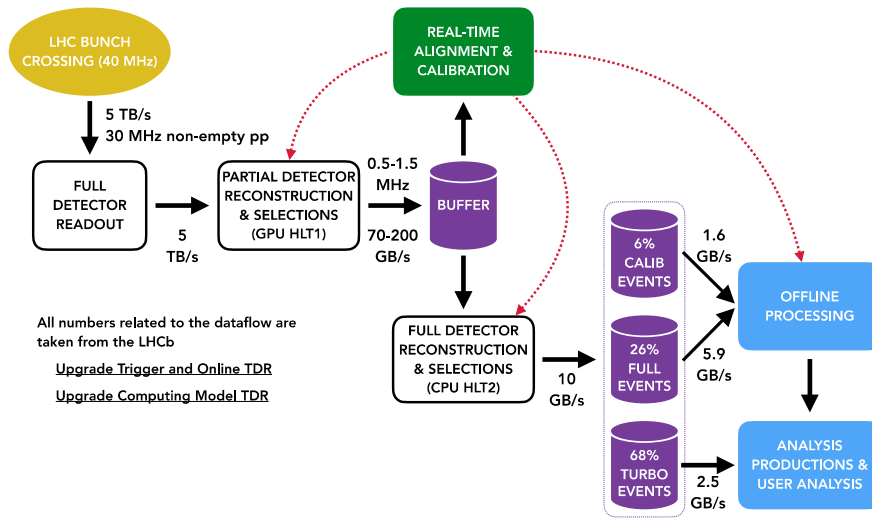


Figure 4.1: LHCb upgrade dataflow from detector readout at 5 TB through the High Level Trigger and in to offline processing and analysis. The majority of events are processed in the Turbo Stream, but only account for a quarter of data throughput [76].

### 4.1.2 Inclusive and Exclusive Selections

Trigger decisions, or selections, are split into two categories, inclusive and exclusive. Inclusive selections only require a part of decay to be reconstructed. They are efficient as they capture more signal events, however this also means they can select more background processes. Exclusive selections only trigger when all decay components are reconstructed. The bulk of events selected by the first stage of the trigger are from a few inclusive trigger lines. A trigger line being the criteria and threshold to select or reject, for example requiring a Muon with  $p_T > 1.76 \text{ GeV } c^{-1}$  reconstructed in the Muon system.

### 4.1.3 Run 1 and 2 Trigger

The original design of the LHCb trigger used during Runs 1 & 2 contained a hardware trigger at its first stage, Level-0 (L0). This reduced the event rate from 40 MHz to about 1 MHz using data from only the detectors that could be readout at the full event rate of the LHC: the calorimeters, muon system and pile up sensor (which recorded the number of interactions per bunch crossing).

The efficiency of the L0 hardware trigger varied depending on the decay channel. For B decays producing two muons the L0 requirements are more than 90% efficient, however fully hadronic charm decays have efficiencies of 20-30% due to the lower mass of the charmed hadrons [77]. The low efficiency of this simple trigger stage for some decays was a major factor in the removal of the L0 trigger for Run 3, see Sec. 4.1.5.

The output of the L0 trigger at 1 MHz was fed into the software based High Level Trigger (HLT). In the first stage, HLT1, events could only be partially reconstructed due to computing restraints. Partial reconstruction involved finding tracks in the VELO and extrapolating them if they had a large Impact Parameter, or if they can be matched with hits in the muon chambers. Tracks were then extended in the tracking stations by searching for hits consistent with a high  $p_T$  track. Events selected by HLT1 were passed on to the second stage of the High Level Trigger, HLT2, for full reconstruction. At this stage, all tracks with a  $p_T > 300 \text{ MeV } c^{-1}$  were reconstructed. The output of HLT2 was stored for offline analysis, in 2012 the output data rate of HLT2 was  $0.35 \text{ GB s}^{-1}$ . The High Level Trigger underwent significant changes through Runs 1 & 2, such as the introduction of the Turbo stream.

### 4.1.4 Turbo Stream

Data storage requirements can be calculated from the product of trigger output bandwidth and experiment running time. In order for LHCb to run at a high output bandwidth, the event size must be minimised to make the most of the limited trigger rate.

$$\text{Bandwidth}(\text{GBs}^{-1}) \propto \text{Trigger rate}(\text{kHz}) \times \text{Average event size}(\text{KB}) \quad (4.1)$$

For Run 2 the *turbo stream* was introduced to improve computing performance, it discards raw data and preserves only high level reconstructed physics objects needed for analysis. Figure 4.2 shows the data flow in Run 2, with a proportion of the HLT2 output heading to the Turbo stream. Only the following information on candidates that passed selection criteria are saved (persisted):

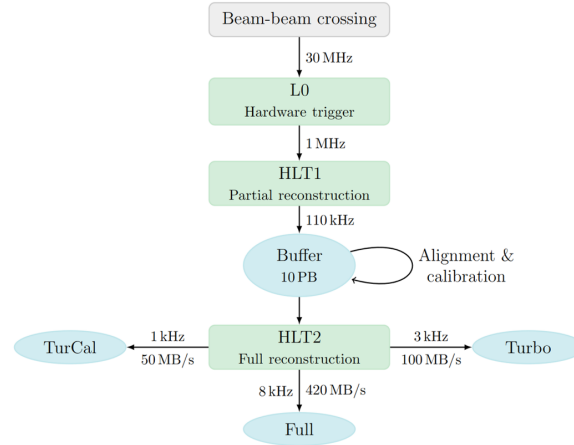


Figure 4.2: Data stream in Run 2 [78].

- All tracks, PID and calorimeter information of objects in decay. And decay vertices.
- All clusters from tracking detectors for further track reconstruction.
- All primary vertices in event, to check for mis-association.

Raw data and other reconstructed objects are not kept. The size of a typical full-stream event is 70 kB, whilst a Turbo stream event can be only 15 kB. A reduced event size enables a broader physics reach because more trigger lines can be studied, in Run 2 almost all charm trigger lines used Turbo stream, in Run 3 this will be expanded to beauty triggers.

Since 2016 full persistence has been possible, meaning all reconstructed objects are additionally kept. This enabled a reduced event size for inclusive decisions. TurboSP (Turbo with Selective Persistence) has been used since 2017 and is a compromise between the flexibility of the Full stream and the data saving of the Turbo stream [78]. Beauty analysis uses mainly inclusive triggers, this means that only a small signature of a process is required to pass the trigger. Therefore extra event information must be selectively persisted for improved offline analysis.

The Turbo stream approach of reducing event size has enabled LHCb to increase physics output within its limited computing resources. By reducing the bandwidth to storage by 50%, more events can be saved and more precise measurements can be made. The percentage of events processed via the Turbo stream will increase in Run 3.

### 4.1.5 Run 3 Trigger

A comprehensive upgrade of the trigger system for Run 3 will significantly increase its ability to select events of interest by running a deeper analysis on every event. The upgrade consists of two key features, a *fully-software trigger*, and a *triggerless readout* [54]. Figure 4.3 shows a comparison between the trigger system in Run 2 and Run 3. The change to a fully-software trigger involves the removal of the Level-0 hardware trigger, which was a major constraint to physics performance. With only basic detector signatures available the L0 trigger had to reduce the event rate to just 1 MHz, this was the greatest inefficiency in the Run 1 & 2 trigger. Removing the L0 trigger will increase the efficiency of much of the physics program by at least a factor a two by increasing the data available to make trigger decisions. For example, VELO tracks will be input into the initial trigger decision for the first time. However this but means that all LHCb sub-detectors must be capable of readout at the full 40 MHz bunch crossing rate of the LHC.

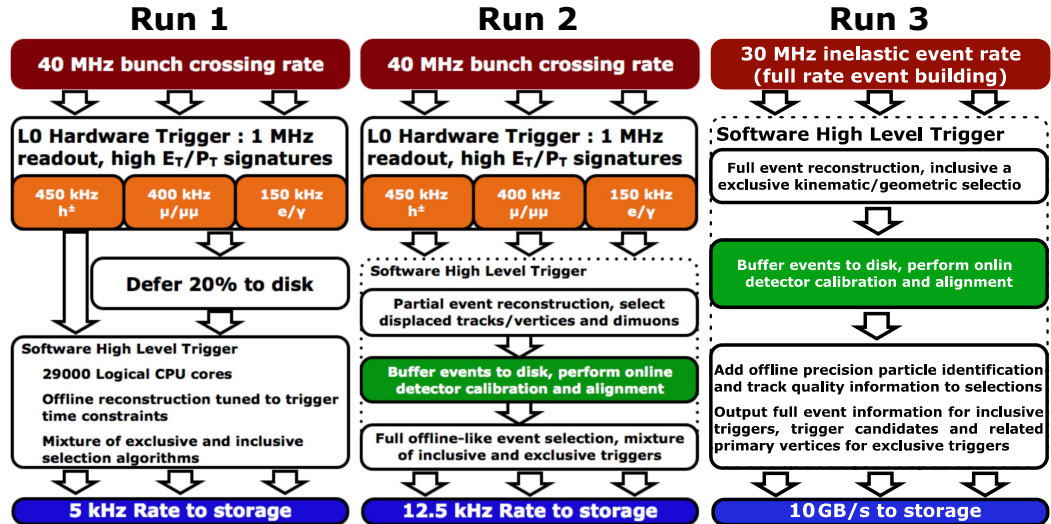


Figure 4.3: Comparison of LHCb trigger schemes between Runs 1, 2 and 3. For Run 3 the L0 hardware trigger is removed. The latest status is that partial reconstruction will occur in HLT1 at real-time, this includes tracking, PV finding and fitting [79].

The change to trigger strategy is required in order to reach the ambitious physics goals of the experiment. In Run 3 the signal rate of charm hadrons will be 1 MHz, and 300 MHz for beauty hadrons. The LHCb trigger is signal dominated, analogous to finding a stalk of hay in a stack of needles. Therefore to achieve high efficiency, a large proportion of events must be studied, and the HLT1 output must be about 2 MHz to capture all interesting physics processes [57].

HLT1 carries out a partial reconstruction of events, and uses a small set of single



and two-track selections to decide whether events should be discarded or sent to HLT2 via a storage buffer [57]. Section 4.1.6 contains details of the implementation of HLT1 with GPUs in the Event Builder. HLT2 is carried out in the Event Filter Farm (EFF), located on-site at point 8. The large data rate flowing into HLT2, up to  $2 \text{ Tbit s}^{-1}$ , means that only a fraction of events can be stored for offline processing. Therefore as much processing as possible must be done in HLT2 itself, such as fully reconstructing all events at this stage. After reconstruction, the majority of events, 68%, will pass through the Turbo scheme, which discards raw event and tracking data and only keeps important high level features and physics objects. During Run 2, the percentage of events processed through the Turbo scheme was much less, 32% [38]. The throughput of HLT2 determines the reduction in event rate required from HLT1.

HLT1 must be able to select the following key detector signatures:

- Tracks or two-track vertices displaced from the primary vertex, these are important signatures of long-lived hadrons and  $\tau$  leptons.
- Leptons, especially muons. Non-displaced leptons are important for spectroscopy studies, exotic searches, and electroweak physics.

To achieve this, HLT1 must be able to:

- Reconstruct all VELO tracks within acceptance and determine primary and secondary vertex locations.
- Reconstruct upstream and long tracks above  $500 \text{ MeV } p_T$ .
- Determine long track momentum to the percent level.
- Return a covariance matrix at the track state closest to the beamline.
- Identify long tracks as muons or non-muons.
- Reconstruct muons from electroweak boson decays.

The global event cut (GEC) rejects the 7% highest occupancy events, which take disproportionate time to process, while reconstruction performance is lower for these events.

HLT1 uses only a few inclusive trigger lines, similar to those that accounted for 95% of trigger rate in Run 2.

- 1-Track: A single displaced track with high  $p_T$ .
- 2-Track: A two-track vertex with significant displacement and  $p_T$ .

- High-pT muon: A single muon with very high pT for electroweak physics.
- Displaced dimuon: A displaced dimuon vertex without a mass requirement.
- High-mass dimuon: A dimuon vertex with mass near or larger than the  $J/\psi$  mass but without displacement requirements.
- Soft dimuon: A displaced dimuon vertex with no transverse momentum but tighter displacement criteria, particularly suited for selecting rare kaon decays.

#### 4.1.6 LHCb Trigger with GPUs

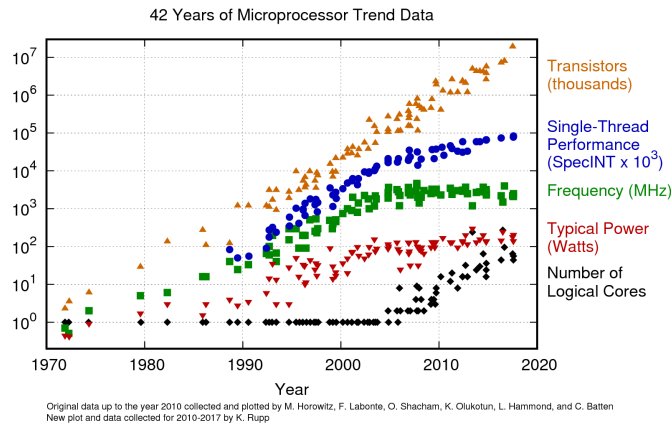


Figure 4.4: General trend in computer processing performance over time, showing how single-thread performance has plateaued in the last decade. This demonstrates the need for parallelism [80].

Trends shown in Fig. 4.4 show that computer processing improvements in recent years have come from parallelism, in the form of multi-threading and multi-core processing, rather than an increase in serial performance or clock speed. There has also been a growth in the use of GPUs to reduce code execution times for less memory intensive applications. Moore’s law, predicting the exponential decrease of transistor size, is still holding, despite predictions otherwise. Where possible particle physics computing systems of the future must exploit parallelism to provide value for money for large increases in computing performance.

To capitalise on recent developments in computing and to provide sufficient processing performance for the LHCb upgrade, a group was established to implement the HLT on GPUs, known as the Allen group [57]. It has now been decided that this will be the baseline for HLT1 in Run 3.

An analysis was made by LHCb in 2021 to decide whether to proceed with either a CPU or hybrid GPU (Allen) trigger strategy [81]. The former option uses CPUs in

the EFF to run HLT1, whilst the latter runs HLT1 on GPUs incorporated into spare slots in the Event Builder. Figure 4.5 describes the two proposed options for HLT data processing schemes.

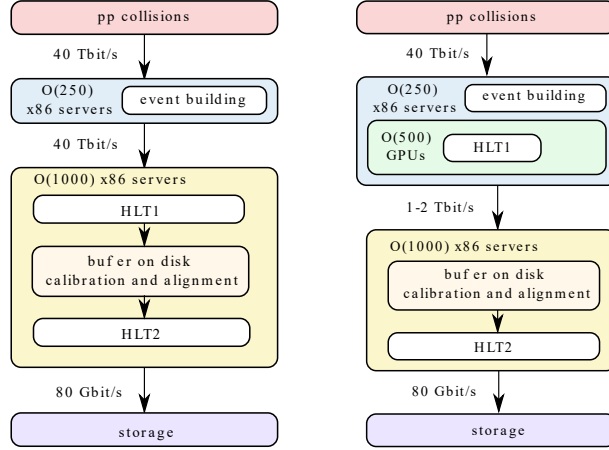


Figure 4.5: HLT data processing scheme options for Run 3. HLT1 will be run on GPUs attached to Event Building PCs, reducing the data rate into the Event Filter Farm from 40 to 1-2 Tbits/s [79].

The physics performance and throughput was measured for each trigger strategy, and little difference could be found between them. Therefore the decision was solely based on a cost-benefit analysis, which found in favour of the hybrid GPU solution. This was due to not only a 20% lower immediate cost, but also the potential for future performance improvements that the next generations of GPUs could offer.

In Allen, the VELO reconstruction algorithm, search-by-triplet, takes 12% of the time budget, compared to 25% for the CPU version. Intra-event and inter-event parallelism will greatly improve throughput of HLT1 with GPUs incorporated into the Event Builder. The small event size of <100 kB means that many events can fit on a GPU with 10 GB memory. Money will also be saved on networking with GPU accelerators implemented in Event Builder, as this will also lead to 30-60 times less data being output from Event Building PCs into the Event Filter Farm. 170 GPUs would be required, each with a throughput of 60 kHz. The LHCb Event Builder would have over 6 PFLOPS of theoretical computing capacity with 2018 GPUs, 10% of this coming from CPUs. Other advantages of using GPUs are that off-the-shelf components can be used, which reduces cost, and there is a big potential to use the resources offline, to run algorithms that rely on parallelism and in the future to train machine learning models.

## 4.2 Track Reconstruction

### 4.2.1 Track Types

Particle tracks in LHCb can be classified into different types, depending on the combination of tracking detectors they are reconstructed in [82]. Examples of these track types are shown in Fig. 4.6.

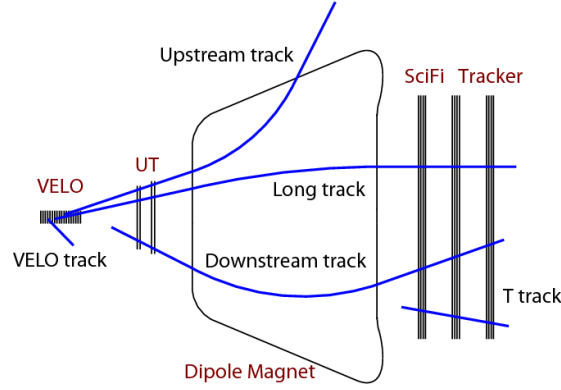


Figure 4.6: Side view of the experiment showing the types of tracks that are reconstructed in LHCb [83].

- **VELO tracks** are reconstructed in the VELO. If within the experiment acceptance, they can be extended with hits from other tracking detectors. Many will have trajectories outside the LHCb acceptance, including backwards tracks, these VELO only tracks are vital for primary vertex reconstruction.
- **Upstream tracks** are reconstructed in the VELO and UT, these are low momentum tracks ejected by the magnet so do not reach the SciFi.
- **Downstream tracks** are reconstructed in the UT and SciFi detectors. These originate from long lived particles that decay outside the VELO such as  $K_S^0$  mesons.
- **T tracks** are only reconstructed in the SciFi, produced from very long lived particles and secondary interactions with detector material.
- **Long tracks** are reconstructed in all three tracking detectors, these have the best momentum resolution and are therefore the most useful for physics analysis.

In LHCb tracks are modelled as parabola, with a 5D track state vector defined along its length,  $\vec{x}$ , shown in Eq. 4.2.

$$\vec{x} = \begin{pmatrix} x \\ y \\ t_x \\ t_y \\ q/p \end{pmatrix} \quad \text{where } t_x = \frac{\partial x}{\partial z} \text{ and } t_y = \frac{\partial y}{\partial z} \quad (4.2)$$

### 4.2.2 Clustering

VELO pixels are only  $55 \mu\text{m} \times 55 \mu\text{m}$  in size. Therefore when a particle passes each detector layer, it can deposit energy in more than one pixel, especially if the particle trajectory has a shallow angle with respect to the sensor plane, as shown in Fig. 4.7. A clustering algorithm combines the connected groups of pixels into clusters which are used by tracking algorithms. The words hit and cluster are used interchangeably, as the distinction is not important for tracking. The clustering algorithm is a focus of work for future experiment upgrades, as it is a time consuming and repetitive task, taking up to 18% of HLT CPU time. For Run 3, VELO clustering will be done on FPGAs, will save approximately 14% of the total trigger time budget [84].

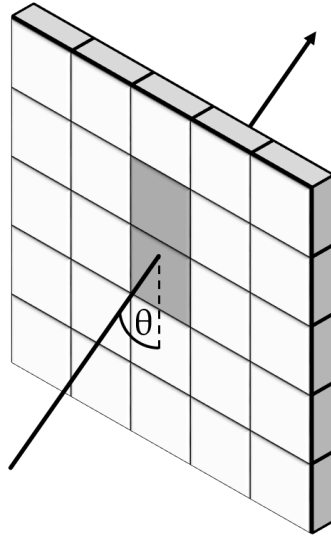


Figure 4.7: Simple diagram of a particle depositing energy in a two pixel cluster as it passes through a sensor at a shallow angle,  $\theta$ .

### 4.2.3 Conventional VELO Pattern Recognition

During Runs 1 and 2 the VELO reconstruction algorithm was based on a sequential process of track seeding and extrapolation. The description of this VELO pattern recognition can be found in [85].

1. Starting from the most downstream module, create track seeds by looking in adjacent modules for pairs of hits with track slopes that point towards the interaction region. This is defined as  $|dx/dz| < 0.4$ ,  $|dy/dz| < 0.4$ , and its purpose is to reduce combinatorics by only searching for tracks that originated near the collision point. When a track seed has been created the hits are marked as *used*.
2. Track seeds are extrapolated into the next module towards the interaction point. The closest hit to the extrapolated position,  $x_p, y_p$ , is added to the track - if the hit is within a search window.
3. If no hits are added, then up to two more modules are searched before the process is declared over and the track is complete.

Track candidates with less than three hits are rejected. In case of tracks comprising only three hits, all hits are required to be unused by other tracks and a cut on the track  $\chi^2$  (obtained from a least-squares straight-line fit) is applied. In case of tracks with more than three hits, at least 50% of the hits are required to be unused, and – if the candidate track passes this cut – all hits on the track are tagged as used.

If the  $z$ -position at which the track is closest to the beam line is larger than the maximal  $z$ -coordinate of all measurements, the track is flagged as a backward track, otherwise it is flagged as a forward track. The tracks are refitted using a Kalman filter [86, 87] with a fixed amount of scattering.

It is clear from this description that the algorithm is based on a serial process, as tracks must be built from one end to the other. A technique where operations can be performed in parallel could lead to a large increase in performance. There is interesting recent work to parallelise these algorithms [88] in the community. There are also a large number of parameters and cuts that require tuning to achieve a good working point, a tracking method that reduces the number of these parameters could be more efficient with less tuning.

#### 4.2.4 Vertex Reconstruction

Knowing the precise location of primary and secondary vertices is crucial to the performance of LHCb. Displaced secondary vertices are indicator of B meson and other heavy-flavour decays and their presence is a important component of the trigger decision in Run 3. Primary Vertex (PV) reconstruction is performed in two steps - seeding, to make an initial estimate of vertex position, and fitting, to find precise vertex positions [89].

A new seeding algorithm for Run 3 works by searching for clusters of tracks which cross the beamline at a similar  $z$ -position. Compressing to the  $z$ -axis only is sufficient

for identifying PV candidates with a high efficiency, Fig. 4.8a shows a density distribution of extrapolated track Points Of Closest Approach  $z_{POCA}$ . Candidate vertices can be easily seen as spikes in the distribution. The seeding is then followed by fitting.

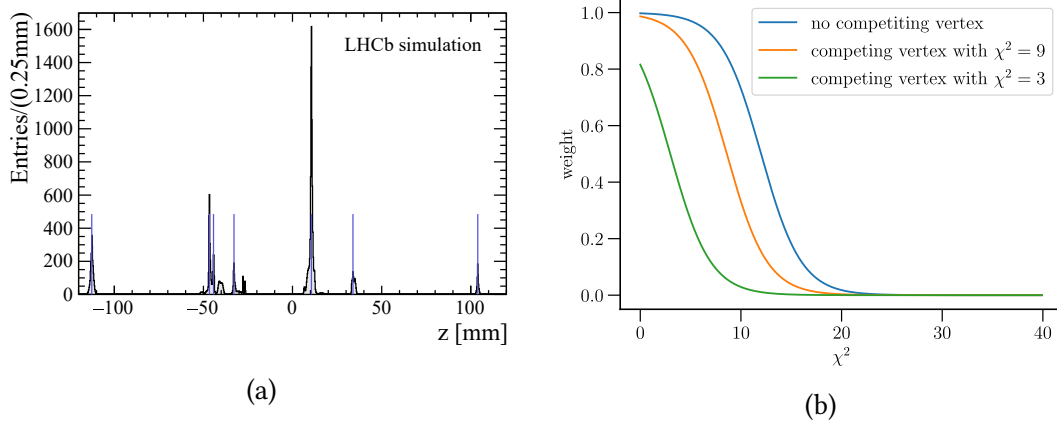


Figure 4.8: a) Histogram used for Primary Vertex seeding. Blue vertical lines denote the peaks in the density distribution which are the likely locations of primary vertices. b) In the vertex fitting process, the weight of a track with respect to a vertex depends on the  $\chi^2$  between the track and vertex, and the presence of competing vertices. [89].

Fitting is performed by a *Adaptive Multi-Vertex* process [90]. Tracks are assigned a weighted contribution to each nearby candidate vertex fit, each track can contribute to multiple primary vertex fits. The function to calculate the weights is shown in Fig. 4.8b. An iterative process of fitting and recomputing each tracks contribution is repeated until convergence.

#### 4.2.5 UT and Forward Tracking

VELO tracks can be extended into VeloUT tracks by adding hits in the Upstream Tracker sub-detector [91]. There is fringe magnetic field here of 0.15 Tm, so track momentum can be estimated with a resolution of 15-25%. To extend tracks, all VELO tracks consistent with the acceptance of the UT are extrapolated as straight lines towards the UT. For each VELO track there can be many candidate VeloUT tracks because of the long distance between the detectors and the fringe magnetic field, a track fit determines the most likely combination. If the  $\chi^2$  of the track fit is below a threshold, then the UT hits are added to the track.

VeloUT tracks can be from low momentum particles that are bent out of acceptance by the magnet, or tracks of high-momentum particles that can be extrapolated in the Scintillating Fibre Tracker (SciFi) to build long tracks. Using VeloUT tracks simplifies the search for long tracks by reducing the number of tracks input to the

forward tracking algorithm. Forward tracking is the extension of tracks further by adding hits in the SciFi. Tracks containing hits in all three tracking detectors are called long tracks and have the best spatial and momentum resolution [92].

A search window is opened for hits in the SciFi by modelling the particle course as a straight trajectory, before receiving a kick from the magnet, changing its course to a different straight path through the SciFi. Because the sign of the charge of particles is not known, the search window must take account of tracks being bent in either direction by the magnet. A track-fitting step minimises the number of ghost tracks.

#### **4.2.6 Detector Alignment and Calibration**

Tracking algorithms rely on the position of the physical detectors being known to an accuracy better than their hit resolution. A well-aligned detector will have better tracking efficiency, vertex resolution and mass resolution. A process of frequent alignment and calibration ensures this [93].

All tracking sub-detectors are aligned at the start of every fill. This is especially important for the VELO, because its two halves move in and out each fill. Within each VELO half, alignment is achieved by studying tracks with large numbers of hits, while tracks which traverse both halves - or pass through the small overlap region between halves - are used to align the halves. During Runs 1 & 2, the alignment of the VELO was found to be very good, with the relative movement of modules within the hit resolution of the sensors.

Many other factors can cause misalignment, such as temperature fluctuations, the reversal of magnet polarity and the physical disturbance for maintenance. Taking account of the magnet is important for the UT and SciFi detectors which are closest and are physically moved by the strong field. Since 2018 alignment and calibration is done automatically online in real-time, taking only a few minutes for complete for the tracking system [94].





---

# Machine Learning

---

*I believe that in about fifty years' time it will be possible to programme computers, with a storage capacity of about  $10^9$  bits, to make them play the imitation game so well that an average interrogator will not have more than 70 per cent, chance of making the right identification after five minutes of questioning. Alan Turing.*

TURING may have been disappointed to learn that 70 years after his prediction, most requests to my phone for the weather forecast seem come back with, “Sorry I didn’t understand, please try again”. Nevertheless advances in learning algorithms have enabled incredible achievements. Whether it is the ability of American supermarket chain Target to predict customer pregnancies, sometimes even before the knowledge of the women’s families, in order to send coupons for baby products [95] or identifying breast cancer from scans more effectively than doctors, by the reduction of false negative and false positives [96]. It is clear that machine learning permeates all aspects of our modern lives. In this chapter I will provide a brief history of machine learning followed by theory and applications both within and outside high energy physics.

## 5.1 History of Machine Learning

Alan Turing asked the question “Can machines think?” in his seminal work *Computing Machinery and Intelligence*[97]. At this time the seeds of artificial intelligence were being sown. Seven years prior, in 1943, a collaboration between neuroscientist Warren McCulloch and logician Walter Pitts produced the McCulloch-Pitts neuron [98]. They attempted to mathematically describe the “Nets of neurons” of the nervous system. The McCulloch-Pitts (MP) neuron is very simple - it sums Boolean inputs and outputs a Boolean value depending on whether the sum of the inputs reached a threshold value. Despite its simplicity, the MP neuron can model linearly separable logical functions (OR, AND, etc...). However this model has limitations. Non-linear functions like XOR cannot be modelled, all inputs have the same weight and it cannot handle non-binary inputs.

The *perceptron* was invented in 1957 by Frank Rosenblatt, it improved upon the MP neuron and would become the basic unit of neural networks[99]. Figure 5.1 shows the diagram of a perceptron drawn by Rosenblatt in his paper. The perceptron added individual weights to each input - which could now be any real number - and a constant bias term. The perceptron turned out to be less useful than hoped, it could still not model non-linear functions. Importantly however, the perceptron was accompanied by an algorithm for learning. This algorithm could adjust the weights of the input to reduce the error between the output and truth labels.

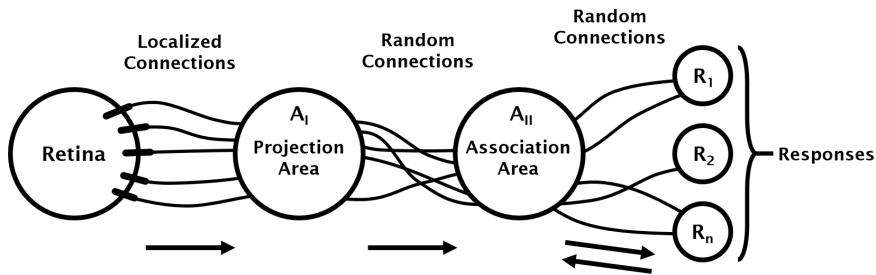


Figure 5.1: Remake of Perceptron diagram from Rosenblatt’s original paper. [99]

The so called *XOR problem* was solved by the introduction of an extra layer of perceptron units between the input and output, the hidden layer. In the 1980s it was found that multi-layer neural networks were very effective at modelling a variety of non-linear functions [100]. Ending a period of doubt known as the AI winter, a new era of excitement over learning algorithms began.

However the history of machine learning is not necessarily tied to the electronic computer, as the Machine Educable Noughts And Crosses Engine, or MENACE showed [101]. Produced by Donald Michie, a British computer scientist who helped break the

German Tunny code during the Second World War. The machine consists of 304 match boxes, each containing coloured beads, there is one match box for each possible state of the noughts and crosses. By playing games against a human, the machine can learn to play the game. By adding or removing beads from boxes that causes wins or losses respectively, the machine can learn very quickly learn to beat human players.

## 5.2 Why use Machine Learning in High Energy Physics?

Traditional Bayesian statistical methods (classification, hypothesis testing, regression, goodness-of-fit testing) use a model  $p(x|\theta)$  describing the probability of observing  $x$  given physics theory parameters  $\theta$ . However  $p(x|\theta)$  cannot be known explicitly in high-dimensional high-energy physics datasets. Simulation is used to estimate the statistical model, however this is only practical with a small number of dimensions, since the number of samples needed increases with the power of the number of dimensions. The VELO alone for example has  $10^7$  sensor channels. Therefore physicists must reduce the number of dimensions at various scales by forming particle hits into tracks, showers and jets - then using these higher level representations to identify particles types and estimate energy and momentum - finally producing event level summaries. Individual events can then selected for analysis. Selections at each step are traditionally based on feature engineering from physics knowledge, such as the shape of showers and cuts on energy deposited in calorimeters. Deep learning could remove this feature engineering allowing subtleties of new physics processes to shine through, which may be lost in the sub-optimal dimensionality reduction of conventional algorithms. The role of Machine Learning in HEP is to improve this dimension reduction to distil more information from raw data into low dimensional high-level objects [102].

A simulated dataset:  $\{x_i\}_{i=1}^n$ , can be thought of as a number of independent samples from the underlying statistical distribution given simulation parameters,  $p(x|\theta)$ . The aim of the simulation is to approximate the probability distribution from an enormous space of unobserved processes. Equation 5.1 describes the statistical model, the probability of observing  $x$  given  $\theta$ . The ground truth of the simulation is represented by  $z$ ; which specifies all aspects of the simulation, from particle momentum to detector interactions.

$$p(x|\theta) = \int p(x, z|\theta) dz \quad (5.1)$$

The goal of reconstruction algorithms is to estimate the ground truth  $z$  from the observed detector data  $x$ . The simulation provides a truth dataset:  $\{x_i, z_i\}_{i=1}^N$ , that can be used to test the effectiveness of reconstruction algorithms, and also to train learning algorithms.

The reduction of high dimensional inputs to a low dimensional output is exactly the kind of problem machine learning algorithms were built to solve. For example taking an image of millions of individual pixels and distilling their features to a simple object label. Or more formally; the process of constructing a function that maps input to output, whilst optimising a metric of our choice by minimising a loss function. Machine learning algorithms can be seen as the natural solution to many problems in high energy physics.

$$\text{Require function } f : X \rightarrow Y \text{ to minimise } L[y, f(x)] \quad (5.2)$$

An ideal learning algorithm would create a function that optimises the loss function over all possible values of  $(x, y)$ . However in reality training data is sampled from a statistical distribution, with a finite number of model parameters to adjust to perform the optimisation. One must be careful to not overfit, that is to over-optimize to training data, creating a model that does not generalise. Given the large numbers of model parameters, finding a truly optimum solution can be difficult, if not impossible. However methods such as regularisation can reduce the effects of overfitting, and empirical evidence suggests that neural networks are good at finding solutions more optimal than human engineered features could do, even if the perfect answer is not found.

The machine learning models cannot however function on their own. Experience and knowledge of machine learning and domain are required for learning algorithms to succeed. There are *hyper-parameters*, the switches and dials that must be adjusted and tuned correctly. Concerning neural networks - the choice of activation and loss functions, depth and breadth of the network, choice of output representation must all be considered carefully in order to best suit the problem trying to be solved. This is before the optimisation of training through the learning rate and length of training are considered.

There are also many different approaches to learning, developed to allow machines to solve a wide variety of problems. The technique I describe most in this chapter is **supervised learning**, where a model makes predictions that are compared

to the known ground truth, adjusting the models weights to steer its predictions closer and closer to the correct answer. Conversely, **unsupervised learning** does not use labelled training data. The model must learn to recognise patterns in data that distinguish different classes. The ability to train without labelled data is a big advantage for domains without large, labelled data sets, but also creates challenges when trying to assess performance. **Adversarial learning** is a fascinating concept; two competing neural networks attempt to outwit each other. One network making predictions as best as it can, whilst the second tries to tell the difference between prediction and truth - much like the cat and mouse of a forger attempting to fool the art specialist.

## 5.3 Types of Machine Learning Algorithm

### 5.3.1 Supervised Learning

The hybrid model described in Chap. 6 of this thesis uses supervised learning. It is probably the most intuitive learning system to understand as it parallels the ways in which humans might learn a new skill. A child learning to read or write may guess wildly to start with, but with supervision from a teacher to correct mistakes and with plenty of time to practise, the child's brain will develop until they can generalise their knowledge to pronounce words they have never seen before. This is exactly the method that a neural network uses so that it can differentiate images into categories or connect the hits of particles traversing a physics experiment. The mathematics are explained in Sec. 5.6 but a qualitative description is useful for understanding.

The basis for supervised learning are large, labelled datasets. In physics large and accurate simulations create datasets with ground truth labels. Labels might be the content of an image ('dog', 'cat') or the identity, momentum and decay history of a particle. Given the available features fed in, a learning algorithm generates a prediction that is compared to the label. The loss function calculates how close the prediction is to the label, called the error. A simple loss function is the mean squared error. The weights and parameters of the connections in the learning algorithm can then be adjusted proportionally to the error to slowly bring the predictions in line with the truth labels.

It is therefore important that training data is as representative as possible. A learning algorithm only knows what it is taught. A person who learnt English will not understand French without a comprehensive retraining. The world is a messy place and the predictions of learning algorithms will only ever be predictions based on available evidence, an unusual pronunciation may throw a speaker the same way



Figure 5.2: Learning to ride a bicycle takes trial and error, each time we fail our brains develop to improve our ability, much like how a neural network *learns* by supervised learning. [103]

unseen data can be confused by a learning algorithm.

The main disadvantage of supervised learning is need for labelled training data. In many situations this may be very difficult to find, either because data labels are not recorded, or cannot be recorded. For example if data was never collected with machine learning in mind, or if confidentiality concerns mean individual labelling of people in a population is not possible.

### 5.3.2 Unsupervised Learning

Supervised learning is not the only method of machine learning. Rather than checking the label of each training example, unsupervised learning relies on learning the intrinsic properties of groups within a dataset. Only after training can the labels be checked to calculate the accuracy, if labels exist at all. Clustering is a form of unsupervised learning, it means to divide a dataset into groups. Simple clustering algorithms such as K-means work to split a dataset into a pre-determined number ( $k$ ) clusters through an iterative process. First by initialising  $k$  centroids, which will become the centre of each cluster. In the first iterative step data points are assigned to their nearest centroid using euclidean distance as a metric, then a new centroid is calculated as the mean of points assigned to each original cluster and these two steps repeated. When the distance between old and new centroids reduces below a threshold the algorithm stops. The drawbacks of K-means are that the number of clusters must be known, it assumes clusters are convex and isotropic (so does not work well

with unusually shaped clusters) and that it does not scale well in higher dimensions.

A more sophisticated clustering algorithm is DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [104]. It defines clusters as areas of high density separated by areas of low density. There is no need to specify a number of clusters, so is more flexible than k-means and it works with arbitrary shaped clusters, not just convex and isotropic. Sec. 6.4.3 describes my attempt to use DBSCAN for VELO tracking.

A fascinating example of unsupervised learning is the ability to learn a depth map from just the video footage taken from front facing camera in a car [105], shown in Fig. 5.3. This opens the possibility of removing lidar or radar from self-driving cars, reducing the cost and complexity of hardware required. However possibly even more importantly is the way in which the neural network learnt to do this task. Using a metric of *consistency* allowed it to learn unsupervised, as this is a error metric which is not related to any ground truth, but simply the patterns in the data itself.

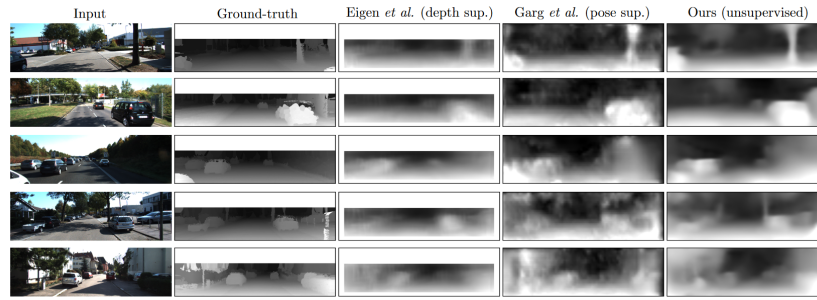


Figure 5.3: A comparison of methods. Input video on left, then ground truth depth map, two supervised learning methods and finally the output of the unsupervised learning algorithm on the right [105]

If data is not labelled then how do we evaluate the performance of a trained model? In this case there are data sets of labelled depth map video that can be used for testing purposes. However in other situations a more qualitative analysis might be required, for example the judgement of human surveyors.

## 5.4 Applications of Machine Learning

### 5.4.1 Current use in High Energy Physics

Machine learning techniques have been used for many years in high energy physics, and have been instrumental in important discoveries. As discussed above, machine learning algorithms provide a natural solution to many problems in high energy physics. Models such as Boosted Decision Trees (BDTs) which often use a small



number of physicist-engineered features are commonly used to discriminate signal from background in analyses, such as this LHCb analysis [106] that uses only 12 input variables. BDTs have mostly replaced *cut* based methods for signal selection and background discrimination due to their ability to separate data with non-linear boundaries.

The Higgs boson search at CMS revolved around the decay of the Higgs to two photons, creating a slight excess in the diphoton mass spectrum. Multiple BDTs were used, one to improve the diphoton mass resolution in order to reject events with artificial photon signals, another trained with a well understood simulation to reject background diphoton events, and finally one to categorise signal events based on signal purity. The increase in sensitivity from using BDTs was equivalent to collecting 50% more data [11]. Also at CMS, Fig. 5.4 shows how a BDT was used to provide energy corrections to improve the resolution of the electromagnetic calorimeter. BDTs can also be used online, meaning fast analysis done in real-time whilst the experiment is running. Data storage constraints mean events can only be stored if deemed interesting for future offline analysis. Therefore online data selection plays an important role in any high energy physics experiment. For the first two years of data taking at LHCb, a BDT was used as the main component in the High Level Trigger. Compared to a cut-based approach the BDT provided much better signal-background discrimination, whilst also being stable under changing detector conditions [107].

There has also been a move towards deep neural networks (DNNs). DNNs comprise many layers and often using a larger number of less engineered features to give even better results than 'shallow' neural networks and BDTs. Relaxing the need to carefully specify features could reduce human workload, and creates the possibility of automatically discovering new and powerful feature combinations. In a pioneering paper of 2014, benchmarks showed that DNNs could outperform shallow neural networks and BDTs by 8% [108].

The Deep Underground Neutrino Experiment (DUNE) will use a Time Projection Chamber to detect neutrinos. The experimental setup produces event images which are naturally suited for particle identification with Convolutional Neural Networks (CNNs). These are a type of neural network that work by passing filters over an image to extract its features such as edges, corners and shapes, and are the go to machine learning algorithm for image analysis. Applying this concept to simulated events the CNN can reliably identify 25 possible interactions, most importantly the classification of neutrino flavour as electron, muon or tau. Approximately 90% efficiency and purity is achieved for lepton flavour classification, critical for maximising the experiments sensitivity to CP-violating effects [109].

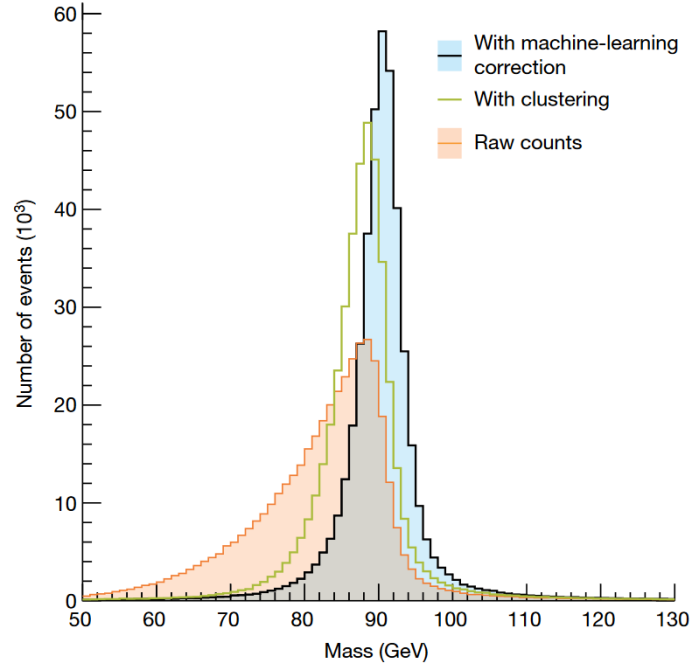


Figure 5.4: The Z boson distribution to electron-positron pairs at CMS, showing raw detector data (orange), after clustering data (green) and after corrections learnt by a machine learning algorithm [11].

### 5.4.2 Future Applications in High Energy Physics

Despite the many and varied uses of machine learning in particle physics, there are areas which until recently have not felt its benefits. Particle tracking algorithms have until now relied on Kalman filter type methods as described in Sec. 4.2. These algorithms work quickly and are very effective, however this may not be the case in future experiments. High data rates from real time readouts will engulf conventional tracking algorithms.

The International Conference on Computing in High-Energy and Nuclear Physics (CHEP) is a showcase for new computing ideas in physics. A wide range of ideas were presented at CHEP 2021, proposing new applications for machine learning in particle physics. Graph Neural Networks (GNNs) are being proposed as a future tracking algorithm [111]. GNNs work by learning the weights of connections of a connected graph structure, which can be thought of as analogous to the connections of hits in a particle track. The performance of machine learning based event classification models can be improved by splitting one large ML model into multiple sub-tasks [112]. Fast simulation with a graph variational autoencoder (GAN) is over 400 times faster than using Monte Carlo simulation without a reduction in accuracy [113]. Figure 5.5 shows an LHCb calorimeter image, a Convolutional Neural Network (CNN) was

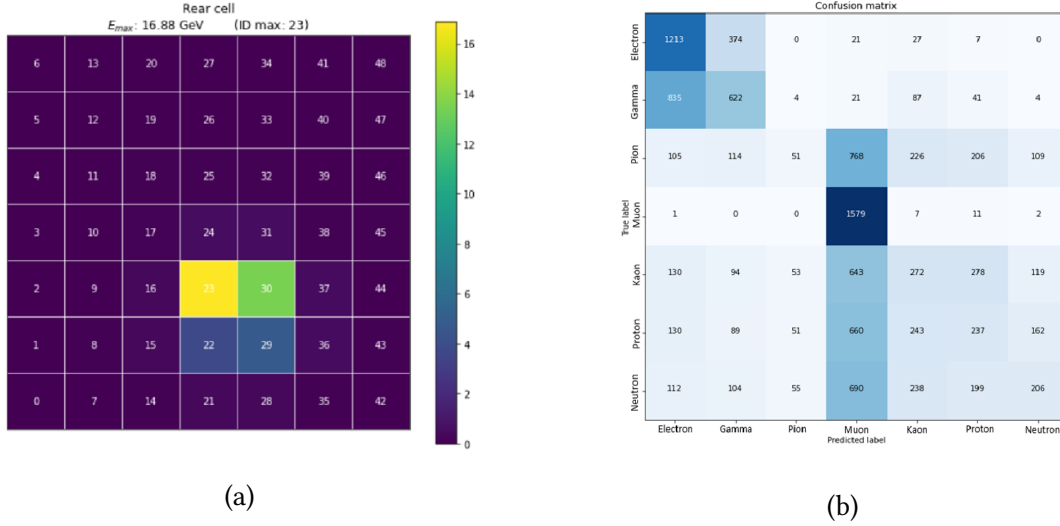


Figure 5.5: a) Calorimeter image input of 7x7 pixels. b) Output confusion matrix when classifying 7 different particle types. CNN outperforms a tuned decision tree called XGBoost. [110]

used to perform particle identification based on the shape of these energy deposits [110]. An AI can monitor experiment data by classifying images of plots [114], rather than analysing the underlying data itself. This significantly reduces the workload of shift staff and lessens the chance of human error, using plot images also makes it easier to transfer to other experiments. The main challenge is the human labelling of thousands of example images for the training process.

## 5.5 Why Choose a Neural Network for VELO Tracking?

Machine learning has many applications within and outside of high-energy physics as demonstrated above, with many different reasons why machine learning can be preferable. The primary challenge facing modern high-energy particle physics experiments is achieving data throughput and processing requirements. Once trained, neural networks can have very fast execution times, especially if the architecture of the neural network itself can be implemented on specialist hardware such as an FPGA or ASIC. This could reduce costs for experiments by reducing hardware requirements. Another advantage is that a general, machine learning tracking algorithm could be applied to different experiments. Transfer learning is the process of taking a part-trained neural network and using a partial-retraining process to suit a new application. Many particle detectors are similar to the VELO, i.e. planes of sensors. This

can forego the needs to create bespoke tracking algorithms for every experiment. A neural network may also be simply better at the tracking problem. A simple Kalman filter based algorithm may miss subtle detector effects that a neural network could learn.

However there are also disadvantages of using machine learning for tracking. It can be difficult to understand how machine learning algorithms come to their decisions, leading them to be called *black boxes*. It can also be challenging to calculate errors. Neural networks requires time and resources to train. Large datasets are often needed for a successful training process – accurate detector simulation in this case - which may not always be available. Training can be computationally expensive and take a long time, and differences between training and real-world data can reduce performance, for example if a detector simulation is not accurate. Neural networks can be difficult to train, with many hyper-parameters that need tuning for best performance. Though this is a larger issue for conventional tracking algorithms, which also require the tuning of many parameters, such as the size of search windows. Once trained, the efficiency of an ideal machine learning tracking algorithm could be tuned by adjusting only one parameter, the output threshold of a neural network.

## 5.6 Neural Networks and How They Work

Machine learning is a broad term describing an array of algorithms designed to learn from data. An example is the neural network, shown in Fig. 5.6 is a minimal example. Neural networks are graphs of nodes and edges. Each node holds a value, and each edge a weight. The flow of data through a neural network can be understood as:

1. The value of a node is multiplied by the weight of an edge traversed.
2. After each node an activation function scales the value appropriately, common activation functions are ReLU and Sigmoid functions.
3. When all values have been processed, the output of the network is compared with a truth label to calculate an error.
4. The weights of edges and biases in the network are adjusted to reduce the error.
5. Iterate process until the difference between prediction and truth is minimised and the output of the network is accurate.

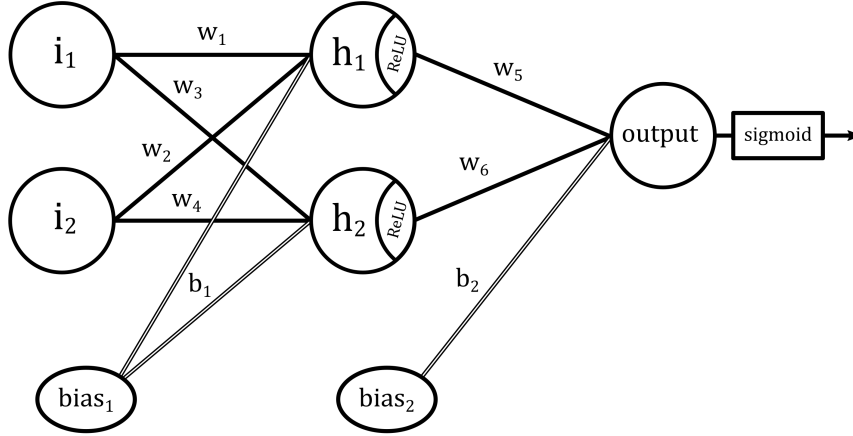


Figure 5.6: A diagram of a minimal neural network. The network has two input variables, a single hidden layer and a single output. Circles are nodes, connected by edges. Nodes have values, edges have weights. ReLU and sigmoid are activation functions.

### 5.6.1 Mathematics of Neural Networks

The weights of the edges of a neural network are often initialised with random values, how are these values adjusted to teach the the network to give us the outputs we want? The mathematical algorithm that achieves this is called *back-propagation*. It takes advantage of the derivatives of the activation functions to determine how much and in which direction weights should be changed.

Consider the simple neural network shown in Fig. 5.6. It contains two input nodes, two hidden layers of two nodes each, and one output node. There are also biases applied at each hidden layer. A weight is randomly assigned to each edge initially. In this example we will use the network to produce a probability output, between 0 and 1, based on two input variables. This could represent for example, telling the difference between an image of a cat or a dog. The equations below apply to the simple model shown above, but can easily generalise to any size of neural network.

#### Forward Pass

The first step is to calculate the initial output of the neural network, one layer at a time, starting with the output of the hidden layer. Neural networks are often initialised with random node and weight values. The value of each hidden node is the sum of the input node values multiplied by the edge weights, plus the bias value.

$$h_{net_1} = i_1 w_1 + i_2 w_2 + b_1 \quad (5.3)$$

$$h_{net_2} = i_1 w_3 + i_2 w_4 + b_1 \quad (5.4)$$

After each hidden node, an activation function acts on the value to apply a non-linear scaling, this is important as it allows the neural network to act as a non-linear function. A popular choice for hidden layers is the Rectified Linear Unit (ReLU) function which turns the output to 0 for negative node values.

$$ReLU(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$$

$$h_n = ReLU(h_{net_n}) \quad (5.5)$$

Now we can calculate the output value of the final layer using the output values of the hidden layer in the same way that hidden node values were calculated.

$$out_{net} = h_1w_5 + h_2w_6 + b_2 \quad (5.6)$$

It is common for neural networks to have an output in the range 0 to 1, for example if the neural network is generating a probability or in binary classification. For this reason a *sigmoid* activation function is used to scale the output value between 0 and 1. A plot of the sigmoid and ReLU activations functions is shown in Figure 5.7, along with their derivatives. The reason these particular function is used will become clear when the derivative is calculated in back-propagation process.

$$sigmoid(x) = \frac{1}{1 + e^{-x}}$$

$$out = sigmoid(out_{net}) \quad (5.7)$$

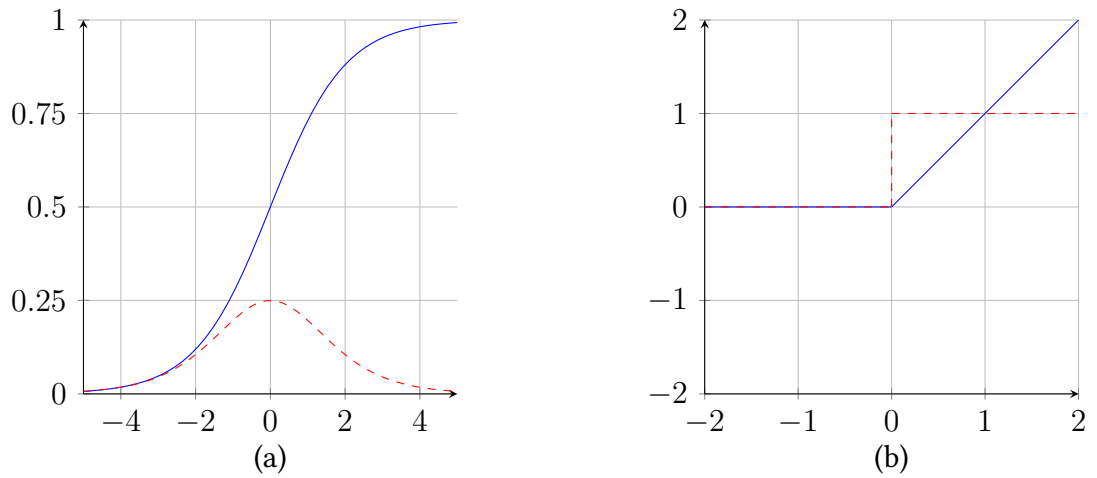


Figure 5.7: Blue solid lines are a) Sigmoid and b) ReLU activation functions and their derivatives as red dashed lines.

Now that we have calculated the initial output of our network, we must compare it to the known truth label. This is done with an error function. A common error function is the *mean squared error* (MSE).

$$MSE = \sum \frac{1}{n} (label - out)^2 \quad (5.8)$$

### Backward pass

Now it is time for the most interesting step, back-propagation itself, where the learning occurs. We must calculate how the error is affected by the value of each weight, so that the weights can be adjusted to minimise the error. Firstly, how does the error depend on each weight in the final layer?

$$\text{Change in error wrt } w_5 = \frac{\partial E}{\partial w_5} \quad (5.9)$$

This can be expanded using the chain rule to a product of partial derivatives.

$$\frac{\partial E}{\partial w_5} = \frac{\partial E}{\partial out} \frac{\partial out}{\partial out_{net}} \frac{\partial out_{net}}{\partial w_5} \quad (5.10)$$

All the elements of this expression can be calculated relatively easily due to the choice of activation and error functions. The first relates the error value to the output. Therefore we must calculate the partial derivative of the error function.

$$E = \frac{1}{2} (label - out)^2$$

With the substitution  $u = (label - out)$

$$\begin{aligned} \frac{\partial E}{\partial out} &= \frac{\partial E}{\partial u} \frac{\partial u}{\partial out} \\ &= (2 \frac{1}{2} u) (0 - 1) \\ &= out - label \end{aligned} \quad (5.11)$$

The second expression of Eq. 5.10 links the output before and after the sigmoid

activation function.

$$\begin{aligned} out &= \frac{1}{1 + e^{-out_{net}}} \\ \frac{\partial out}{\partial out_{net}} &= out(1 - out) \end{aligned} \quad (5.12)$$

The final component of Eq. 5.10 is how the output before the sigmoid activation function depends on the edge weight.

$$\begin{aligned} out_{net} &= h_1 w_5 + h_2 w_6 + b_2 \\ \frac{\partial out_{net}}{\partial w_5} &= h_1 \end{aligned} \quad (5.13)$$

We now have a very simple expression to describe how a change in final layer weights effects the error value, Eq. 5.14.

$$\frac{\partial E}{\partial w_5} = (out - label)out(1 - out)h_1 \quad (5.14)$$

We can use this to adjust the weights by an amount proportional to the size of the error. In order to minimise the error, the change is subtracted from the initial weight value. There is a similar expression regarding all other weights. The updated weight is calculated using the following Eq. 5.15. The learning rate is a hyper-parameter that must be chosen by the user, it determines how granular the learning process is.

$$w_5^+ = w_5 - (\text{learning rate})(out - label)out(1 - out)h_1 \quad (5.15)$$

Using the updated final layer weights, the same process is used to update the first layer weights. The expressions will actually be more simple due to the derivative of the ReLU function being just 0 or 1. After this process, with the same input values, the error will be smaller. The process of changing weight values to reduce the error is called *optimisation*, Eq. 5.15 is a very simple optimiser. More sophisticated optimisers, such as *Adam*[115], are generally used. After many iterations of this learning process called epochs, the outputs of the network should converge on the truth labels.



## 5.7 Six Month Industry Placement at OnTrac

The Liverpool Big Data Science Centre for Doctoral Training (LIV.DAT CDT)<sup>1</sup> is a program to provide PhD students with training in data science and careers, supported by STFC<sup>2</sup>. The LIV.DAT CDT training program included a six month placement, with complete freedom to find a data science related company or institute. I have always had a fascination with railways, so this was an obvious area to look into. I was sure that data science would play a big role in many railway sectors, and it could lead to a future career path. I emailed OnTrac Ltd, based in Gateshead [116], and they were excited to have me join the team as they wanted to improve how they handled their data and trusted I could help them. They explained that their data sources were fragmented, making it difficult to analyse and collaborate. This meant that their valuable data was not utilised to its full potential. They had ambitions to incorporate machine learning into their data analysis, something my PhD had given me experience with. I worked specifically in the context of track worker safety, which is sadly one remaining area of the railway industry where serious incidents still occur. Tragically two track workers were killed in July 2019 in Margam, South Wales. The investigation by the RAIB<sup>3</sup> [117] found that a significant cause of the incident was that the work was not planned properly, leading to a misunderstanding that left the workers in a dangerous situation, thinking they were protected from moving trains when they were not. It is therefore vitally important that work is planned safely.



Figure 5.8: Track workers undertake maintenance on the line, these are the people OnTrac works to protect. [118]

<sup>1</sup><https://www.liverpool.ac.uk/livdat>

<sup>2</sup>Science and Technology Facilities Council

<sup>3</sup>Rail Accident Investigation Board

OnTrac produce a system known as Safe Work Planning (SWP) to enable planners to correctly generate document packs which are used by track workers. Packs include details for track workers: such as the location of work, where they will access the railway, what work will be undertaken and what method of protection is used. The method of protection is how the workers will be protected from trains; there is a range of options ranging from closing the line to providing a lookout to provide warning of approaching trains. The SWP system has been running for about 10 years and has produced a wealth of data that OnTrac asked me to look at. It is worth noting that in the accident at Margam mentioned above, they were not using the OnTrac SWP system, but a separate system produced by Network Rail, the national rail infrastructure owner.

There were a lot of ideas about how this ‘big data’ could be used to improve the product for OnTrac customers, and of course increase safety. For example enabling planners from different contractors to collaborate to share line closures, which could reduce the reliance on lookout warning methods. Another idea was a learning algorithm to identify dangerous locations or working practices. Finding patterns in data which are not currently analysed. However the largest difficulty is data management, supervised learning algorithms require large quantities of labelled training data. This requires careful reporting of safety incidents, and structured datasets. The quantity of data is also an issue. In particle physics vast quantities of data can be created through simulation. However serious safety incidents on the railway are few and far between, and therefore provide a limited dataset with which to train a learning algorithm.

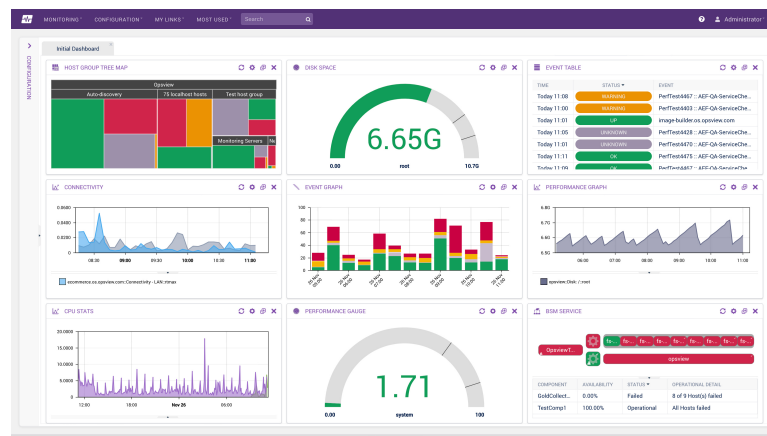


Figure 5.9: Example of a Microsoft Power BI data dashboard [119].

Working with my manager, David Smith, we created a dashboard using Microsoft Power BI to visualise the data. An example of a dashboard is shown in Fig. 5.9, for confidentiality reasons I cannot show mine. The first challenge was to access the data itself, as it is stored in an online database not designed for the task we wanted to

do. Therefore we downloaded a subset of data of about one month, giving a sample of approximately 10,000 packs to work with. I wrote a python script to clean and organise the data, as well as run some analysis such as grouping free-text fields in categories. This could then be imported into Power BI. The dashboard could be used by a contractor to understand the packs that their planners are creating, to make sure they are up to standard and safe. Currently this is a manual and time consuming job. I also created a tool that could help planners to avoid conflicts with other packs, by informing them of packs planned at the same time in the same location, that might also be using the same access points. Ideas from my work was incorporated into a larger project by OnTrac to improve the SWP system. The placement was a great experience of a different and fascinating work environment, despite Covid-19 pandemic disruption. I was inspired to search for careers in rail, and will start work as a guard on Merseyrail trains in Liverpool after graduation.

---

# A New Hybrid Model for Tracking

---

THE next generation of particle detectors will produce unprecedented volumes of data at rates not seen before. Processing this data in a high-throughput environment will become a major challenge, such as the need for real-time VELO tracking at LHCb for Run 3 and beyond. In this chapter I will describe the tracking problem itself in Sec. 6.1, the requirements for VELO tracking in Run 3 in Sec. 6.2 and the challenges these create in Sec. 6.3. In Sec. 6.4 I will introduce the initial work I undertook to learn about tracking, different types of machine learning algorithm and how these can work together. The hybrid tracking model I developed is described in Sec. 6.5, with details of each step in the tracking process and results showing the tracking performance that the algorithm achieved compared to the conventional tracking approach. I spent time characterising the robustness of the algorithm to small detector misalignments, and measured how much faster the neural network could run on a GPU compared to on a CPU. In Sec. 6.6 I discuss possible improvements that could be made to the hybrid algorithm to improve performance such as the use of graph neural networks, as well as showing results from a retraining of the fully connected neural network used in the hybrid model.

## 6.1 The Tracking Problem

Tracking is the process of identifying the paths that individual particles take through the detector. In Silicon pixel tracking detectors, when a particle passes through a detector layer, it deposits energy in the pixels, this is called a hit. A tracking algorithm must be able to connect the hits from each particle together correctly to generate tracks for each particle. Figure 6.1 shows a view of a simulated event in the upgrade VELO detector.

At LHCb the VELO tracking must be completed in the first stage of event processing, VELO tracks are then joined with tracks from the UT and SciFi tracking detectors. There is no magnetic field in the VELO so particles follow straight trajectories. However tracks do not all originate from the same point due to the spread in primary vertex location, and tracks from secondary vertices are offset further. Hits are most densely located near the interaction region, they include hits from particles with very high transverse momentum which fly outside the acceptance of the detector. With full VELO tracking in the first stage of the new High Level Trigger, this step must be done in real-time, whilst also giving time for the rest of the HLT1 algorithms to run before the next collision. VELO tracking is the most time consuming process in the algorithm chain due to the density and complexity of collision events and the high spacial resolution of the silicon sensors.

## 6.2 Velo Tracking Requirements

### 6.2.1 Efficiency

Particles trajectories in LHCb originate in the VELO, therefore track reconstruction at this level is vital for the reconstruction of physics processes. The VELO tracking algorithm must have good efficiency, meaning it correctly reconstructs a very high percentage of particle tracks. It must do this whilst minimising the number of incorrect and duplicate tracks, called ghost and clone tracks respectively. Reconstruction of simulated Run 3 events provide a baseline for new tracking algorithms. Efficiency for VELO tracks with momentum more than 5 GeV should be 99%, and higher for long tracks, with a ghost and clone rate of 2.5% and 1.0% respectively. Some comparisons between the tracking requirements of Runs 2 & 3 are shown in Fig. 6.3. Furthermore this should be robust to the number and position of primary vertices in a collision event and to the angle of tracks [37].

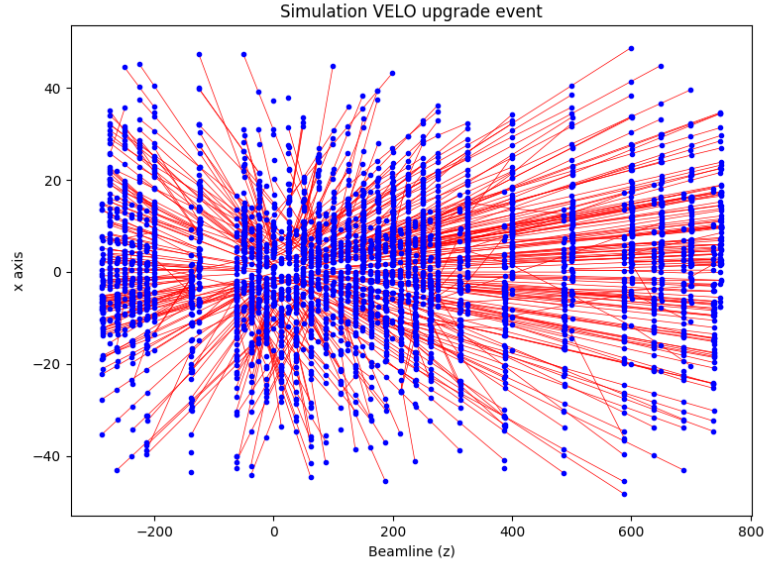


Figure 6.1: View in the x-z plane of a simulated collision event in the VELO. The zero point of the z-axis is the approximate proton collision point in LHCb, the downstream direction is the positive z-direction. Blue points are hits in the detector, red lines are tracks connecting the hits of each particle, drawn using the simulation truth information. This data was used to train tracking algorithms.

### 6.2.2 Timing

The tracking algorithm must be able to be executed in real-time at up to 40 MHz, as it is an important step in the first stage of the LHCb trigger. In the future, as more experiments move to a real-time analysis framework, the importance of fast tracking algorithms will only increase.

## 6.3 Current Challenges Facing Tracking Systems

Almost half the time taken to run the LHCb HLT1 is devoted to VELO tracking only, therefore any increase in the speed of the tracking algorithm would make a significant impact on the trigger system, including lowering cost from reduced hardware requirements or an increase in the reach of the physics program. Future experiments such as the High Luminosity LHC (HL-LHC) will exacerbate these issues. There will be many more collisions per second, and many more hits in each event.

There has been increased effort to develop new tracking algorithms, often utilising machine learning techniques. For example the TrackML challenge was a competition to find the best new methods, both in terms of tracking performance and speed [120]. Current tracking methods are bespoke to each experiment, but what if algorithms

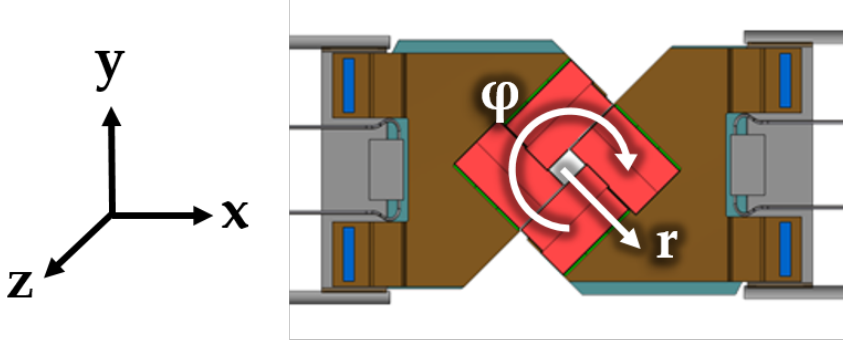


Figure 6.2: Coordinate frame of the VELO. The  $z$ -axis is along the beamline, with the  $x$ -axis horizontal and the  $y$ -axis vertical. The angle around the  $z$ -axis is measured with  $\phi$ ,  $r$  is the radial distance from the beam.

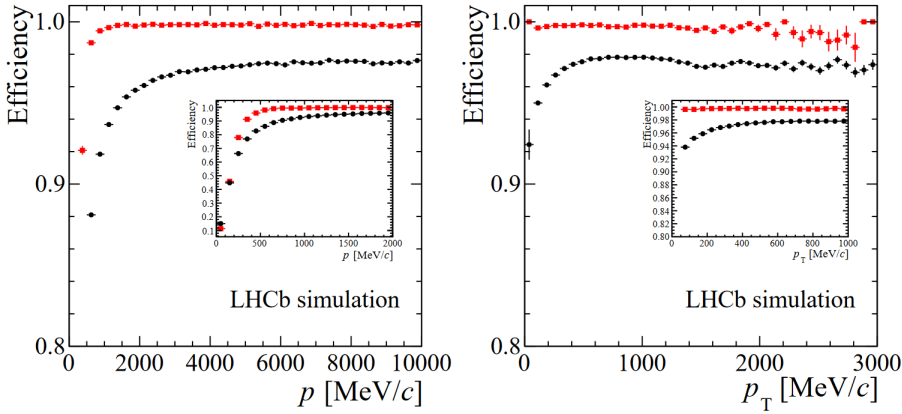


Figure 6.3: Velo tracking efficiency requirements with respect to particle momentum and transverse momentum, taken from the VELO upgrade TDR. Black points are original VELO, red points are upgraded VELO [37]. These plots illustrate the high standards any tracking algorithm must achieve.

could be shared and adapted to multiple experiments? Transfer learning is a machine learning technique that allows pre-trained neural networks to be quickly re-trained and adapted to new environments. Many experiments use detectors formed of a series of regular shaped detector planes, so tracking algorithms should be similar. Transfer learning has the potential to speed up software development for new detectors and upgrades.

## 6.4 Initial Machine Learning Tracking Investigations

This section describes the initial work I undertook in my first year. This work helped me to understand the tracking problem, machine learning techniques and the simulated dataset of VELO events, and also improve my coding skills,. I started with

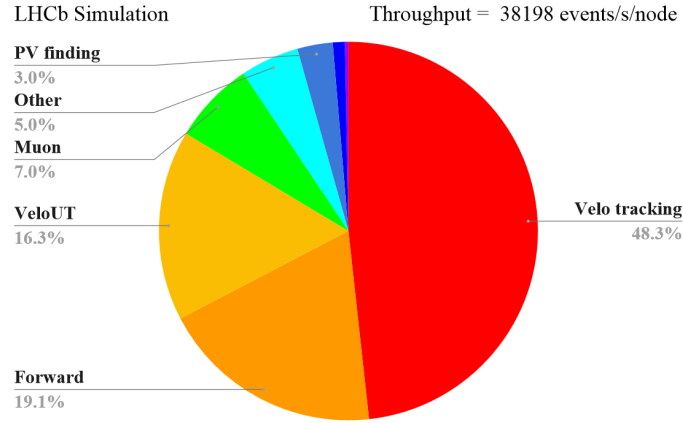


Figure 6.4: Pie chart of HLT1 processing time. Almost half of the time taken to run HLT1 on CPU is devoted to VELO tracking [2]. This is partly due to the complexity of VELO tracking, but also due to VELO tracks being an input to the first stage of the trigger system. The trigger decisions made using VELO tracks means that other tracking algorithms, further along the trigger chain, process fewer events.

very basic problems that were easy to understand. I experimented with toy models, which are simplified datasets and situations. For example generating single tracks or a small number of tracks in 2D. From there I could move onto using the proper simulated VELO data, which can be broken down into small sets of data, such as only using hits from two adjacent layers.

I experimented with different machine learning techniques to understand how they worked and in what context they worked best. For example trying both supervised and unsupervised machine learning methods, and developing knowledge of how hyper-parameters effect training and performance of neural networks. I also experimented with how to represent hits, either as points with coordinates, or as pixels in an image.

#### 6.4.1 Joining Closest Points and Building Tracks

To begin my investigations into using machine learning for tracking, I started with the most basic problem I could think of, and worked to increase the complexity at each step to move closer to a realistic tracking problem.

Instead of beginning by trying to connect many hits into tracks, I asked whether a neural network could successfully identify the closest pair of points in a 2D space. The inputs to a simple fully connected neural network were the 2D coordinates of a small number of points. I generated a large number of training samples and used



these to train the neural network to output a vector of 1's and 0's with a length equal to the number of points. A correct output is a 1 in the position of the two closest points, and zeros in every other position. The neural network was generally capable of correctly identifying the closest pair - doing this 70% of the time - from a set of 10 randomly placed points. An example where the neural network failed to join the correct pair, and a histogram of results, is shown in Fig. 6.5.

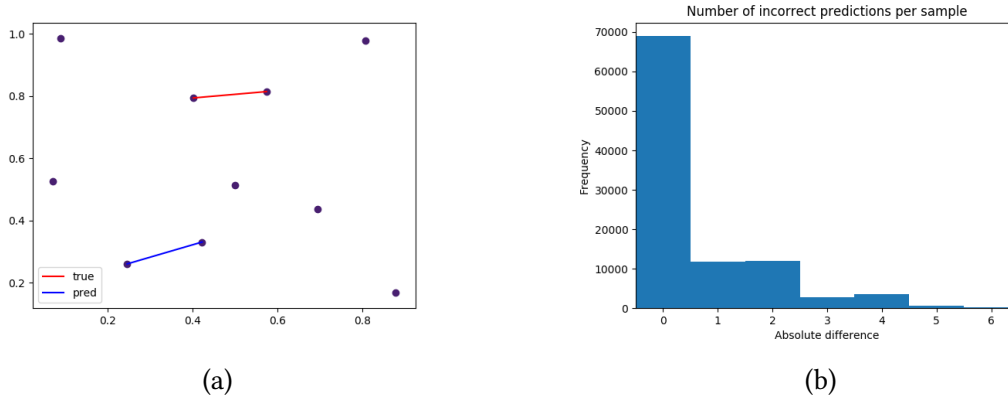


Figure 6.5: Results of using a neural network to connect closest points. a) Example of where the neural network failed to identify the closest pair, but was close. b) Histogram of results for 100,000 samples. Number of incorrect predictions in output vector, 0 means correct pair (69.2%), 1 means one hit correct (11.7%), 2 means wrong pair (11.9%), 3 or more means more than one pair connected (7.2%).

The next question was how could I scale this up to connect hits on a track? I still used a random set of 10 2D points, this time connected into a track by connecting closest hits together in a chain, shown in Fig. 6.6a. I trained the neural network to output an adjacency matrix, which is a way of showing which pairs of hits are connected. After training, the neural network was tested on track-like patterns of hits, arranged in straight lines with some approximate scattering effect, shown in Fig. 6.6b. For one or two distinct tracks this method works quite well, the neural network could connect the hits correctly.

However there were significant issues with this method that would make it difficult to scale further. This method would only work for a fixed number of hits, and I only tried with a small number of 2D points. The neural network could be confused if tracks crossed or passed close to each other. Significantly, adjacency matrices scale with the number of points squared, this means that for a large number of points the adjacency matrix becomes unmanageably large. The output is also very sparse, meaning almost all elements of the adjacency matrix are zeros. When training the neural network will have a tendency to predict a completely zero adjacency matrix, since this results in an almost minimal loss function value.

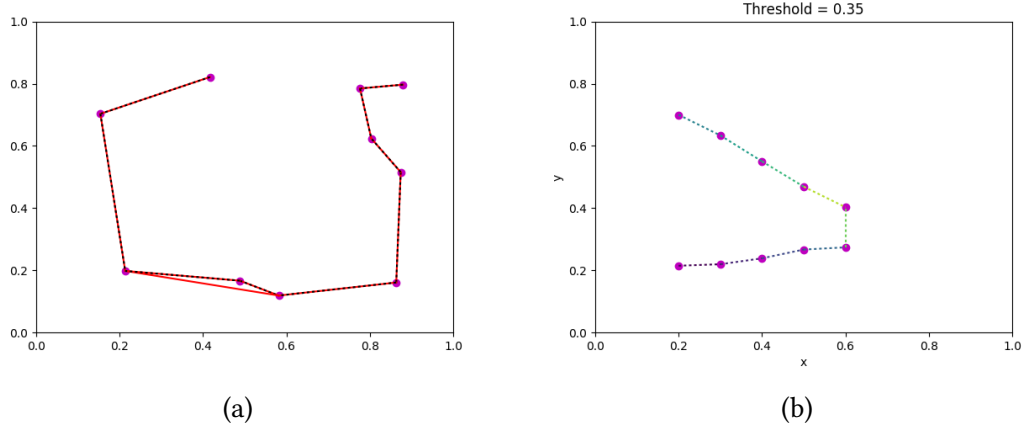


Figure 6.6: Results of toy model, using a neural network to connect hits into tracks. a) Random training sample of hits. b) Testing sample of hits arranged into two tracks, with a simulated small amount of scattering. The neural network successfully connected the hits in this case, generalising what it had learnt on random points. However it also connected the close ends of each track, as to the neural network these are just another close pair of hits.

## 6.4.2 Predicting Hits in Adjacent Modules

### Detector Layers as Images

Convolutional Neural Network (CNN) are a type of neural network designed to work with images. Intuitively a pixel detector layer can be thought of as an image with each detector pixel representing an image pixel. The intensity of an image pixel can represent the state of each detector pixel, for example a value for 0 for the sensor area, changing to 255 for a pixel hit by a particle, and a grey elsewhere. These detector layer images could be used by a CNN for tracking. This was my first attempt to use VELO simulated data for tracking, until now I had only used toy models. To reduce complexity, I only used data from one half of the VELO. I used an input image of one detector layer, and tried to generate an image of the next detector layer. However this was not a success.

CNNs are used for tracking in particle physics experiments, however in this case it was not the right choice. A CNN tracking system works well when complete tracks can be captured in a single image, such as in a TPC detector. CNNs work by finding shapes and textures in images. In the VELO where tracks span multiple images of sparse hits, I found the technique to be limited.

### Hits as Coordinates

Representing hits as pixels in an image did not work for this problem, so I instead started to look at hits as points with coordinates. This is also the way that hits are

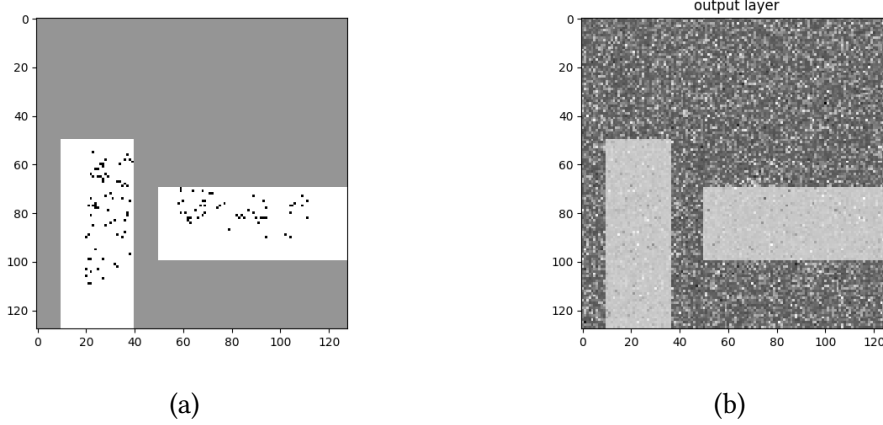


Figure 6.7: Results of predicting hits in adjacent layers whilst representing the detector as an image. a) Input image to the Convolutional Neural Network (CNN), white areas are sensors, black pixels are hits, grey area is non-sensor areas. b) Output should be prediction of hits in the next layer, however results were noisy.

described in the VELO simulation dataset. In previous investigations I had applied different machine learning methods experimentally, now I used domain knowledge to focus on a better solution. These coordinates could be used as the input to a fully connected neural network, rather than the CNN used previously. The task of the neural network would be to take hits on one module as inputs, and output the coordinates of hits on the adjacent layer, and then compare these to the true coordinates. If this worked then it would be evidence that tracking with machine learning could be possible. This also allowed me to experiment with the parameters of the neural network. For example changing the size and shape of the neural network, and using different training batch sizes, loss functions and optimisers.

A challenge for any machine learning tracking algorithm is that there are a different number of hits in each layer of the detector, and in each event. This does not fit well into most neural network models, which have fixed input and output sizes. Therefore, the input to this neural network was simply the coordinates of a single hit, and the output was another set of coordinates for a hit in the adjacent module. This meant that it would work with any number of input hits. The modules chosen were the two furthest from the interaction point, which are the sparsest. Figure 6.8 shows an example output of this method. Black crosses show the input hits, blue crosses are the true hits in the next layer, and red crosses are the neural network prediction. By inspecting the plots it can be seen that the neural network did a good job at predicting the location of hits on the next layer. This gave me evidence that a neural network could be a promising candidate for a tracking algorithm.

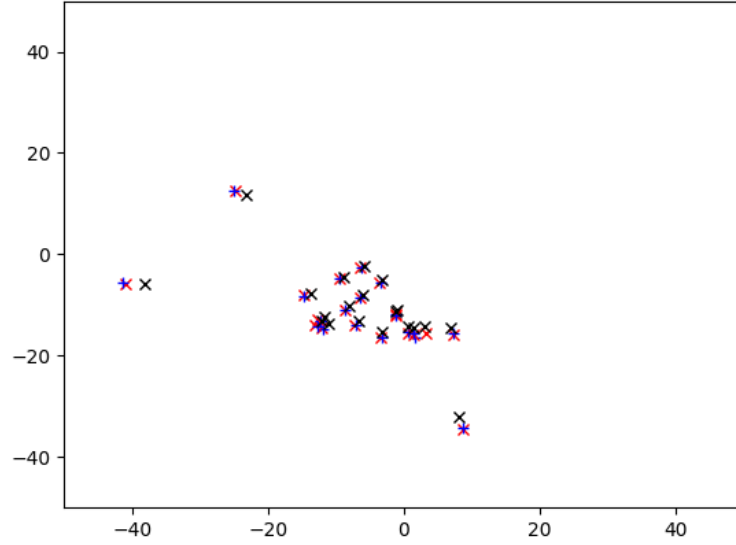


Figure 6.8: Results of predicting hits in adjacent layers, whilst representing hits as coordinates. Input hit coordinates are black crosses, true next layer coordinates are blue crosses and next layer predictions are red crosses. The neural network is good at predicting the location of hits on these two downstream modules, this can be seen as the red crosses fall close to the blue crosses.

### 6.4.3 Unsupervised Learning

After experimenting with supervised learning techniques, I investigated the use of unsupervised learning for tracking. Clustering algorithms are a type of unsupervised learning, as discussed in Sec. 5.3.2; they are used to separate datasets into groups. For example grouping a retailer’s customers based on their shopping habits. When considering the tracking problem, a clustering algorithm could be used to group hits into tracks. This should not be confused with the clustering of pixels in the VELO, explained in Sec. 4.2.2. In this section, I use the word *clusters* to mean groups of hits joined by the clustering algorithm, i.e. tracks. The first challenge is to find a projection of the dataset where hits from each track group closely together, then a clustering algorithm can be used to identify each cluster of hits. I transformed the hits into the  $(\eta, \phi)$  coordinate plane. In these coordinates, all hits on straight tracks from the origin should group closely together. A popular clustering algorithm is called Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [121], which works by grouping together high density regions of data points.

Figure 6.9 shows an image of a simulated event. The lower plot shows hits coloured by their true track\_id. The upper plot shows the same hits after clustering using the DBSCAN algorithm, with one colour per cluster. The upper plots shows only tracks

of more than three hits. It can be seen that there are fewer, larger clusters in the upper plot, showing that DBSCAN is grouping hits from multiple tracks. Figure 6.10 shows histograms of the results over 1000 minimum bias simulated events. These confirm that DBSCAN produces fewer and larger clusters. Figure 6.10a shows the Adjusted Mutual Information Score (AMIS), a popular metric to quantify the effectiveness of clustering. A perfect clustering gives a score of 1, a random clustering gives a of 0. The mean AMIS over 5000 events was  $\text{AMIS} = 0.301 \pm 0.001$ , demonstrating that the clustering did not perform well.

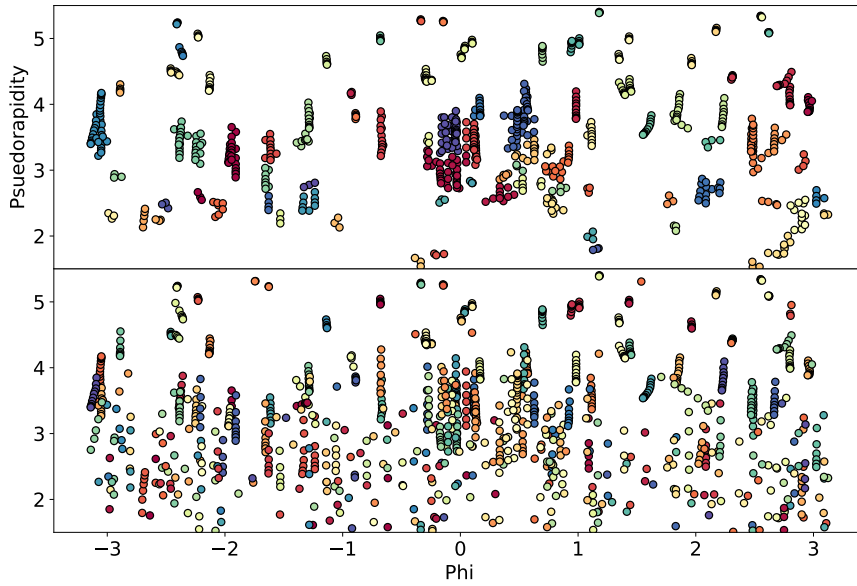


Figure 6.9: Results of using DBSCAN for VELO tracking for simulated tracks covering the forward acceptance. The lower plot shows hits coloured by their true track\_id. The upper plot shows the same hits after clustering using the DBSCAN algorithm, with one colour per cluster. The (pseudorapidity,  $\phi$ ) phase space is used to transform the hits coordinates. This space is used because it should group hits from each track together, especially if they originate near the proton collision point.

There are a number of reasons that this method was not a success. Firstly, there is no projection of the hit coordinates that neatly separates the tracks, due to some tracks not pointing to the origin point. This means that hits from each track are spread out and overlap, which makes them very difficult for the clustering algorithm to identify as tracks. Secondly there are only a small number of hits in each track. Clustering algorithms work by identifying areas of higher density and there are simply not enough hits in a track to create large dense groups. From the results of these initial investigations into tracking algorithms, I decided that supervised learning methods were the most promising.

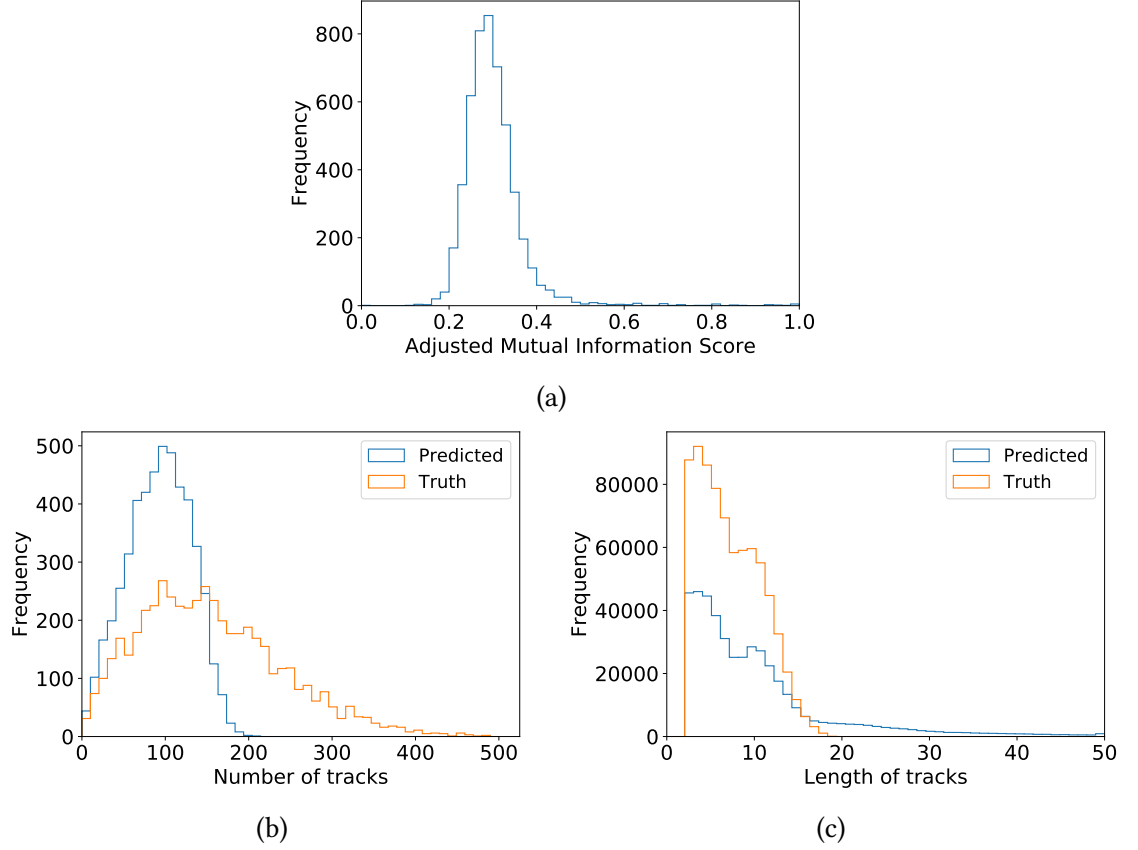


Figure 6.10: Histograms of results of DBSCAN clustering of VELO hits from 5000 events. For tracks in the forward acceptance and with at least 3 hits per track. a) Adjusted Mutual Information Score is a popular clustering metric, which varies from 0 to 1. This was measured to be  $AMIS = 0.301 \pm 0.001$ . b) The mean number of tracks in each event was predicted at 93.0, compared to a truth value of 154.6. c) The mean length of tracks predicted by the clustering was 11.5, whilst the truth value was 7.6. The distribution of predicted track length has a large tail not visible on the histogram. These results show that DBSCAN produced less tracks per event, each containing more hits. DBSCAN joins hits from multiple tracks.

## 6.5 Hybrid Tracking Method and Results

### 6.5.1 Initial Development of the Hybrid Model

The origins of the hybrid VELO tracking algorithm are in work done by Kurt Rinnert<sup>1</sup> and Marco Cristoforetti<sup>2</sup> and presented at CHEP<sup>3</sup> 2018 [122]. One reason the VELO tracking problem may be well suited to initial investigations into using machine learning is because the lack of a magnetic field in the detector, meaning that tracks are not-curved. They developed a hybrid tracking algorithm, meaning that it replaced only selected stages of the track reconstruction pipeline with Machine

<sup>1</sup>University of Liverpool, UK

<sup>2</sup>FBK, Trento, Italy

<sup>3</sup>International Conference on Computing in High-Energy and Nuclear Physics

Learning. The algorithm, known as *TrackNN*, used a fully connected neural network to output probabilities that pairs of hits were connected, these pairs were called doublets. Working back from the furthest detector module, the track doublets with the greatest probability in each module were joined together. The track ended when no more doublets reached a probability threshold. This method achieved high efficiency in the outer regions of the detector, however it became less effective closer to the interaction region. Table 6.1 shows the tracking performance of the TrackNN algorithm for different sectors of the detector. I wanted to improve upon this algorithm to work more effectively near the interaction region, and to characterise its performance.

Algorithm	Efficiency %	Ghost %	Clone %
Conventional	98.9	2.5	1.0
TrackNN, module 50 $\rightarrow$ 30	98.0	1.6	0.8
TrackNN, module 50 $\rightarrow$ 20	93.2	3.1	0.9
TrackNN, module 50 $\rightarrow$ 0	71.1	3.9	0.7

Table 6.1: Table of results for original TrackNN algorithm, for tracks passing the following requirements: At least 3 clusters in the VELO,  $p > 5$  GeV,  $2 < \eta < 5$ , particle originates from interaction region. Module 50 is the furthest upstream from the interaction region [122]. Definitions of performance metrics are given in 6.5.7.

## 6.5.2 Overview of My Revised Hybrid Model

Data from the VELO is in the form of zero-suppressed  $(x, y, z)$  coordinates for each hit, in this approach the coordinate system is moved to  $(r, \phi, z)$ , where  $r$  is the radius of a hit from the  $z$ -axis and  $\phi$  is the angle on the  $xy$ -plane. The algorithm I developed is comprised of three main steps and is shown in Fig. 6.11:

1. The first stage is a fully connected neural network which inputs the hits of adjacent layers, and outputs the probability that a seed hit should be connected to each target hit on the next layer of the detector. Connections between hits on adjacent layers are called doublets. Any doublet with a probability more than a threshold move forward to step 2.
2. The second stage filters the doublets. The  $\phi$  and  $r$  gradient wrt  $z$  of each doublet is calculated. Where more than one doublet from each  $z$ -direction connects to a hit, only the 2 doublets with the most similar  $r$  gradients are kept. Where all doublets connected to a hit are from the same  $z$ -direction, the doublet with the smallest  $\phi$  gradient is kept. When projected in  $\phi$ , doublets from tracks that point to the origin have a  $\phi$  gradient of zero.

3. The final step is to connect adjoining doublets into tracks and then join track fragments split by the algorithm. These are known tracks are known as clone tracks in LHCb.

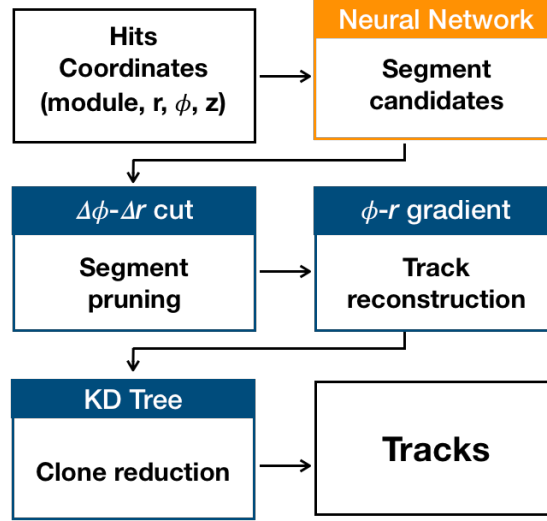


Figure 6.11: The three stages of the hybrid reconstruction algorithm: 1) **Neural Network**: Machine Learning based doublet generation connecting hits on adjacent modules. 2) **Post processing**: Pruning of the doublets with  $r$  and  $\phi$  gradient criteria. 3) **Clone Killing**: Reduction of the clones.

### 6.5.3 Find Doublet Candidates with a Neural Network

The task for the neural network is to return the probability that two hits on subsequent modules belong to the same track. A neural network acts as a function, with inputs and outputs. In this situation, the inputs are hit-coordinates and the outputs are the probability that input hits are connected into pairs called doublets. Doublets are track seeds that are joined into full tracks later in the algorithm. The input is a seed hit and several target hits. The seed hit is the origin of the doublet on the module further from the interaction point (module-1). Target hits are possible hits in the doublet on the adjacent module, closer to the interaction point (module-2). A diagram showing a possible arrangements of seed and target hits is shown in Fig. 6.12.

The initial model, TrackNN, and presented at CHEP 2018 [122] considered all hits in module-2 as target hits. To fit the rigid input size of a neural network, the target hits were batched into groups of four. Therefore the neural network had to be run once for each batch. With the final batch containing 'padding hits' to maintain the correct input shape for all batches.

I wanted to use a method that did not involve batching and padding. This is because running the neural network fewer times will increase the speed of the algo-



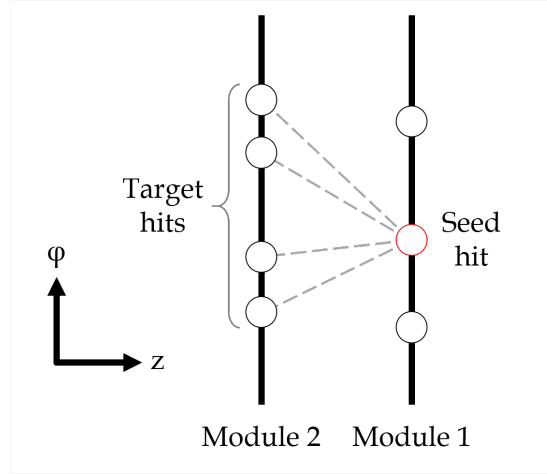


Figure 6.12: Diagram of one seed and four target hits on adjacent modules, linked by possible doublets. The positive  $z$ -direction is away from the interaction point.

rithm. I also wanted to use the fewest number of target hits possible, in order to reduce the size of the neural network. I found that the 4 target hits closest in  $\phi$  were the most useful. Figure 6.13 shows that in the vast majority of cases the correct choice of target hit is the nearest neighbour in  $\phi$  to the seed hit. Also, 98.7% of true target hits are within the four nearest neighbours in  $\phi$ . So the method I chose only uses the closest four hits to the seed hits in  $\phi$  as the input target hits.

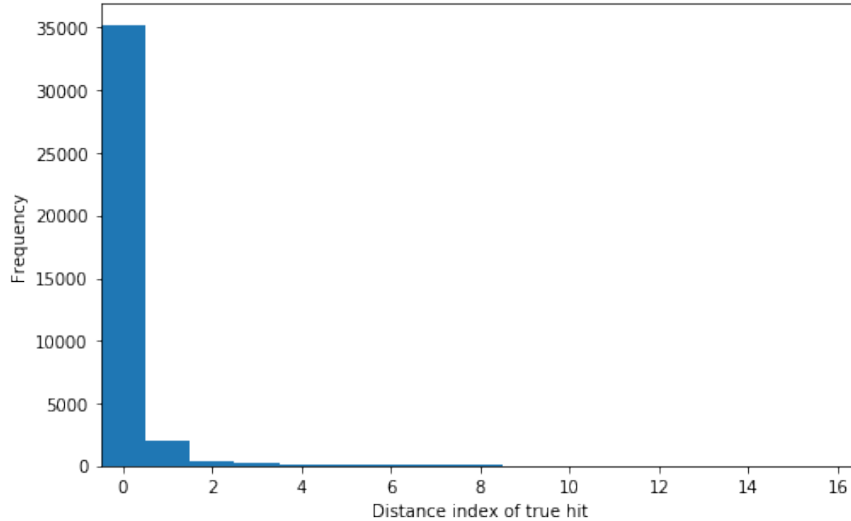


Figure 6.13: Distance Index. Many but not all true doublets have hits on adjacent modules that are nearest neighbours in  $\phi$  (91.9%). Therefore only connecting closest hits in  $\phi$  is not a good strategy. However 98.7% of true doublets have hits that are within the nearest 4 neighbours in  $\phi$ . Thus using only the four closest target hits should not affect efficiency and should increase speed.

I also experimented with using a different number of closest target hits. Figure 6.14 shows that using 8 or 12 closest target hits gives a better neural network output. It is a trade off between a smaller neural network and a faster execution time. Having a larger number of target hits also creates a challenge when a module contains a very small number of hits, padding would again be required in this case.

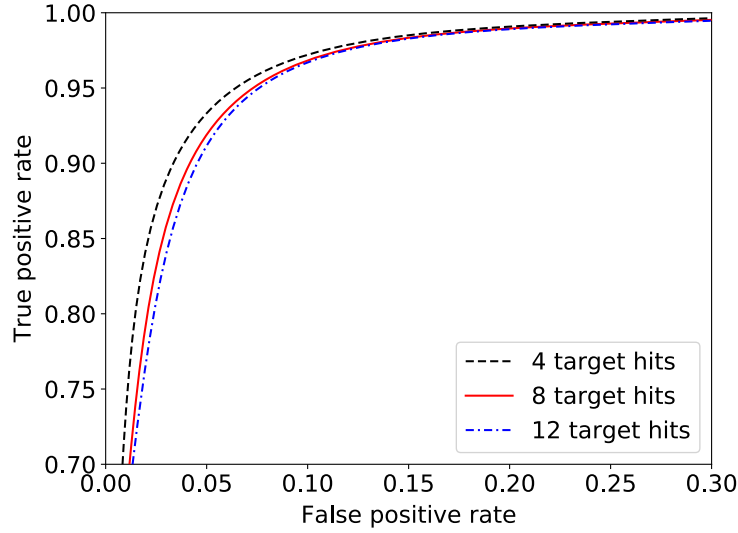


Figure 6.14: Neural network performance for 1000 events. This ROC curve shows results after a cut on the doublets produced by the neural network, requiring  $|\Delta\phi| < 0.0003$  and  $|\Delta r| < 1$ . Using the 4 closest hits gives the best performance, with an area under the curve of 0.985, with 0.982 and 0.981 for 8 and 12 closest hits respectively.

Of course, calculating the closest target hits to the seed hit takes time to compute. However, it is worth it if this time outweighs the extra time taken to run neural network multiple times. Finding the  $n$ -closest numbers is a common problem in computer science and there are different ways to do it. The time complexity for an optimised search is  $O(m \log(k))$ , where  $m$  is the number of target hits on a module and  $k$  is the number of closest hits required. This also requires the values to be sorted first.

In order to reduce the time taken to execute neural network inference, the number of layers and the size of each layer in the network should be as small as possible. The neural network has 10 inputs and 4 output values, composed of just four fully connected layers of dimension (10, 129), (129, 130), (130, 65), (65, 4), with a ReLU activation function after the three hidden layers and a sigmoid activation after the output. A diagram of the neural network is shown in Fig. 6.15. This neural network layout is the same as used in the TrackNN model. I did not make any changes because the number and depth of hidden layers had been previously optimised for the similar problem, also because small changes in these parameters have little effect

of performance. The ReLU activation function is commonly used in fully connected neural networks because it is a very simple and effective non-linear function, with a very simple derivative. The equations for ReLU and sigmoid activation functions are shown in Eqs. 6.1 & 6.2.

$$ReLU(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases} \quad (6.1)$$

$$Sigmoid(x) = \frac{1}{1 + e^{-x}} \quad (6.2)$$

The output of the neural network is four numbers, which are scaled by a sigmoid activation function to be values between 0 and 1. These numbers represent the probability of each target hit being correct. If the probability of a target hit is above a threshold then it is carried forward as a possible doublet. More than one target hit can be above threshold, meaning a seed hit can be a part of more than one doublet. This requires a process to select the best doublets in order to build tracks, described in Sec. 6.5.5.

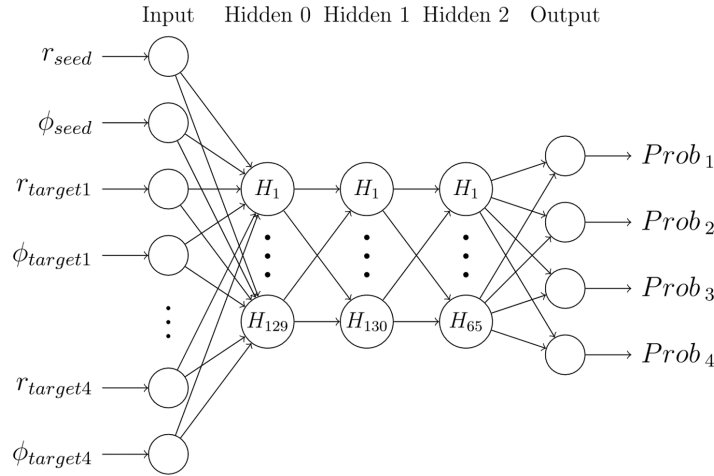


Figure 6.15: The neural network used to classify track doublets as true or false. The input contains a seed hit and four target hits on an adjacent module. There are three hidden layers. The four nodes in the output layer correspond to the probability of each target hit. Each output node has a sigmoid activation function to scale the output between 0 and 1. This allows the network to predict any number of connections between 0 and 4.

An important feature of any future tracking algorithm is the ability to be run in parallel. This can greatly reduce the run time when optimised for specialist hardware such as a GPU or FPGA. In this neural network, each seed hit is queried independently. This means the neural network can be run in parallel for each seed hit. This is in

contrast to conventional tracking algorithms that are serial processes.

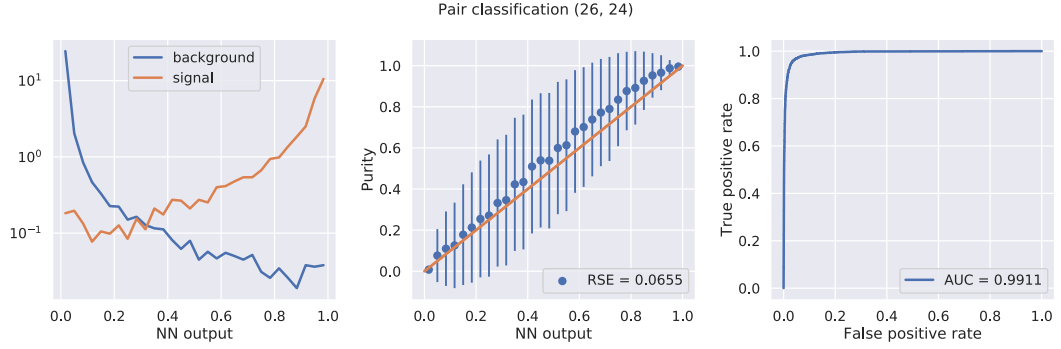


Figure 6.16: Pair selection performance for an outer pair of adjacent modules. a) Log plot showing the number of signal and background doublets produced by the neural network. A choice on the threshold of the neural network output is a compromise between signal efficiency and background rejection b). This plot shows how well the neural network output represents a probability. Points lying on the diagonal mean that hit purity scales linearly with the neural network output, this increases the readability of the neural network and the ease of error propagation. Points far from the diagonal could hint at training issues such as a biased training sample or over-training. The results for this neural network indicate a successful training process, with only a slight excess at high neural network output values. Error bars are binomial error. c) ROC curve of the neural network output.

An important parameter of the neural network is the threshold to accept doublets. It must be tuned for best performance, a low threshold will reject fewer true doublets, but let through more fake doublets and vice versa. Figure 6.16 shows how the background and signal (fake and true doublets) change with the threshold of the neural network output. However just measuring the doublet performance does not give the full picture. The performance of the full tracking algorithm must be investigated. I tuned the threshold of the neural network to give best efficiency, finding a value of 0.3 optimised efficiency. This value also led to a very low fake track rate.

#### 6.5.4 Data Sample and Network Training Details

The neural network was developed in PyTorch, which is a flexible and easy to use framework for creating neural networks in Python<sup>4</sup>. The neural network was trained with a dataset of 10 000 minimum bias events, simulated at upgrade conditions. Minimum bias events have almost no selections applied, so test algorithm performance on raw proton collisions. On average each event contains 134 tracks, with 14 tracks remaining after applying efficiency criteria described in Sec. 6.5.7. In order to simplify the tracking problem, only hits from one half of the VELO were used. Doing

<sup>4</sup><https://pytorch.org>

this removes the complication of tracks passing through the small overlap region between opposite sensors, but could reduce efficiency by missing tracks that transit both halves of the detector. However the geometry of the system means that almost all tracks remain within the same half. The popular *Adam* optimiser [115] was used for training the neural network, with a *Binary Cross Entropy with Logits* loss function. To avoid over-training, an early stopping procedure was applied to the training. If the loss function decreased by less than  $5 \times 10^{-3}$  in five consecutive epochs, then the training was stopped. A different set of 2000 simulated min-bias events were used for testing the reconstruction algorithm. It is important that these events are different to the training events.

### 6.5.5 Track Reconstruction

The neural network can predict more than one target hit for each seed hit. This results in tracks that branch. In the track reconstruction step, doublets are removed to eliminate branching so that they can be joined into tracks. Two doublets are needed to make a track with three hits, which is the minimum length for VELO tracks.

The first stage of the process is to apply a cut on  $r$  and  $\phi$  gradients to the doublets which removes spurious track doublets. This cut was set at  $\Delta\phi < 0.0003$  and  $\Delta r < 1$ .

The second step is to remove doublets that branch. Each hit should only be associated with one or two doublets, depending on whether it is at the end of a track, or in the middle of a track respectively. Therefore it is only necessary to check hits associated with more than two doublets. For hits in the middle of a track, there will be upstream *and* downstream doublets. The pair of up and downstream doublets with closest  $r$  gradient are kept for each hit. For hits at the end of a track, there will only be upstream *or* downstream doublets. The doublet with the minimum  $\phi$  gradient is kept. Because of the lack of a magnetic field in the VELO, tracks pointing to the origin have a constant  $\phi$  value. Doublets that share hits are joined into tracks.

### 6.5.6 Clone Reduction

As described above this hybrid tracking algorithm produced a high clone rate. Clone tracks are created when a true particle track is reconstructed as more than one track, for example being split into two sections. If a track has been split in multiple parts, one part is counted as a reconstructed track, the rest are counted as clone tracks. Figure 6.17 shows an image of clone tracks in an event. The clone rate is the number of clone tracks divided by the number of reconstructed tracks. For one track split in half, the clone rate would be 50%.

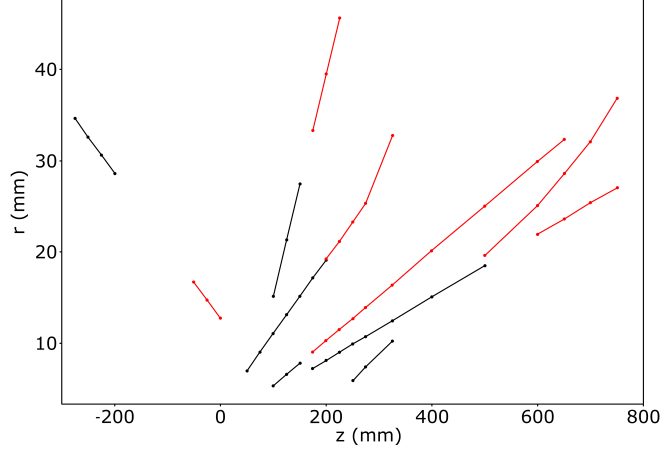


Figure 6.17: Example of clone tracks from a simulated event, before clone reduction. Clone tracks are tracks that are split into multiple segments, each track in this plot has two parts, black and red. Some have gaps where hits in some modules have been missed, others connect to incorrect hits on the same module. The clone reduction stage of the tracking algorithm is designed to join these parts together into single tracks.

To reduce the clone rate, clone tracks can be joined back together. If tracks share similar parameters, it may be a sign that they should have been reconstructed as a single track. The track parameters I chose were the  $r$  gradient,  $z$  intercept and mean  $\phi$  value. The euclidean distance between parameters of track pairs was calculated and if the distance was below a threshold, then the tracks would be joined together. This reduced clone rate from 5% to 1.1%. The pairwise comparisons of track parameters would not be fast if done by brute force, scaling with  $O(N^2)$ . Therefore a KDTree was used instead [123], this is a geometric data structure where k-nearest neighbour searches can be done scaling with  $O(N \log(N))$ .

### 6.5.7 Efficiency Criteria

Efficiency results are commonly quoted only for a selection of tracks to reflect physics performance and to allow comparison to other tracking methods. The requirements for VELO reconstruction are defined in the VELO upgrade TDR as [37]:

- The pseudorapidity must be within the LHCb acceptance of  $2 < \eta < 5$ .
- The track must have a momentum  $p > 5$  GeV
- The radius of the origin vertex must be less than 1 mm
- The  $z$ -position of the origin vertex must be within  $\pm 126$  mm of the interaction point.

### 6.5.8 Reconstruction Efficiency

In order to measure the performance of our algorithm I used the same definition of efficiency adopted in [37] for the conventional track reconstruction method. Simulated minimum bias events are used to determine the efficiency of the tracking algorithm. The number of correctly reconstructed tracks is compared to the number of possible reconstructible tracks, known from simulation truth information. The following definitions are used when talking about track reconstruction:

- A particle is reconstructible as a VELO track if there are clusters associated to it on three or more modules.
- A particle is considered reconstructed if at least 70% of the hits on a track are associated with that particle.
- A ghost track or fake track is a track which less than 50% of hits are associated with one simulated particle.
- If more than one reconstructed track is associated to a particle the extra tracks are counted as clone tracks.

The standard tracking metrics; efficiency, ghost rate and clone rate are calculated as follows:

$$\text{Efficiency} = \frac{\# \text{ Reconstructed}}{\# \text{ Reconstructable}} \quad (6.3)$$

$$\text{Ghost rate} = \frac{\# \text{ Ghost}}{\# \text{ All reconstructed}} \quad (6.4)$$

$$\text{Clone rate} = \frac{\# \text{ Clone}}{\# \text{ All reconstructed}} \quad (6.5)$$

Having found the best parameters for the neural network, I could compare the efficiency achieved by the hybrid algorithm to that achieved by the conventional approach. Results for the hybrid tracking algorithm are encouraging, demonstrating a performance similar to the requirements of the Run 3 LHCb tracking system, with a significant reduction in ghost rate. Figure 6.18 show the efficiency as a function of momentum and pseudorapidity. Table 6.2 presents results for efficiency, ghost rate and clone rate compared to the requirements set out in the VELO TDR [37].

$p > 5\text{GeV}$	Efficiency %	Ghost %	Clone %
<b>ML</b>	<b>97.3</b>	<b>0.11</b>	<b>1.04</b>
Conventional	98.9	2.5	1.0

Table 6.2: Table of results for 2000 minimum-bias events. Our method is compared to the conventional pattern recognition algorithm.

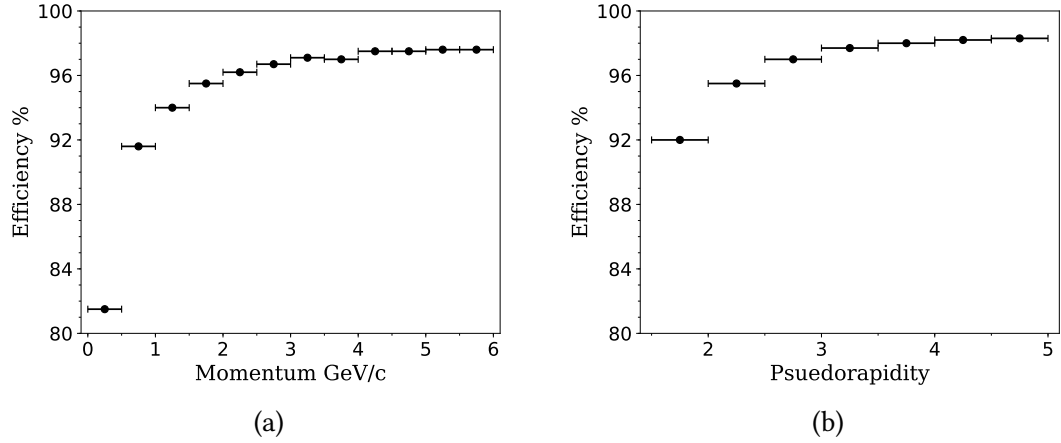


Figure 6.18: Tracking efficiency of the hybrid algorithm as a function of a) momentum and b) pseudorapidity for a minimum bias sample of 2000 events. High efficiency is generally maintained over each range, however it falls short of the standard shown in Figure 6.3 at low momentum especially.

### 6.5.9 Module Misalignment Study

The physical hardware of tracking detectors will never be installed perfectly accurately, and over time, the alignment of detector modules is likely to change by a small amount. Therefore, a tracking algorithm must be able to cope with these small misalignments. I measured the impact of small random misalignment of VELO modules on both the conventional tracking algorithm and my hybrid approach. The VELO upgrade TDR states that maximum module misalignments should be  $20\text{ }\mu\text{m}$  in  $x$ ,  $y$  and  $100\text{ }\mu\text{m}$  in the  $z$ -direction. In order to simulate this misalignment I displaced each modules  $x$  and  $y$  position coordinate by a value taken from a Gaussian distribution, centred on 0 with a standard deviation of 25, 50, 75 or  $100\text{ }\mu\text{m}$ . There was no displacement in  $z$  due to the hybrid model neural network not accepting  $z$  as an input. Five alignments were generated for each value of the standard deviation, and each tracking algorithm was run over 1000 events. I measured the efficiency, ghost rate and clone rate and calculated the error as the standard error on the mean of the five samples. I measured these metric without applying cuts on momentum or pseudorapidity. This is because it is important to have all tracks available for vertex finding



and detector alignment.

Figure 6.19 (a, b and c) show the results of this study presented as deltas, the difference in each metric from a perfectly aligned detector. My hybrid tracking method has lower deltas and therefore is more robust to small random misalignments than the conventional tracking algorithm, especially for misalignments of 50  $\mu\text{m}$  and above. This magnitude of misalignment is high considering the levels of alignment measured in Run 2, however an algorithm more robust to any misalignment would increase the speed at which the calibration procedure can occur. I believe this results also shows that the neural network generalised the tracking problem in a way that the conventional approach cannot.

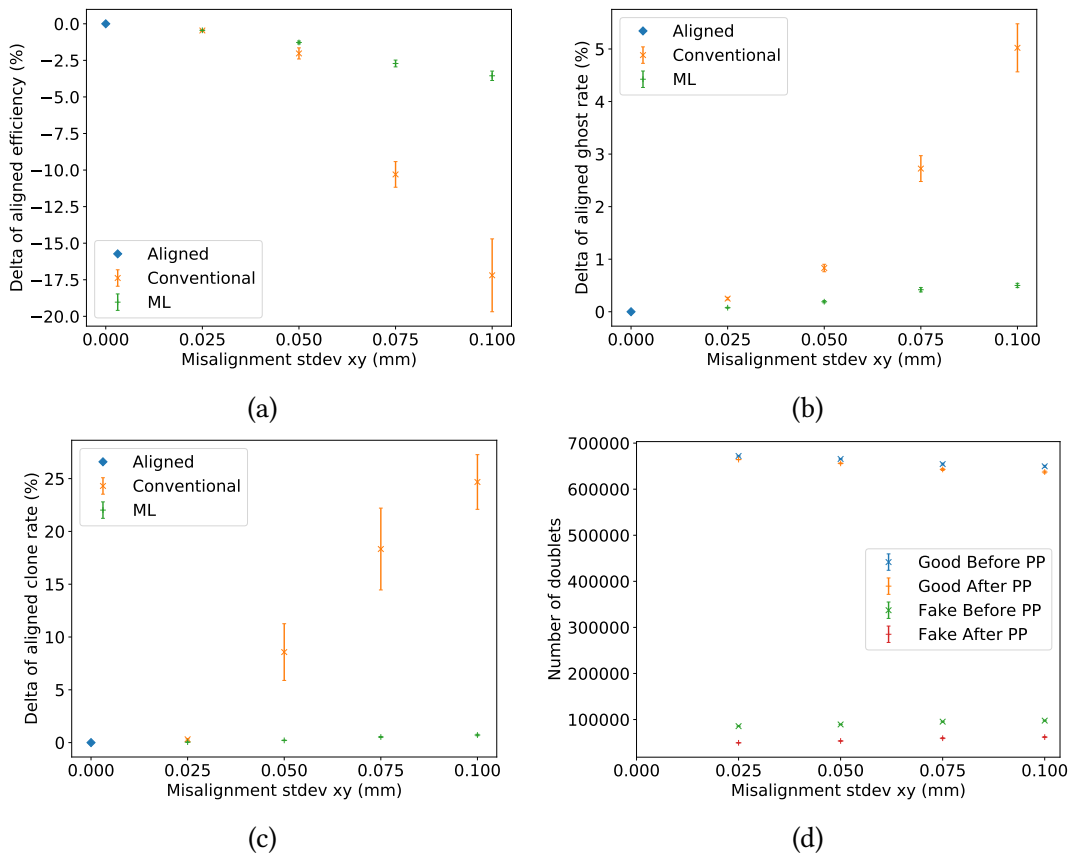


Figure 6.19: Results of alignment study. Deltas are the difference between each measure when misaligned and when perfectly aligned. a) Efficiency delta. b) Ghost rate delta. c) Clone rate delta. d) Number of doublets before and after post processing.

I also investigated the effect of these misalignments on the performance of the doublet reconstruction stage. Figure 6.19 (d) shows how the doublet processing step retains almost all good doublets, whilst rejecting a large proportion of fake doublets. Importantly, misalignment has a very small effect.

### 6.5.10 Closest target hit vs neural network

The most likely target hit is the closest in  $\phi$ , as seen in Fig. 6.13, and as seen in my earlier tracking investigations, neural networks can ‘cheat’ to reduce their loss function without taking the route intended by their programmer. I considered whether the neural network was more effective than simply choosing the closest target hit in  $\phi$ . Figure 6.20 shows the efficiency of the tracking algorithm, either using the neural network to select doublets, or simply choosing the closest target hit in  $\phi$  every time. Using the neural network outperforms choosing the closest target hit, especially for low momentum and low pseudorapidity tracks.

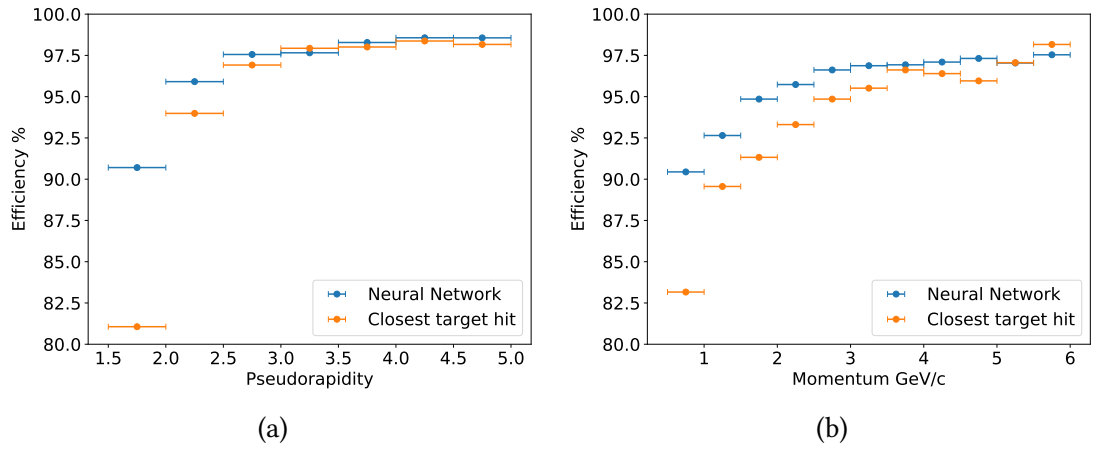


Figure 6.20: Plot showing comparison of using neural network, or only selecting the target hit closest in  $\phi$ . Efficiency is higher when the neural network is used to select doublets, compared to using the closest target hit in  $\phi$  every time. This is especially pronounced with low momentum and low pseudorapidity tracks, which are the most difficult to reconstruct.

### 6.5.11 Inference Time Investigation

Despite not developing a fully optimised tracking pipeline, I was interesting in investigating the power of hardware acceleration to increase performance. The neural network is the stage in the model that most benefits from hardware acceleration. Therefore, I tested how much faster neural network inference would be when run on a GPU. The University of Liverpool computing cluster is known as Barkla, and is ideal to run these tests as I could rely on consistent computing performance. I could use the cluster to run both on CPU and GPU.

I ran the neural network inference whilst changing two variables, the batch size of data entering the neural network and a PyTorch parameter called *num\_workers*. A neural network can work on an array of inputs, the size of this array is the batch size, a large batch size means the neural network can process a large set of inputs at once

at the cost of requiring more memory. The parameter *num\_workers* is the number of processes that generate batches in parallel, ensuring efficient use of computing resources. This only affects the data loading step.

The dataset was 100,000 neural network inputs, each the 5 pairs of hit coordinates. 100 inputs were used to 'warm-up' the neural network, this is required because the neural network takes a small amount of time to initiate, therefore using a warm-up batch I made sure not to include this initialisation time in the result. The sample of 100,000 inputs are split into batches given by the batch size, with the sum of the time taken to process all batches recorded. This was repeated 100 times for each value of the batch size and *num\_workers* and the final result was the minimum of these 100 times.

I used a batch size between 1 and 4000, which was the memory limit of the system. As well as measuring the time taken to run the neural network, I also measured the time to load data into neural network. The input data I used were random numbers, with a similar scale and in the correct shape to fit into the network. I used random numbers as this is quicker and more simple than importing simulation data and it does not affect the result of this study.

I found that running on GPU with a large batch size can increase the speed by approximately a factor of three compared to running on CPU. Increasing the batch size reduced the time taken to run on GPU, but has little effect when using CPU. This is because the CPU can only process one input at a time. I also found that the optimum value of *num\_workers* is 0, this is the automatic setting where PyTorch optimises the best value. GPU process has constraint of loading data onto GPU, however even with this time penalty, it is still faster than CPU. Figure 6.21 shows the results of this study, when running the neural network model only (a), and when including the data-loading step (b). The best times for 100,000 inferences can be seen in Tab. 6.3.

Method	Memory	CPU Cores	GPUs	Model	Data load	Combined
GPU	90 GB	6	1	7.63 ms	1.72 ms	9.35 ms
CPU	90 GB	6	0	29.6 ms	0 ms	29.6 ms

Table 6.3: Table of results inference study. Best times for 100,000 inferences.

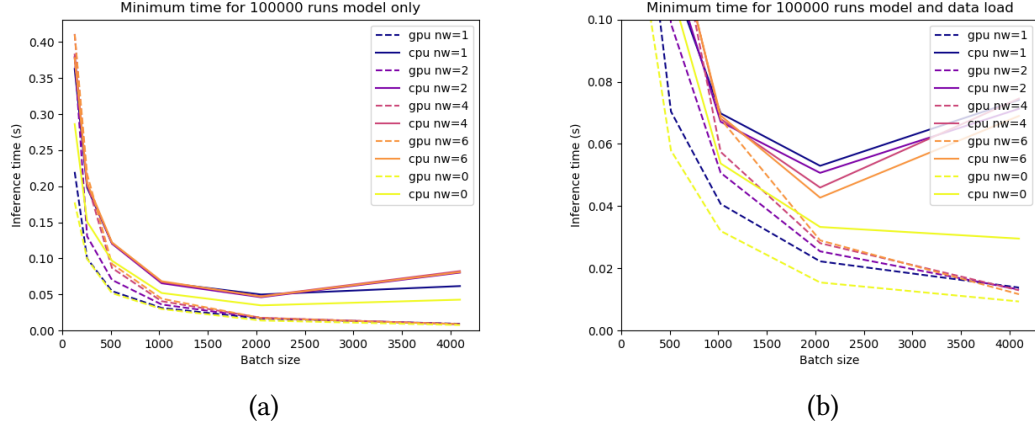


Figure 6.21: Plots showing the Neural network inference time with respect to the neural network batch size, comparing inference on CPU (solid lines) and GPU (dashed lines). Each line represents a different value for *num\_workers*. a) When running the NN model only, it is faster to run on GPU, especially at high batch sizes. b) Results of running the NN model and loading the data into the neural network. Loading the data is an overhead when running on GPU, meaning that larger batch sizes are needed for it to be faster to run on GPU.

## 6.6 Possible Improvements

### 6.6.1 Graph Neural Networks

A graph is a data structure of nodes connected by edges. The hits and tracks of the tracking problem can be seen as analogous to the nodes and edges of a graph. Some work was starting to use Graph Neural Network (GNN) in the context of particle physics tracking systems in 2018 [124]. The graph is initialised with a node representing each hit, connected to many adjacent nodes, analogous to many initial track seeds. Could the neural network learn to vary the strength of the edges, so that only the true connections (doublets) would remain?

At the time, work focused on toy models, I tried to apply a GNN to the VELO simulated dataset. I used the GNN as an addition to the fully connected network of the hybrid model, rather than a replacement. The loose doublet selection of the fully connected network was used as the initial selection of *edges* in the GNN, which would fine tune the selection. I found that I could increase efficiency and reduce the ghost rate by including the GNN. The doublet selection performance of the GNN is shown in Fig. 6.22.

However, this proved difficult at the time. GNNs were in their infancy, and packages for machine learning in PyTorch were not fully developed. I also found that the method of comparing gradients to remove branching doublets was just as effective at improving the doublet selection as the GNN. Due to these issues I decided to move

forward using only the fully connected neural network.

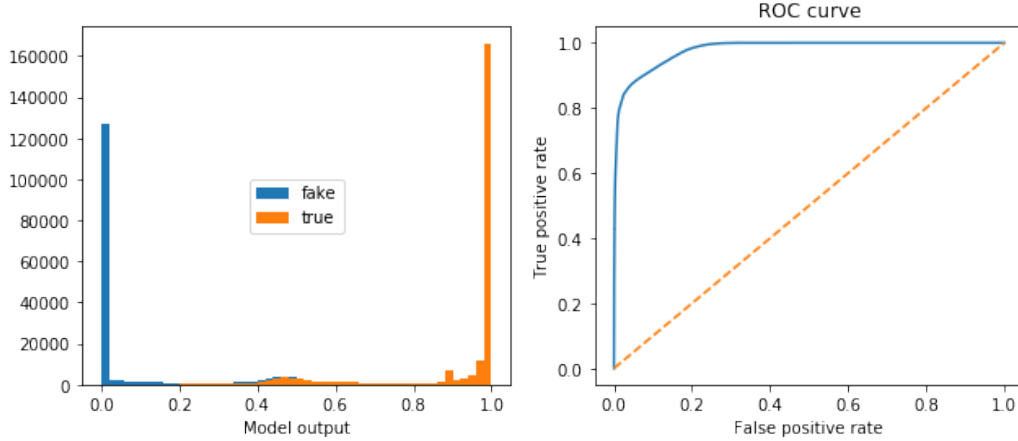


Figure 6.22: Pair selection performance of the Graph Neural Network. Most doublets are confidently classified as true or fake, however the bump in the histogram at 0.5 shows that the graph network is not sure about a number of doublets.

### 6.6.2 Connecting Doublets

I developed a custom method for discriminating and joining doublets into tracks, as described in Sec. 6.5.5. The ‘Graph walk’ method for connecting doublets might have worked better than my custom algorithm. It works by ‘walking’ along a track, selecting the longest chains of doublets to keep, instead of choosing at each hit depending on  $r$  and  $\phi$  gradients.

### 6.6.3 Retraining the Model

The hybrid tracking algorithm uses a separate neural network for each pair of VELO modules, each trained separately with only the  $x$  and  $y$  coordinate of each hit. But could a single neural network, trained on all modules work better? By including the  $z$ -coordinate of each hit in the training data, the neural network should learn at least as well because the data is equivalent.

After the spacial coordinates of each hit, there is one more piece of information that can be included in the neural network input. That is the number of pixels in each cluster. The pixels are so small that tracks passing at an angle to a module deposit energy in more than one pixel. In general, the more angled the track, the more pixels will be in a cluster. Therefore, this can give implicit information about the pseudorapidity of a track.

I used the opportunity of retraining to measure the neural network performance with respect to a number of different parameters. The number of target hits, size

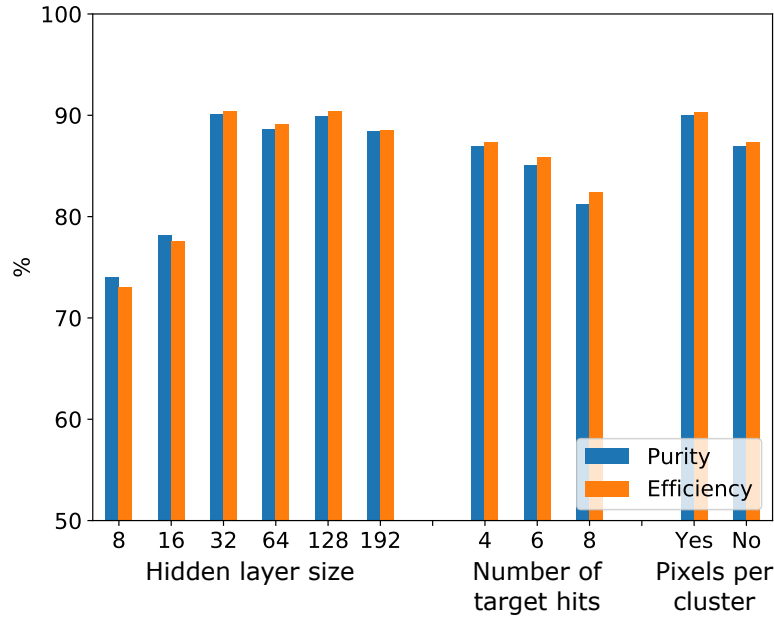


Figure 6.23: Neural network output performance after retraining. Showing average efficiency and purity of each target hit. Measured for different sizes of hidden layer, different numbers of target hits, and whether to include the number of pixels per cluster as an input feature. This figure shows that the best combination of variables are 32 nodes in each hidden layer, 4 closest target hits, and to include the number of pixels per cluster.

Method	Criteria	Efficiency %	Ghost %	Clone %
<b>Hybrid <math>r, \phi</math></b>	$p > 5 \text{ MeV } 2 < \eta < 5$	<b>97.3</b>	<b>0.11</b>	<b>1.04</b>
Hybrid $r, \phi$	all tracks	97.5	1.26	2.69
Retrain $r, \phi, z$	all tracks	97.8	0.38	3.28
Retrain $r, \phi, z, \# \text{ pixels/cluster}$	all tracks	97.8	0.23	2.86

Table 6.4: Table of results for 2000 minimum-bias events. Original hybrid method compared to the same method with a retrained neural network.

of neural network hidden layers and the inclusion of the number of pixels in each cluster. Figure 6.23 shows the results of optimising each of these variables. Above a hidden layer size of 32, the time taken to train and run the neural network increase, but with no increase in tracking performance. There is also a risk of over-training by using a size too large for the data.

I compared the retrained model to the original, the results are shown in Tab. 6.4. A lack of available data meant that I was unable to give results for only tracks meeting criteria of momentum and pseudorapidity. By only retraining the model, efficiency did increase and the ghost rate was reduced, however the clone rate did increase.

Perhaps more tuning of parameters would account for these differences. When I added the extra input variable, the number of pixels per cluster, the result was better, with only a small increase in clone rate over the original model. The main benefit of retraining this way - with only one neural network model - is that it reduced the complexity of the algorithm, and makes future retraining more simple.

## 6.7 Summary

In this chapter, I have described the development of a machine learning based particle tracking algorithm for the LHCb VELO detector. I started with basic toy models to learn and understand machine learning and the tracking problem. Firstly, training a neural network to find closest pairs of 2D points, then testing this algorithms on points arranged on a track-like shape; at each stage increasing complexity to better represent the real situation.

I then moved onto attempting to predict hits in adjacent layers of the VELO detector using simulation data. At this stage I investigated how hits should be represented, as pixels or coordinates. Using a Convolutional Neural Network (CNN) to process images of detector layers in intuitive as each layer is a pixel sensor. CNNs are widely used in high-energy physics experiments to identify tracks, such as those that use a Time Projection Chamber (TPC). However this was unsuccessful in this case, so I instead used a fully-connected neural network to process individual hit coordinates. Qualitative analysis gave me confidence that machine learning could work for the tracking problem.

Building on previous work to develop the hybrid tracking method, I used a neural network in the first stage of the tracking pipeline to select pairs of hits in adjacent detector layers (doublets) as seeds for tracks. I altered the method to increase the number of doublets that passed the neural network selection, then discriminated in a later step, rather than only choosing the neural networks *favourite* doublet.

I measured standard performance metrics of the hybrid tracking algorithm, finding high efficiency and low ghost and clone rates. I characterised performance with small detector misalignments, finding that the hybrid algorithm was more robust than the conventional tracking system to misalignment. I measured the time taken to run the neural network, comparing CPU to GPU. I found running on GPU to be 3 times faster.

I tried ways to improve the hybrid algorithm, such as retraining the model and adding an additional input feature, the number of pixels in a cluster. Adding this did improve the tracking performance, especially reducing the ghost rate. The original

neural network of the hybrid algorithm only used  $x$  and  $y$  as inputs, with a separate neural network trained for each pair of detector modules. When I retrained, I used a single neural network but adding  $z$  as an input instead. This greatly simplified the model and did not reduce tracking performance.

In Chap. 7 I will describe what more I would liked to have achieved if I had the time, such as reducing the number of post processing steps by developing an end-to-end machine learning solution. I would also have liked to better measure the inference time of the algorithm, and how this would compare when implemented on specialist hardware.





---

# Expanding the Model

---

THE hybrid tracking model described in Chapter 6 worked as a proof of concept that machine learning techniques could be applied to tracking algorithms. However this method does have limitations, outlined in Sec. 7.2. The doublet pruning and track joining processes takes time and could eliminate any savings made by using a neural network to generate track doublets, these additional steps also add human tuned parameters which could reduce tracking performance and it makes the algorithm more difficult to implement on accelerators described in this chapter. To fully utilise hardware acceleration it would be better for the full tracking process to be done in one stage on a neural network, this would vastly simplify the algorithm and give the very high throughput required for future tracking applications. In Sec. 7.2.1 I describe how an end-to-end machine learning tracking method could work, and the challenges I faced attempting to develop one. Then in Sec. 7.3 I explain the fundamentals of hardware acceleration and two levels of hardware that could be used, FPGAs and ASICs.

### 7.1 Shortcomings of the Hybrid Model

The hybrid model does have shortcomings, which could be addressed with further research and development. In the hybrid model, the neural network is only used for the initial step of the reconstruction sequence. This means that multiple post-processing steps are required to join doublets and create tracks, each of these steps

have parameters that need tuning and human decisions that might not be optimal. For example, the cut on doublet slopes after the neural network and the method used to join doublets at disputed hits. There is also an additional clone reduction step. One of the main benefits of a machine learning approach is the reduction in complexity and in human tuned parameters of the model; this is not entirely resolved by using a hybrid model with post-processing to produce tracks.

The fixed number of inputs to a neural network clashes with highly variable number of hits in each detector layer. I overcame this constraint in the hybrid model by only using the four target hits in the neural network. The need to calculate these closest hits takes time, and the number of target hits to use is another human tuned parameter in the model.

Even after tuning and post processing steps the tracking efficiency of the hybrid model is close but does not quite reach standard of the conventional tracking algorithm. The number of steps in the algorithm mean that it is not simple to tune the results, for example trading off efficiency for ghost rate and vice versa.

## 7.2 Possible Improvements

### 7.2.1 End to End Machine Learning

As opposed to a hybrid approach, an end-to-end machine learning method directly processes inputs into outputs with a single neural network, without other intermediate steps. In the context of tracking this means building a neural network that takes hits as inputs, and outputs tracks directly.

The advantages of end-to-end machine learning are that it cuts out time-consuming processes, which is especially important in a time critical environment such as the LHCb trigger. Another way it can be faster is by implementation on specialist hardware such as an FPGA or ASIC, this would be much more difficult to do with a hybrid tracking algorithm. It could also increase the performance of the tracking algorithm because the neural network can optimise the full process.

There can be disadvantages of end-to-end machine learning. Neural networks can be opaque in their reasoning, which could make it difficult to understand errors or failures. A hybrid approach allows for human readable parts of the algorithm whilst taking advantage of the neural network for the most difficult parts.

To achieve an end-to-end method for tracking, one would need to build a neural network that could take hits as inputs, and output tracks. The main challenge would be how to represent the inputs and outputs in a way the neural network can use to train, this is more difficult than it sounds at first. Figure 7.1 shows two possible ways

the output of the neural network could represent the grouping of hits into tracks. However, due to the changing number of hits and tracks in each event, the index and value of the output vector become meaningless between events. This makes it impossible for the neural network to learn the pattern. This is very different from an array of pixels in an image for example.

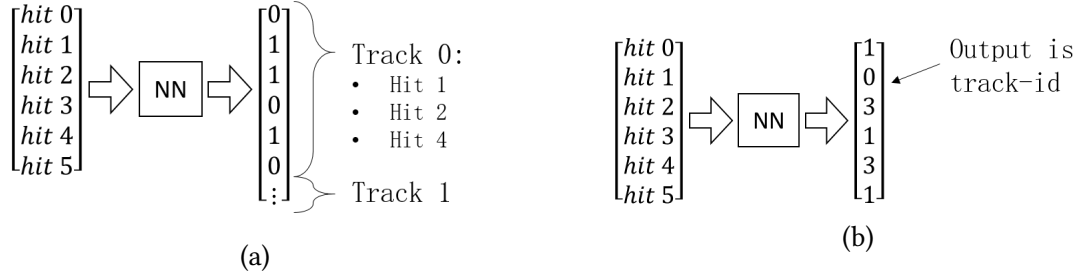


Figure 7.1: Two possible approaches for an End-to-end machine learning model. a) A binary output stating which hit belongs to each track. b) Each value of the output vector is a track id.

## 7.2.2 Multiple Neural Networks

The hybrid model used a neural network in the first stage of the tracking pipeline and then used conventional algorithms for post-processing to join doublets, build tracks and remove clones. There has been research to suggest that using multiple neural networks in a sequence could produce better performance than an end-to-end approach with a single large neural network [112]. It might be possible to replace the post-processing steps in the hybrid model with neural networks. This may overcome the issues I found when attempting to create an end-to-end model, namely how to represent hits and tracks and inputs and outputs of neural networks.

## 7.3 Hardware Acceleration

Hardware acceleration is the use of dedicated hardware to improve the performance of computation over what can be achieved with software on a CPU. In a time-sensitive environment such as the LHCb trigger, hardware acceleration can greatly reduce the run-time of algorithms such as VELO tracking. One way that hardware can accelerate processes is by parallelisation, for example being able to run the neural network over all hits simultaneously in the hybrid model, instead of the serial processing of a CPU. Other throughput improvements can be gained in the transfer of data. The structure of a neural network after training is a fixed arrangement of weights and biases which could be replicated directly on hardware. In the previous chapter I described how I

compared the execution time of a neural network when running on CPU and GPU, It would be interesting to test the speed of algorithm on more specialist hardware described below, however this was out of the scope of this work.

### FPGAs

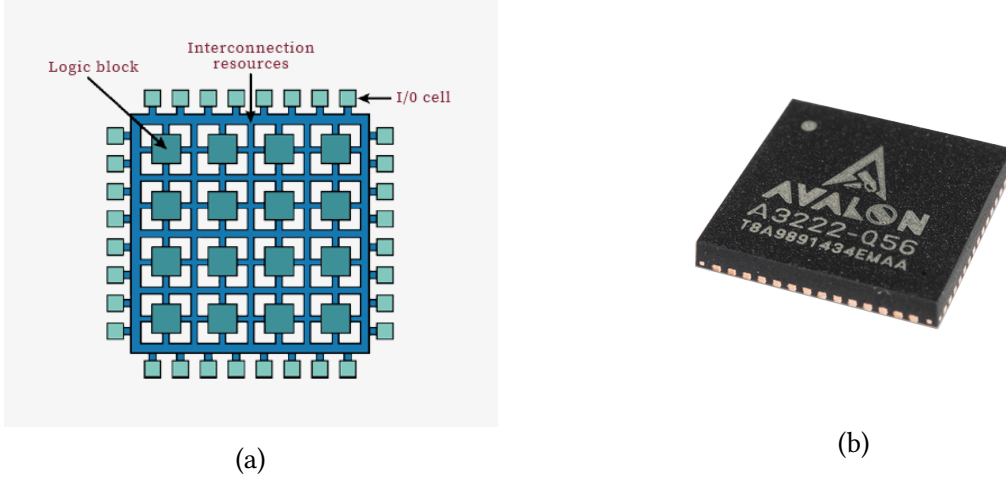


Figure 7.2: a) Diagram of FPGA functionality. b) Image of ASIC [125].

An FPGA is a computer chip containing a configurable array of digital circuits, known as configurable logic blocks (CLBs), connected together. They can be programmed by the user and offer high flexibility and performance. The user defines the circuit, as opposed to a CPU program where the user only defines the software.

Work has been done to implement neural networks on FPGAs in particle physics to increase the speed of data processing [126]. Much progress has been made to increase the speed of learning algorithms by utilising FPGAs [127]. Real-time clustering of pixels in the VELO will be done on FPGAs in Run 3 [84]. Considering that FPGAs are already used in LHCb data acquisition, it is realistic that tracking algorithms could also be moved to run this way.

### ASICs

An ASIC is not flexible like an FPGA, circuits are created in permanent hardware. This enables optimisation of execution time, without the compromises of an FPGA, such as larger power consumption. ASICs are used widely in high energy physics experiments. For example the VeloPix ASIC used to readout VELO pixel sensors. Google Deepmind reduced the power consumption of their AlphaGo AI by order of magnitude using ASICs compared to GPUs [128]. The downsides of such specialist hardware are longer, more complex and expensive development. However for simple machine learning models this complexity could be reduced.

---

## Conclusion

---

The next generation of high energy physics experiments will create data at a scale and rate that will stretch the capability of conventional event processing algorithms. The upgrade to LHCb for Run 3 of the LHC will enable full detector readout at the maximum bunch crossing rate of the LHC, alongside a five times increase in instantaneous luminosity, to greatly increase the reach of the future physics program. For the first time, tracks reconstructed in the VELO will be input into the first stage of a new software based trigger system. A significant increase in the speed of track reconstruction algorithms was required to meet the stringent throughput targets of the upgraded experiment.

In this thesis I have described my work to develop a machine learning based VELO track reconstruction algorithm. A machine learning approach was chosen because models, such as neural networks, have the potential for very fast inference times by exploiting parallel processing and hardware acceleration. I have also undertaken work at CERN to visually inspect front-end hybrids, a component of upgrade VELO modules. I found defects on wire bonding pads, caused by solder spattering, poorly soldered connection, due to hybrids not lying flat, and damage to electrical traces. As a result of this feedback, later batches of hybrids were of a high standard, ready for module construction. As a student of the LIV.DAT CDT, I spent six months on a data science focused industry placement. I worked with OnTrac Ltd based in Gateshead, UK, who are a software company that develop tools for planning safe engineering work on railway lines. I developed a dashboard to visualise data to clients that could help increase efficiency, and also safety for track workers.

The hybrid tracking algorithm incorporated a neural network in its first stage to

join pairs of hits into doublets, which is conventionally a process slowed down by combinatorics due to the high density of hits in the detector. Post processing steps filtered doublets and joined them into tracks, followed by a clone reduction procedure. The hybrid tracking algorithm showed promising performance, with over 97% efficiency and very low fake track rate, based on standard tracking criteria. This is close to the standard achieved by the conventional tracking algorithm, and was further improved by a retraining process to simplify the model and include extra hit information. I characterised the performance of the hybrid model when applied to simulation data with small random misalignments between detector layers, and compared this against the conventional algorithm. I measured standard tracking metrics and found that the hybrid model outperformed the conventional method, especially at larger misalignments. Showing that a machine learning tracking algorithm can be more robust than a conventional approach, demonstrating that the neural network learnt an understanding of the dataset.

Embracing the technology and methods of big data and machine learning will enable a bright and exciting future for high energy physics. The prize of overcoming the challenges of processing the vast output of upcoming experiments, such as the LHCb upgrade and HL-LHC, should be answers to the most elusive questions of the universe.

---

## Bibliography

---

- [1] Are Strandlie and Rudolf Frühwirth. “Track and vertex reconstruction: From classical to adaptive methods”. In: *Rev. Mod. Phys.* 82 (2 May 2010), pp. 1419–1458. DOI: 10.1103/RevModPhys.82.1419. URL: <https://link.aps.org/doi/10.1103/RevModPhys.82.1419>.
- [2] “Throughput and resource usage of the LHCb upgrade HLT”. In: (Apr. 2020). URL: <https://cds.cern.ch/record/2715210>.
- [3] R. Frühwirth. “Application of Kalman filtering to track and vertex fitting”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 262.2 (1987), pp. 444–450. ISSN: 0168-9002. DOI: [https://doi.org/10.1016/0168-9002\(87\)90887-4](https://doi.org/10.1016/0168-9002(87)90887-4). URL: <https://www.sciencedirect.com/science/article/pii/0168900287908874>.
- [4] 2m Bubble Chamber. “Bubble Chamber pictures for education”. In: (Aug. 1972). General Photo. URL: <https://cds.cern.ch/record/2307419>.
- [5] Alessandro Thea. *Introduction to trigger concepts*. Apr. 2019. URL: [https://indico.cern.ch/event/739424/contributions/3052225/attachments/1673751/2982840/ISOTDaq2019\\_TriggerIntro\\_thea.pdf](https://indico.cern.ch/event/739424/contributions/3052225/attachments/1673751/2982840/ISOTDaq2019_TriggerIntro_thea.pdf).
- [6] J. Albrecht et al. “The upgrade of the LHCb trigger system. The upgrade of the LHCb trigger system”. In: *JINST* 9 (Oct. 2014). Comments: Proceedings of the Workshop on Intelligent Trackers, 14-16 May 2014, University of Pennsylva-



- nia, C10026. 8 p. DOI: 10.1088/1748-0221/9/10/C10026. arXiv: 1410.5012. URL: <https://cds.cern.ch/record/1956613>.
- [7] Johannes Albrecht et al. “A Roadmap for HEP Software and Computing R&D for the 2020s”. In: *Computing and Software for Big Science* 3.1 (Mar. 2019). ISSN: 2510-2044. DOI: 10.1007/s41781-018-0018-8. URL: <http://dx.doi.org/10.1007/s41781-018-0018-8>.
  - [8] M Migliorini et al. “Machine Learning Pipelines with Modern Big Data Tools for High Energy Physics”. In: *Computing and Software for Big Science* 4 (June 2020). DOI: 10.1007/s41781-020-00040-0. URL: <https://doi.org/10.1007/s41781-020-00040-0>.
  - [9] Valentina Avati and et al. “Big Data Tools and Cloud Services for High Energy Physics Analysis in TOTEM Experiment”. In: *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*. 2018, pp. 5–6. DOI: 10.1109/UCC-Companion.2018.00018.
  - [10] E Sexton-Kennedy. “HEP Software Development in the Next Decade: the Views of the HSF Community”. In: *Journal of Physics: Conference Series* 1085 (Sept. 2018), p. 022006. DOI: 10.1088/1742-6596/1085/2/022006. URL: <https://doi.org/10.1088/1742-6596/1085/2/022006>.
  - [11] Alexander Radovic et al. “Machine learning at the energy and intensity frontiers of particle physics”. In: *Nature* 48 (2018). DOI: <https://doi.org/10.1038/s41586-018-0361-2>.
  - [12] P. et al Abratenko. “Convolutional neural network for multiple particle identification in the MicroBooNE liquid argon time projection chamber”. In: *Phys. Rev. D* 103 (9 May 2021), p. 092003. DOI: 10.1103/PhysRevD.103.092003. URL: <https://link.aps.org/doi/10.1103/PhysRevD.103.092003>.
  - [13] Kim Albertsson and et al. “Machine Learning in High Energy Physics Community White Paper”. In: (July 2018). DOI: arXiv:1807.02876. URL: <https://arxiv.org/abs/1807.02876>.
  - [14] Dan Guest, Kyle Cranmer, and Daniel Whiteson. “Deep Learning and Its Application to LHC Physics”. In: *Annual Review of Nuclear and Particle Science* 68.1 (Oct. 2018), pp. 161–181. ISSN: 1545-4134. DOI: 10.1146/annurev-nucl-101917-021019. URL: <http://dx.doi.org/10.1146/annurev-nucl-101917-021019>.

- 
- [15] Stephane Fartoukh et al. *LHC Configuration and Operational Scenario for Run 3*. Tech. rep. Geneva: CERN, Nov. 2021. URL: <https://cds.cern.ch/record/2790409>.
- [16] Lyndon Evans and Philip Bryant. “LHC Machine”. In: *Journal of Instrumentation* 3.08 (Aug. 2008), S08001–S08001. DOI: 10.1088/1748-0221/3/08/s08001.
- [17] The ATLAS Collaboration. “The ATLAS Experiment at the CERN Large Hadron Collider”. In: *Journal of Instrumentation* 3.08 (Aug. 2008), S08003–S08003. DOI: 10.1088/1748-0221/3/08/s08003. URL: <https://doi.org/10.1088/1748-0221/3/08/s08003>.
- [18] The CMS Collaboration. “The CMS experiment at the CERN LHC”. In: *Journal of Instrumentation* 3.08 (Aug. 2008), S08004–S08004. DOI: 10.1088/1748-0221/3/08/s08004. URL: <https://doi.org/10.1088/1748-0221/3/08/s08004>.
- [19] G. Aad et al. “Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC”. In: *Physics Letters B* 716.1 (2012), pp. 1–29. ISSN: 0370-2693. DOI: <https://doi.org/10.1016/j.physletb.2012.08.020>. URL: <https://www.sciencedirect.com/science/article/pii/S037026931200857X>.
- [20] S. Chatrchyan et al. “Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC”. In: *Physics Letters B* 716.1 (2012), pp. 30–61. ISSN: 0370-2693. DOI: <https://doi.org/10.1016/j.physletb.2012.08.021>. URL: <https://www.sciencedirect.com/science/article/pii/S0370269312008581>.
- [21] The ALICE Collaboration. “The ALICE experiment at the CERN LHC”. In: *Journal of Instrumentation* 3.08 (Aug. 2008), S08002–S08002. DOI: 10.1088/1748-0221/3/08/s08002. URL: <https://doi.org/10.1088/1748-0221/3/08/s08002>.
- [22] The LHCb Collaboration. “The LHCb Detector at the LHC”. In: *Journal of Instrumentation* 3.08 (Aug. 2008), S08005–S08005. DOI: 10.1088/1748-0221/3/08/s08005. URL: <https://doi.org/10.1088/1748-0221/3/08/s08005>.
- [23] Mike Lamont. “Status of the LHC”. In: *Journal of Physics: Conference Series* 455 (Aug. 2013), p. 012001. DOI: 10.1088/1742-6596/455/1/012001. URL: <https://doi.org/10.1088/1742-6596/455/1/012001>.

- [24] J.T. Boyd. *LHC Run-2 and Future Prospects*. Tech. rep. 9 pages, 6 figure, proceedings from lecture given at CERN-JINR ESHEP 2019 summer school in St Petersburg, Russia. Submitted for publication in a CERN Yellow Report. Jan. 2020. arXiv: 2001.04370. URL: <http://cds.cern.ch/record/2707815>.
- [25] Daniel Dominguez and Arzur Catel Torres. *CERN OVERVIEW animation*.
- [26] Julie Haffner. “The CERN accelerator complex. Complexe des accélérateurs du CERN”. In: (Oct. 2013). General Photo. URL: <http://cds.cern.ch/record/1621894>.
- [27] Oliver Sim Brüning et al. *LHC Design Report*. CERN Yellow Reports: Monographs. Geneva: CERN, 2004. DOI: 10.5170/CERN-2004-003-V-1. URL: <https://cds.cern.ch/record/782076>.
- [28] Klaus Hanke et al. “The LHC Injectors Upgrade (LIU) Project at CERN: Proton Injector Chain”. In: (2017), WEPVA036. 4 p. DOI: 10.18429/JACoW-IPAC2017-WEPVA036. URL: <http://cds.cern.ch/record/2289466>.
- [29] HiLumi HL-LHC prject. *Project Schedule*. URL: <https://project-hl-lhc-industry.web.cern.ch/content/project-schedule> (visited on 01/18/2022).
- [30] Apollinari G. et al. *High-Luminosity Large Hadron Collider (HL-LHC): Technical Design Report V. 0.1*. CERN Yellow Reports: Monographs. Geneva: CERN, 2017. DOI: 10.23731/CYRM-2017-004. URL: <https://cds.cern.ch/record/2284929>.
- [31] CERN. *The Worldwide LHC Computing Grid (WLCG)*. URL: <https://home.cern/science/computing/grid> (visited on 01/12/2022).
- [32] CERN. *Storage*. URL: <https://home.cern/science/computing/storage> (visited on 01/12/2022).
- [33] CERN. *The network challenge*. URL: <https://home.cern/science/computing/network> (visited on 01/12/2022).
- [34] Maria Arsuaga-Rios and et al. “LHC Data Storage: Preparing for the Challenges of Run-3”. In: *EPJ Web Conf.* 251 (2021), p. 02023. DOI: 10.1051/epjconf/202125102023. URL: <https://doi.org/10.1051/epjconf/202125102023>.
- [35] Esra Ozcesmeci. *LHC: pushing computing to the limits*. Mar. 2019. URL: <https://home.cern/news/news/computing/lhc-pushing-computing-limits> (visited on 01/20/2022).

- [36] LHCb Collaboration. *LHCb Tracker Upgrade Technical Design Report*. Tech. rep. CERN-LHCC-2014-001. LHCb-TDR-015. Feb. 2014. URL: <http://cds.cern.ch/record/1647400>.
- [37] LHCb Collaboration. *LHCb VELO Upgrade Technical Design Report*. Tech. rep. CERN-LHCC-2013-021. LHCb-TDR-013. Nov. 2013. URL: <https://cds.cern.ch/record/1624070>.
- [38] CERN (Meyrin) LHCb Collaboration. *Computing Model of the Upgrade LHCb experiment*. Tech. rep. CERN-LHCC-2018-014. LHCb-TDR-018. Geneva: CERN, May 2018. URL: <https://cds.cern.ch/record/2319756>.
- [39] V Parameswaran Nair. *Concepts in Particle Physics. A Concise Introduction to the Standard Model*. City College of the City University of New York, USA, 2018. DOI: 10.1142/10640.
- [40] Roel Aaij and et al. “Measurement of the  $W$  boson mass. Measurement of the  $W$  boson mass”. In: *JHEP* 2201 (Sept. 2021), 036. 38 p. DOI: 10.1007/JHEP01(2022)036. arXiv: 2109.01113. URL: <http://cds.cern.ch/record/2780004>.
- [41] John Ellis. “Physics at the LHC”. In: *The European Physical Journal C-Particles and Fields* 34.1 (2004), pp. 51–56.
- [42] N Tuning. *Lectures Notes on CP violation*. Feb. 2020. URL: <https://www.nikhef.nl/~h71/Lectures/2015/ppII-cpviolation-29012015.pdf>.
- [43] J. H. Christenson et al. “Evidence for the  $2\pi$  Decay of the  $K_2^0$  Meson”. In: *Phys. Rev. Lett.* 13 (4 July 1964), pp. 138–140. DOI: 10.1103/PhysRevLett.13.138. URL: <https://link.aps.org/doi/10.1103/PhysRevLett.13.138>.
- [44] CKMfitter Group (J. Charles et al.) In: *Eur. Phys. J.* C41 (2005). updated results and plots available at: <http://ckmfitter.in2p3.fr>, pp. 1–131. DOI: 10.1140/epjc/s2005-02169-1.
- [45] LHCb Collaboration. *Precise determination of the  $B_s^0-\overline{B}_s^0$  oscillation frequency. Precise determination of the  $B_0s-B_0sbar$  oscillation frequency*. Tech. rep. Geneva: CERN, Apr. 2021. arXiv: 2104.04421. URL: <http://cds.cern.ch/record/2764338>.
- [46] Karol Hennessy. “LHCb VELO upgrade”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 845 (2017). Proceedings of the Vienna Conference on Instrumentation 2016, pp. 97–100. ISSN: 0168-9002. DOI: <https://doi.org/>

- 10.1016/j.nima.2016.04.077. URL: <http://www.sciencedirect.com/science/article/pii/S016890021630290X>.
- [47] The LHCb Collaboration. “The LHCb Detector at the LHC”. In: *Journal of Instrumentation* 3.08 (Aug. 2008), S08005–S08005. DOI: 10.1088/1748-0221/3/08/s08005. URL: <https://doi.org/10.1088/1748-0221/3/08/s08005>.
  - [48] J. Andre et al. “Status of the LHCb dipole magnet”. In: *IEEE Transactions on Applied Superconductivity* 14.2 (2004), pp. 509–513. DOI: 10.1109/TASC.2004.829705.
  - [49] C. Joram, Guido Haefeli, and B. Leverington. “Scintillating Fibre Tracking at High Luminosity Colliders”. In: *Journal of Instrumentation* 10 (Aug. 2015), pp. C08005–C08005. DOI: 10.1088/1748-0221/10/08/C08005.
  - [50] Plamen Hopchev. *SciFi: A large Scintillating Fibre Tracker for LHCb*. 2017. eprint: 1710.08325.
  - [51] LHCb Collaboration. *LHCb PID Upgrade Technical Design Report*. Tech. rep. CERN-LHCC-2013-022. LHCb-TDR-014. Nov. 2013. URL: <https://cds.cern.ch/record/1624074>.
  - [52] *LHCb calorimeters: Technical Design Report*. Technical Design Report LHCb. Geneva: CERN, 2000. URL: <https://cds.cern.ch/record/494264>.
  - [53] *LHCb muon system: Technical Design Report*. Technical Design Report LHCb. Geneva: CERN, 2001. URL: <http://cds.cern.ch/record/504326>.
  - [54] *LHCb Trigger and Online Upgrade Technical Design Report*. Tech. rep. CERN-LHCC-2014-016. LHCb-TDR-016. May 2014. URL: <http://cds.cern.ch/record/1701361>.
  - [55] T Colombo et al. “The LHCb Online system in 2020: trigger-free read-out with (almost exclusively) off-the-shelf hardware”. In: *Journal of Physics: Conference Series* 1085 (Sept. 2018), p. 032041. DOI: 10.1088/1742-6596/1085/3/032041. URL: <https://doi.org/10.1088/1742-6596/1085/3/032041>.
  - [56] P Moreira et al. “The GBT Project”. In: (2009). DOI: 10.5170/CERN-2009-006.342. URL: <https://cds.cern.ch/record/1235836>.
  - [57] CERN (Meyrin) LHCb Collaboration. *LHCb Upgrade GPU High Level Trigger Technical Design Report*. Tech. rep. Geneva: CERN, May 2020. URL: <https://cds.cern.ch/record/2717938>.

- [58] Maximilien Brice and Julien Ordan. “LHCb new datacenters CERN-PHOTO-201905-123-5”. In: (May 2019). General Photo. URL: <https://cds.cern.ch/record/2674755>.
- [59] “LHCb - Large Hadron Collider beauty experiment”. URL: <http://lhcb-public.web.cern.ch/lhcb-public/>.
- [60] “LHCb detector performance”. In: *International Journal of Modern Physics A* 30.07 (Mar. 2015), p. 1530022. ISSN: 1793-656X. DOI: 10.1142/s0217751x15300227. URL: <http://dx.doi.org/10.1142/S0217751X15300227>.
- [61] LHCb Collaboration. “Observation of CP Violation in Charm Decays”. In: *Physical Review Letters* 122.21 (May 2019). ISSN: 1079-7114. DOI: 10.1103/physrevlett.122.211803. URL: <http://dx.doi.org/10.1103/PhysRevLett.122.211803>.
- [62] R. Aaij et al. “Search for Lepton-Universality Violation in  $B^+ \rightarrow K^+ \ell^+ \ell^-$  Decays”. In: *Physical Review Letters* 122.19 (May 2019). ISSN: 1079-7114. DOI: 10.1103/physrevlett.122.191801. URL: <http://dx.doi.org/10.1103/PhysRevLett.122.191801>.
- [63] R. Aaij et al. “Measurement of the  $B_s^0 \rightarrow \mu^- \mu^+$  Branching Fraction and Effective Lifetime and Search for  $B^0 \rightarrow \mu^- \mu^+$  Decays”. In: *Physical Review Letters* 118.19 (May 2017). ISSN: 1079-7114. DOI: 10.1103/physrevlett.118.191801. URL: <http://dx.doi.org/10.1103/PhysRevLett.118.191801>.
- [64] R. Aaij et al. “Updated measurement of time-dependent CP-violating observables in  $B_s^0 \rightarrow J/\psi K^+ K^-$  decays”. In: *The European Physical Journal C* 79.8 (Aug. 2019). ISSN: 1434-6052. DOI: 10.1140/epjc/s10052-019-7159-8. URL: <http://dx.doi.org/10.1140/epjc/s10052-019-7159-8>.
- [65] I. Belyaev et al. “The history of LHCb”. In: *The European Physical Journal H* 46.1 (Mar. 2021). DOI: 10.1140/epjh/s13129-021-00002-z.
- [66] R. Aaij et al. “Observation of  $J/\psi p$  Resonances Consistent with Pentaquark States in  $\Lambda_b^0 \rightarrow J/\psi K^- p$  Decays”. In: *Physical Review Letters* 115.7 (Aug. 2015). ISSN: 1079-7114. DOI: 10.1103/physrevlett.115.072001. URL: <http://dx.doi.org/10.1103/PhysRevLett.115.072001>.
- [67] R. Aaij et al. “Near-threshold  $D\bar{D}$  spectroscopy and observation of a new charmonium state”. In: *Journal of High Energy Physics* 2019.7 (July 2019). ISSN: 1029-8479. DOI: 10.1007/jhep07(2019)035. URL: [http://dx.doi.org/10.1007/JHEP07\(2019\)035](http://dx.doi.org/10.1007/JHEP07(2019)035).

- [68] LHCb Collaboration. “Branching Fraction Measurements of the Rare  $B_s^0 \rightarrow \phi \mu^+ \mu^-$  and  $B_s^0 \rightarrow f_2'(1525) \mu^+ \mu^-$  - Decays”. In: *Phys. Rev. Lett.* 127 (May 2021). All figures and tables, along with any supplementary material and additional information, are available at <https://cern.ch/lhcbproject/Publications/p/LHCb-PAPER-2021-014.html>(LHCbpublic pages), 151801. 11 p. DOI: 10.1103/PhysRevLett.127.151801. arXiv: 2105.14007. URL: <https://cds.cern.ch/record/2770827>.
- [69] et al Barbosa-Marinho. *LHCb VELO (VERtex LOCator): Technical Design Report*. Technical design report. LHCb. Geneva: CERN, 2001. URL: <https://cds.cern.ch/record/504321>.
- [70] R. Aaij and et al. “Performance of the LHCb Vertex Locator”. In: *JINST* 9 (May 2014). Comments: 61 pages, 33 figures, P09007. 61 p. DOI: 10.1088/1748-0221/9/09/P09007. arXiv: 1405.7808. URL: <https://cds.cern.ch/record/1707015>.
- [71] Oxford University. “Vertex Locator”. URL: <https://www2.physics.ox.ac.uk/research/lhcb/detector-activities/vertex-locator>.
- [72] Paula Collins and et al. *Microchannel Cooling for the LHCb VELO Upgrade I. Microchannel Cooling for the LHCb VELO Upgrade*. Tech. rep. 31 pages, 27 figures. Geneva: CERN, Dec. 2021. arXiv: 2112.12763. URL: <https://cds.cern.ch/record/2792295>.
- [73] Peter Svihira. “The Design and Construction of LHCb VELO Upgrades Modules”. In: (Aug. 2020). Poster-2020-1041. URL: <http://cds.cern.ch/record/2727215>.
- [74] Shantam Taneja and John Cobbledick. *4D Pattern Recognition in VELO Upgrade-II*. RTA Workshop 2019. July 2019. URL: <https://indico.cern.ch/event/793125/contributions/3502212/>.
- [75] *LHCb trigger system: Technical Design Report*. Technical Design Report LHCb. revised version number 1 submitted on 2003-09-24 12:12:22. Geneva: CERN, 2003. URL: <https://cds.cern.ch/record/630828>.
- [76] “RTA and DPA dataflow diagrams for Run 1, Run 2, and the upgraded LHCb detector”. In: (Sept. 2020). URL: <https://cds.cern.ch/record/2730181>.
- [77] T Head. “The LHCb trigger system”. In: *Journal of Instrumentation* 9.09 (Sept. 2014), pp. C09015–C09015. DOI: 10.1088/1748-0221/9/09/c09015. URL: <https://doi.org/10.1088%2F1748-0221%2F9%2F09%2Fc09015>.

- [78] R. Aaij et al. “A comprehensive real-time analysis model at the LHCb experiment”. In: *JINST* 14.arXiv:1903.01360. 04 (Mar. 2019). 15 pages, 4 figures, 1 table. All figures and tables are available at <https://cern.ch/lhcbproject/Publications/LHCbProjectPublic/LHCb-DP-2019-002.html>, P04006. 15 p. DOI: 10.1088/1748-0221/14/04/P04006. URL: <https://cds.cern.ch/record/2665946>.
- [79] Dorothea Vom Bruch. *Physics performance and event throughput of the GPU High Level Trigger 1 of LHCb*. CHEP 2019. Nov. 2019. URL: [https://indico.cern.ch/event/773049/contributions/3474298/attachments/1938619/3213523/vom\\_Bruch\\_Allen\\_chep2019.pdf](https://indico.cern.ch/event/773049/contributions/3474298/attachments/1938619/3213523/vom_Bruch_Allen_chep2019.pdf).
- [80] Karl Rupp. “42 Years of Microprocessor Trend Data”. Feb. 2018. URL: <https://www.karlrupp.net/2018/02/42-years-of-microprocessor-trend-data/>.
- [81] R Aaij et al. “A Comparison of CPU and GPU Implementations for the LHCb Experiment Run 3 Trigger”. In: *Computing and Software for Big Science* 6 (Dec. 2021). DOI: 10.1007/s41781-021-00070-2. URL: <https://doi.org/10.1007/s41781-021-00070-2>.
- [82] Renata Kopečna. *Tracking and vertexing in LHCb*.
- [83] Jianchun Wang. “The Upstream Tracker for the LHCb upgrade”. In: 2015.
- [84] Federico Lazzari et al. “Real-time cluster finding for LHCb silicon pixel VELO detector using FPGA”. In: 1525 (Apr. 2020), p. 012044. DOI: 10.1088/1742-6596/1525/1/012044. URL: <https://doi.org/10.1088/1742-6596/1525/1/012044>.
- [85] T Bird et al. *VP Simulation and Track Reconstruction*. Tech. rep. LHCb-PUB-2013-018. CERN-LHCb-PUB-2013-018. Geneva: CERN, Oct. 2013. URL: <https://cds.cern.ch/record/1620453>.
- [86] R. Fruehwirth. “Application of Kalman Filtering to Track Fitting in the DELPHI Detector”. In: (1987).
- [87] J. M. Amoraal. “Alignment of the LHCb detector with Kalman filter fitted tracks”. In: *J. Phys. Conf. Ser.* 219 (2010), p. 032028. DOI: 10.1088/1742-6596/219/3/032028.
- [88] Giuseppe Cerati et al. “Parallelized and Vectorized Tracking Using Kalman Filters with CMS Detector Geometry and Events”. In: (2018). arXiv: 1811.04141 [physics.comp-ph].



- 
- [89] Florian Reiss. “Fast parallel Primary Vertex reconstruction for the LHCb Upgrade”. In: (Apr. 2020). URL: <https://cds.cern.ch/record/2717609>.
- [90] R Frühwirth and W Waltenberger. *Adaptive Multi-vertex fitting*. Tech. rep. Geneva: CERN, Sept. 2004. DOI: 10.5170/CERN-2005-002.280. URL: <https://cds.cern.ch/record/803519>.
- [91] E Bowen and B Storaci. *VeloUT tracking for the LHCb Upgrade*. Tech. rep. LHCb-PUB-2013-023. CERN-LHCb-PUB-2013-023. LHCb-INT-2013-056. Geneva: CERN, Apr. 2014. URL: <http://cds.cern.ch/record/1635665>.
- [92] Luis Miguel Garcia et al. “Tracking performance for long-lived particles at LHCb”. In: *J. Phys.: Conf. Ser.* 1525 (Oct. 2019). Proceeding for Connecting the Dots and Workshop on Intelligent Trackers (CTD/WIT 2019), 012095. 5 p. DOI: 10.1088/1742-6596/1525/1/012095. arXiv: 1910.06171. URL: <https://cds.cern.ch/record/2697348>.
- [93] Agnieszka Dziurda et al. “Real-time alignment and reconstruction: performance and recent developments at the LHCb experiment”. In: *Journal of Physics: Conference Series* 1085 (Sept. 2018), p. 042003. DOI: 10.1088/1742-6596/1085/4/042003. URL: <https://doi.org/10.1088/1742-6596/1085/4/042003>.
- [94] Chris Burr. *LHCb full-detector real-time alignment and calibration - Latest developments and perspectives*. CHEP 2018. July 2018. URL: [https://cds.cern.ch/record/2633213/files/2018\\_CHEP\\_LHCb-realtime-alignment-and-calibration%5C%20Burr%5C%2010.07.pdf](https://cds.cern.ch/record/2633213/files/2018_CHEP_LHCb-realtime-alignment-and-calibration%5C%20Burr%5C%2010.07.pdf).
- [95] Charles Duhigg. “How Companies Learn Your Secrets”. In: *New York Times Magazine* (Feb. 2012). URL: <https://www.nytimes.com/2012/02/19/magazine/shopping-habits.html?pagewanted=1&r=1&hp>.
- [96] Scott Mayer McKinney, Marcin Sieniek, and Shravya Shetty. “International evaluation of an AI system for breast cancer screening”. In: *Nature* 577 (Jan. 2020), pp. 89–94. DOI: 10.1038/s41586-019-1799-6. URL: <https://www.nature.com/articles/s41586-019-1799-6>.
- [97] A. M. TURING. “I.—COMPUTING MACHINERY AND INTELLIGENCE”. In: *Mind* LIX.236 (Oct. 1950), pp. 433–460. ISSN: 0026-4423. DOI: 10.1093/mind/LIX.236.433. eprint: <http://oup.prod.sis.lan/mind/article-pdf/LIX/236/433/30123314/lix-236-433.pdf>. URL: <https://doi.org/10.1093/mind/LIX.236.433>.

- [98] W.S. McCulloch and W. A Pitts. “A logical calculus of the ideas immanent in nervous activity.” In: *Bulletin of Mathematical Biophysics* (). DOI: 10 . 1007 / BF02478259. URL: <https://doi.org/10.1007/BF02478259>.
- [99] F. Rosenblatt. “The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain”. In: *Psychological Review* (1958), pp. 65–386.
- [100] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. “Learning Internal Representations by Error Propagation”. In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*. Cambridge, MA, USA: MIT Press, 1986, pp. 318–362. ISBN: 026268053X.
- [101] Donald Michie. “Experiments on the Mechanization of Game-Learning Part I. Characterization of the Model and its parameters”. In: *The Computer Journal* 6.3 (Nov. 1963), pp. 232–236. ISSN: 0010-4620. DOI: 10 . 1093 / comjnl / 6 . 3 . 232. eprint: <http://oup.prod.sis.lan/comjnl/article-pdf/6/3/232/1030939/6-3-232.pdf>. URL: <https://doi.org/10.1093/comjnl/6.3.232>.
- [102] Dan Guest, Kyle Cranmer, and Daniel Whiteson. “Deep Learning and Its Application to LHC Physics”. In: *Annual Review of Nuclear and Particle Science* 68.1 (2018), pp. 161–181. DOI: 10 . 1146 / annurev - nucl - 101917 - 021019. URL: <https://doi.org/10.1146/annurev-nucl-101917-021019>.
- [103] Phạm Hồ Thanh. *Learn bicycling*. Attribution-NonCommercial-ShareAlike 2.0 Generic (CC BY-NC-SA 2.0). Aug. 2018. URL: <https://www.flickr.com/photos/phamhothanh/44283825802>.
- [104] Martin Ester et al. “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise”. In: *KDD*. 1996, pp. 226–231. URL: <http://www.aaai.org/Library/KDD/1996/kdd96-037.php>.
- [105] Tinghui Zhou et al. “Unsupervised Learning of Depth and Ego-Motion from Video”. In: *CVPR*. 2017. DOI: 10 . 48550 / 1704 . 07813. URL: <https://arxiv.org/abs/1704.07813>.
- [106] R. et al Aaij. “Measurement of the  $B_s^0 \rightarrow \mu^+ \mu^-$  Branching Fraction and Search for  $B_s^0 \rightarrow \mu^+ \mu^-$  Decays at the LHCb Experiment”. In: *Phys. Rev. Lett.* 111 (10 Sept. 2013), p. 101805. DOI: 10 . 1103 / PhysRevLett . 111 . 101805. URL: <https://link.aps.org/doi/10.1103/PhysRevLett.111.101805>.
- [107] Vladimir Vava Gligorov and Mike Williams. *Efficient, reliable and fast high-level triggering using a bonsai boosted decision tree*. 2012. arXiv: 1210 . 6861 [physics.ins-det].

- [108] Whiteson D Baldi P Sadowski P. “Searching for exotic particles in high-energy physics with deep learning”. In: *Nature Communications* 5 (1 2014). DOI: 10 . 1038/ncomms5308. URL: <https://doi.org/10.1038/ncomms5308>.
- [109] DUNE Collaboration. *Neutrino interaction classification with a convolutional neural network in the DUNE far detector*. 2020. arXiv: 2006 . 15052 [physics.ins-det].
- [110] Rua Herrera, Alex, Calvo Gómez, Míriam, and Vilasís Cardona, Xavier. “Particle identification with an electromagnetic calorimeter using a Convolutional Neural Network”. In: *EPJ Web Conf.* 251 (2021), p. 04032. DOI: 10 . 1051 / epjconf / 202125104032. URL: <https://doi.org/10.1051/epjconf/202125104032>.
- [111] Catherine Biscarat et al. “Towards a realistic track reconstruction algorithm based on graph neural networks for the HL-LHC”. In: *EPJ Web of Conferences* 251 (2021). Ed. by C. Biscarat et al., p. 03047. ISSN: 2100-014X. DOI: 10 . 1051 / epjconf / 202125103047. URL: <http://dx.doi.org/10.1051/epjconf/202125103047>.
- [112] Saito, Masahiko et al. “Event Classification with Multi-step Machine Learning”. In: *EPJ Web Conf.* 251 (2021), p. 03036. DOI: 10 . 1051 / epjconf / 202125103036. URL: <https://doi.org/10.1051/epjconf/202125103036>.
- [113] Hariri, Ali, Dyachkova, Darya, and Gleyzer, Sergei. “Graph Variational Autoencoder for Detector Reconstruction and Fast Simulation in High-Energy Physics”. In: *EPJ Web Conf.* 251 (2021), p. 03051. DOI: 10 . 1051 / epjconf / 202125103051. URL: <https://doi.org/10.1051/epjconf/202125103051>.
- [114] Britton, Thomas, Lawrence, David, and Rajput, Kishansingh. “AI Enabled Data Quality Monitoring with Hydra”. In: *EPJ Web Conf.* 251 (2021), p. 04010. DOI: 10 . 1051 / epjconf / 202125104010. URL: <https://doi.org/10.1051/epjconf/202125104010>.
- [115] D Kingma and J B. “Adam: A Method for Stochastic Optimization”. In: (Dec. 2014). DOI: 10 . 48550. arXiv: 1412 . 6980. URL: <https://doi.org/10.48550/arXiv.1412.6980>.
- [116] *OnTrac website*. URL: <https://on-trac.co.uk/>.
- [117] RAIB. *Track workers struck by a train at Margam, Neath Port Talbot, 3 July 2019*. 2020. URL: <https://www.gov.uk/government/news/report-112020-track-workers-struck-by-a-train-at-margam>.

- [118] Paul Harrop. *Track workers, Hull*. Attribution-ShareAlike 2.0 Generic (CC BY-SA 2.0). Mar. 2014. URL: <https://www.geograph.org.uk/photo/3872591>.
- [119] Ltd Opsview. *Screen shot of Opsview Monitor 6.0 (IT monitoring system) custom dashboard*. Dec. 2018. URL: [https://commons.wikimedia.org/wiki/File:Opsview\\_Monitor\\_6.0\\_Dashboard.jpg](https://commons.wikimedia.org/wiki/File:Opsview_Monitor_6.0_Dashboard.jpg). License: Creative Commons Attribution-ShareAlike 3.0 Unported (CC BY-SA 3.0).
- [120] Moritz Kiehn et al. “The TrackML high-energy physics tracking challenge on Kaggle”. In: *EPJ Web Conf.* 214 (2019). Ed. by A. Forti et al., p. 06037. DOI: 10.1051/epjconf/201921406037.
- [121] Martin Ester et al. “A density-based algorithm for discovering clusters in large spatial databases with noise”. In: AAAI Press, 1996, pp. 226–231.
- [122] Kurt Rinnert and Marco Cristoforetti. “Deep Learning Approach to Track Reconstruction in the upgraded VELO”. In: *EPJ Web of Conferences* 214 (Jan. 2019), p. 06038. DOI: 10.1051/epjconf/201921406038.
- [123] Jon Louis Bentley. “Multidimensional Binary Search Trees Used for Associative Searching”. In: *Commun. ACM* 18.9 (Sept. 1975), pp. 509–517. ISSN: 0001-0782. DOI: 10.1145/361002.361007. URL: <https://doi.org/10.1145/361002.361007>.
- [124] Steven Farrell et al. *Novel deep learning methods for track reconstruction*. 2018. arXiv: 1810.06111 [hep-ex].
- [125] ngzhang. *Avalon bitcoin mining ASIC, A3222*. Jan. 2015. URL: [https://commons.wikimedia.org/wiki/File:Avalon\\_ASIC\\_A3222.png](https://commons.wikimedia.org/wiki/File:Avalon_ASIC_A3222.png). License: Attribution-ShareAlike 3.0 Unported (CC BY-SA 3.0).
- [126] J. Duarte et al. “Fast inference of deep neural networks in FPGAs for particle physics”. In: *Journal of Instrumentation* 13.07 (July 2018), P07027–P07027. DOI: 10.1088/1748-0221/13/07/p07027. URL: <https://doi.org/10.1088/1748-0221/13/07/p07027>.
- [127] Sparsh Mittal. “A survey of FPGA-based accelerators for convolutional neural networks”. In: *Neural Computing and Applications* 32 (4 Oct. 2018). DOI: 10.1007/s00521-018-3761-1. URL: <https://doi.org/10.1007/s00521-018-3761-1>.
- [128] David Silver and Demis Hassabis. *AlphaGo Zero: Starting from scratch*. Oct. 2017. URL: <https://www.deepmind.com/blog/alphago-zero-starting-from-scratch>.



---

## List of Figures

---

1.1	A film photo of tracks in a bubble chamber from the CERN Proton Synchrotron [4]. . . . .	2
1.2	Yield of LHCb Run 2 trigger at upgrade luminosity's saturates [6]. . . . .	4
1.3	2017 estimate of the huge increase in CPU and disk space requirements of CMS for the HL-LHC era [10] . . . . .	4
1.4	An example of tracks seen by the MicroBooNE neutrino detector [12]. . . .	5
2.1	Diagram of Large Hadron Collider (LHC) facility[26]. . . . .	9
2.2	Timeline of the LHC from Long Shutdown 2 to the HL-LHC [29]. . . . .	10
2.3	Diagram of the upgraded LHCb detector [36]. . . . .	12
2.4	Components of the standard model of particle physics. . . . .	13
2.5	CKM Triangles from 2009 and 2019 [44]. . . . .	14
2.6	(a) Illustration of one half of the VELO detector. (b) A view of the VELO in the $xz$ plane [37]. . . . .	16
2.7	Diagram of Upstream Tracker (UT) [36]. . . . .	16
2.8	Diagram of LHCb magnet with a human for scale [47]. . . . .	17
2.9	Diagram of Scintillating Fibre tracker (SciFi) [49]. . . . .	18
2.10	Diagram of RICH detector [51]. . . . .	19
2.11	Layout of sensors in ECAL, one quarter of face shown [52]. . . . .	20
2.12	Layout of sensors in HCAL, one quarter of face shown [52]. . . . .	21
2.13	Diagram of the face of one station of the muon detector. Details of each sector are shown [53]. . . . .	22

2.14	(a) Upgraded LHCb readout system [54]. (b) Timing and Fast Control system [55] . . . . .	22
2.15	The new LHCb containerised data centre under construction at point 8. [58]	24
2.16	Integrated luminosity recorded each year of Runs 1 and 2 [59]. . . . .	24
3.1	View in the $xz$ plane showing VELO module positions and pseudorapidity acceptance. . . . .	28
3.2	. . . . .	29
3.3	Data rate per ASIC in $\text{Gbs}^{-1}$ for most active VELO module [37]. . . . .	31
3.4	Schematic of VELO electronics [37]. . . . .	31
3.5	VELO simulation - a) IP resolution and b) efficiency with respect to $\phi$ [37].	33
3.6	Radiation dose as a function of radius from beam for two VELO sensors [37].	33
3.7	Images of front-end hybrid visual inspection service task. . . . .	36
4.1	LHCb upgrade dataflow [76]. . . . .	38
4.2	Data stream in Run 2 [78]. . . . .	40
4.3	Comparison of LHCb trigger schemes between Runs 1, 2 and 3 [79]. . . . .	41
4.4	General trend in computer processing performance over time [80]. . . . .	43
4.5	HLT data processing scheme options for Run 3 [79]. . . . .	44
4.6	Side view of the experiment showing the types of tracks that are reconstructed in LHCb [83]. . . . .	45
4.7	Simple diagram of a particle depositing energy in a two pixel cluster as it passes through a sensor at a shallow angle, $\theta$ . . . . .	46
4.8	a) Histogram for primary vertex seeding and b) function for vertex fitting [89].	48
5.1	Remake of Perceptron diagram from Rosenblatt's original paper. [99] . . .	52
5.2	Girl riding a bicycle [103] . . . . .	56
5.3	Unsupervised learning for calculating depth map [105]. . . . .	57
5.4	The Z boson distribution to electron-positron pairs at CMS [11]. . . . .	59
5.5	CNN for classifying particle types from calorimeter images [110]. . . . .	60
5.6	A diagram of a minimal neural network. . . . .	62
5.7	Blue solid lines are a) Sigmoid and b) ReLU activation functions and their derivatives as red dashed lines. . . . .	63
5.8	Railway track workers undertake maintenance on the line [118]. . . . .	66
5.9	Example of a Microsoft Power BI data dashboard [119]. . . . .	67
6.1	View in the $x$ - $z$ plane of a simulated event in the VELO. . . . .	71
6.2	Coordinate frame of the VELO. . . . .	72
6.3	Velo tracking efficiency requirements with respect to momentum [37]. . . .	72

6.4	Pie chart of HLT1 processing time [2]. . . . .	73
6.5	Results of using a neural network to connect closest points. . . . .	74
6.6	Results of toy model, using a neural network to connect hits into tracks. . .	75
6.7	Results of predicting hits in adjacent layers whilst representing the detector as a 128x128 pixel image. . . . .	76
6.8	Results of predicting hits in adjacent layers, whilst representing hits as co- ordinates. . . . .	77
6.9	Results of using DBSCAN for VELO tracking. . . . .	78
6.10	Histograms of results of DBSCAN clustering of VELO hits. . . . .	79
6.11	The stages of the hybrid reconstruction algorithm. . . . .	81
6.12	Diagram of one seed and four target hits on adjacent modules, linked by possible doublets. . . . .	82
6.13	Plot showing how most true target hits are very close in $\phi$ to the seed hit. .	82
6.14	Neural network performance for 1000 events. . . . .	83
6.15	Diagram of the neural network used in the hybrid algorithm. . . . .	84
6.16	Pair selection performance of the neural network. . . . .	85
6.17	small Example of clone tracks from a simulated event, before clone reduction. . . . .	87
6.18	Tracking efficiency of the hybrid algorithm as a function of a) momentum and b) pseudorapidity. . . . .	89
6.19	Results of alignment study. . . . .	90
6.20	Plot showing comparison of using neural network, or only selecting the tar- get hit closest in $\phi$ . . . . .	91
6.21	Plots showing the minimum neural network inference time with respect to the neural network batch size, comparing inference on CPU and GPU. . . .	93
6.22	Pair selection performance of the Graph Neural Network. . . . .	94
6.23	Neural network output performance after retraining. . . . .	95
7.1	Two possible approaches for an End-to-end machine learning model. . . . .	101
7.2	Diagram of FPGA and ASIC image. . . . .	102
7.3	Thesis wordcloud. . . . .	123
7.4	Combination of segments that give clone track, shown in $\phi$ and $z$ . . . . .	126
7.5	Three stages of the hybrid model reconstruction, shown in $\phi$ and $z$ . a) Hits in the detector planes. b) Doublets selected by the neural network, black are true, red are fake. c) Tracks, with clone tracks at dashed lines. . . . .	127



---

## List of Tables

---

3.1	Table of results for the visual inspection of 62 front-end hybrids at CERN. Showing the number and percentage of hybrids that passed each test of the inspection. Less than half of hybrid reached grade A status. Inspections were performed by Phillip Marshall (PM) or Jan Buytaert (JB). . . . .	35
6.1	Table of results for original TrackNN algorithm, for tracks passing the following requirements: At least 3 clusters in the VELO, $p > 5$ GeV, $2 < \eta < 5$ , particle originates from interaction region. Module 50 is the furthest upstream from the interaction region [122]. Definitions of performance metrics are given in 6.5.7. . . . .	80
6.2	Table of results for 2000 minimum-bias events. Our method is compared to the conventional pattern recognition algorithm. . . . .	89
6.3	Table of results inference study. Best times for 100,000 inferences. . . . .	92
6.4	Table of results for 2000 minimum-bias events. Original hybrid method compared to the same method with a retrained neural network. . . . .	95
7.1	Results of Front-end hybrid visual inspection at CERN. 1 indicates a pass, 0 indicates a fail of a test. Operator either Phillip Marshall or Jan Buytaert. . . . .	125

---

## Appendix

---

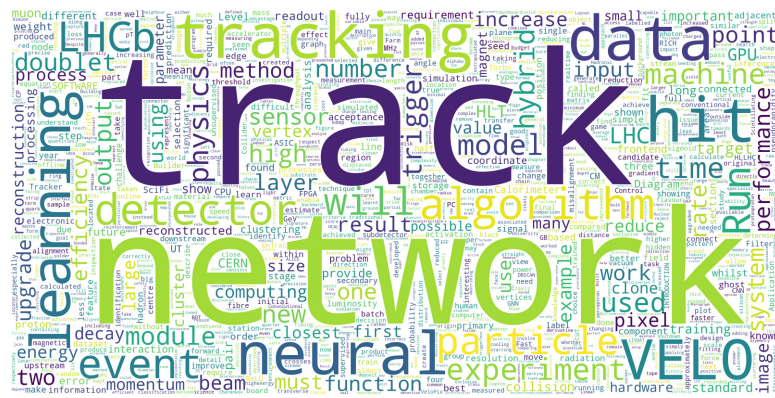


Figure 7.3: Thesis wordcloud.

## Front-End Hybrid Visual Inspection

FEH id	Grade	Operator	Flatness	Traces	Bond pads	Connectors
1	A	PM	1	1	1	1
2	A	PM	1	1	1	1
3	F	PM	1	0	0	1
4	A	PM	1	1	1	1
5	A	PM	1	1	1	1
6	F	PM	1	0	0	1
7	A	PM	1	1	1	1
9	A	PM	1	1	1	1
10	A	PM	1	1	1	1
12	F	PM	1	0	0	1
13	A	PM	1	1	1	1
15	A	PM	1	1	1	1
17	F	PM	1	1	0	1
18	F	PM	1	0	0	1
19	A	PM	1	1	1	1
20	A	PM	1	1	1	1
23	F	PM	1	0	1	1
25	F	PM	1	1	0	1
27	F	PM	1	0	1	1
28	F	PM	1	1	0	1
30	F	PM	1	0	0	1
31	A	PM	1	1	1	1
32	F	PM	0	1	1	1
33	F	PM	1	1	0	1
37	F	PM	1	1	1	0
38	F	PM	1	1	0	1
39	F	PM	1	1	0	1
40	F	PM	1	1	0	0
42	F	PM	1	1	0	1
46	F	PM	1	1	0	1
47	F	PM	1	1	0	1
54	F	PM	1	1	0	1
57	A	PM	1	1	1	1
59	F	PM	1	0	1	1

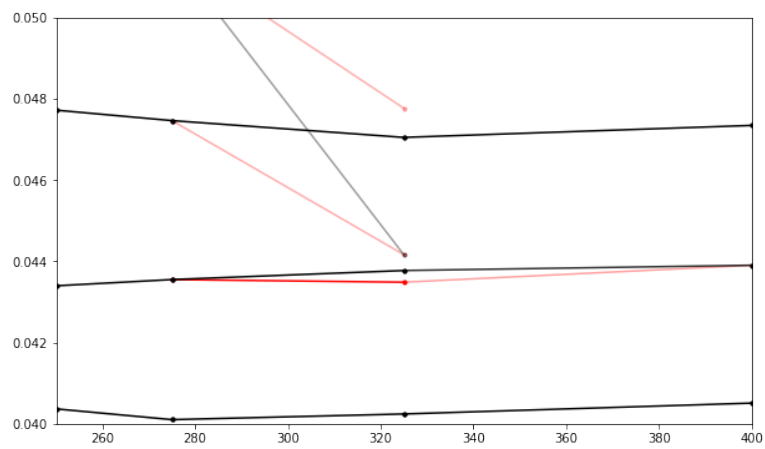
---

61	A	PM	1	1	1	1
62	A	PM	1	1	1	1
63	A	PM	1	1	1	1
64	F	PM	1	1	0	1
65	A	PM	1	1	1	1
66	A	PM	1	1	1	1
67	A	PM	1	1	1	1
68	A	PM	1	1	1	1
71	F	PM	1	0	1	0
74	F	PM	1	1	0	1
76	F	PM	1	1	0	1
78	F	PM	1	1	0	1
79	F	PM	1	1	0	1
8	F	JB	1	1	0	1
21	A	JB	1	1	1	1
22	A	JB	1	1	1	1
24	A	JB	1	1	1	1
29	A	JB	1	1	1	1
36	F	JB	1	1	0	1
41	F	JB	1	1	0	1
43	A	JB	1	1	1	1
45	A	JB	1	1	1	1
48	F	JB	0	1	0	1
53	F	JB	1	0	0	1
69	F	JB	1	1	0	1
70	F	JB	1	1	0	1
72	F	JB	1	0	1	1
80	A	JB	1	1	1	1

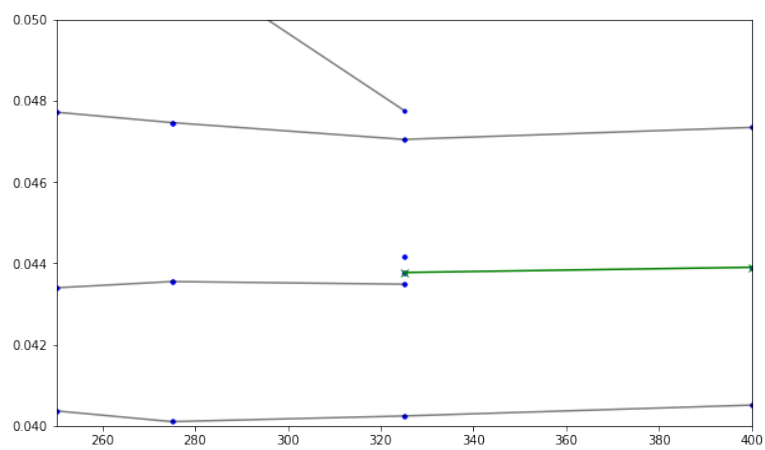
---

Table 7.1: Results of Front-end hybrid visual inspection at CERN. 1 indicates a pass, 0 indicates a fail of a test. Operator either Phillip Marshall or Jan Buytaert.

# Tracking

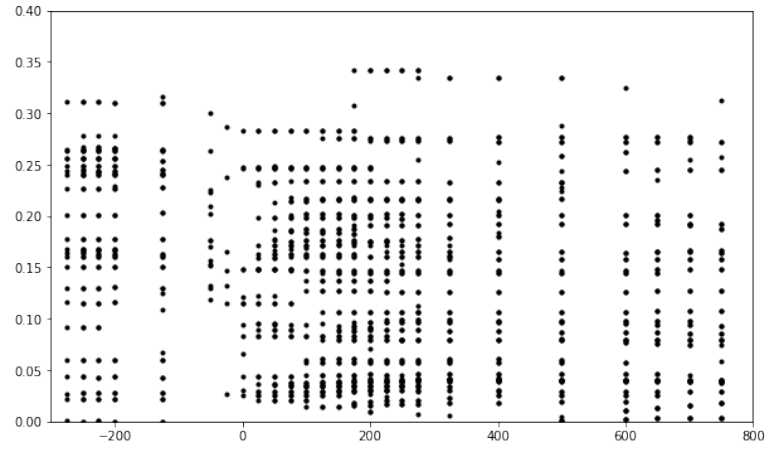


(a)

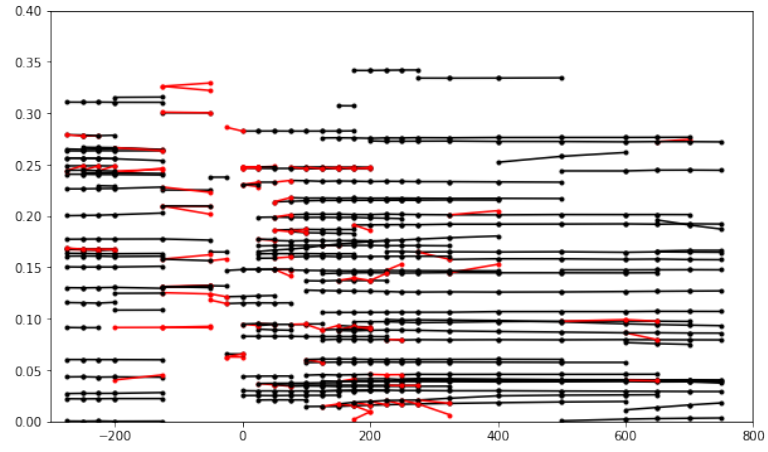


(b)

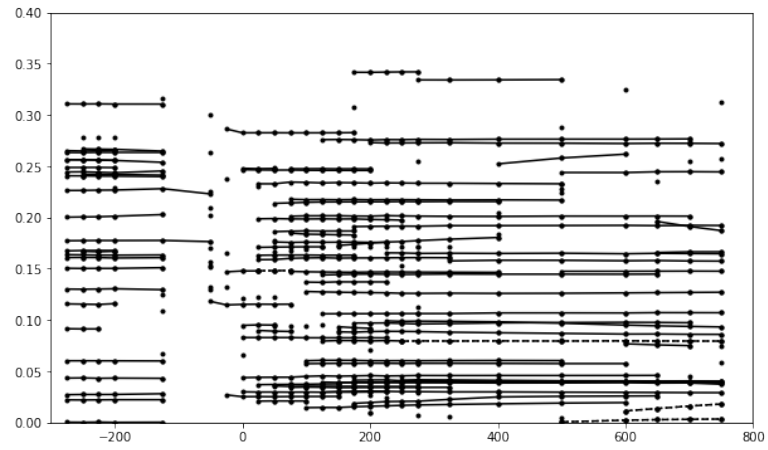
Figure 7.4: Combination of segments that give clone track, shown in  $\phi$  and  $z$ .



(a)



(b)



(c)

Figure 7.5: Three stages of the hybrid model reconstruction, shown in  $\phi$  and  $z$ . a) Hits in the detector planes. b) Doublets selected by the neural network, black are true, red are fake. c) Tracks, with clone tracks at dashed lines.