Department of Physics and Astronomy

University of Heidelberg

Master Thesis in Physics
submitted by
Poppy Hicks
born in Dorchester, UK
2024

Investigations into Momentum-Specialised Graph Neural Networks for Online Track Reconstruction in the ATLAS Event Filter

This Master thesis has been carried out by
Poppy Hicks
at the
Physikalisches Institut Heidelberg
under the supervision of
Prof. Dr. André Schöning

Acknowledgments

I would like to express my gratitude to my supervisors, Prof. Dr. Andre Schoening and Dr. Sebastian Dittmeier, for allowing me the opportunity to work on this project and for enabling my growth as a student.

Abstract

The upcoming High Luminosity upgrade to the LHC poses several challenges, most notably the huge increase in data to process. This necessitates improvements to the Trigger and Data Acquisition systems at the ATLAS experiment, including to its final stage, the Event Filter. Significant effort is being invested into computing R&D for the Event Filter, to keep resources within capacity; one potential area is to utilise the recent gains made in machine learning and highly-parallel architectures such as GPUs. Of those, algorithms based on graph neural networks (GNNs) are most promising for track reconstruction in the Inner Tracker detector.

GNNs demonstrate exceptional capability at modelling complex relationships within graph-structured data. Here, a graph represents detector hits as nodes; edges connecting these nodes represent the possibility the hits belong to the same particle. A GNN is used to score these edges to quantify that probability. This thesis presents an overview of the use of GNNs for track reconstruction in the ITk; a final track efficiency of 0.983 is achieved. The ability of the pipeline to adjust to high transverse momentums is assessed, and an iterative approach to track reconstruction performed. This achieves a final track efficiency of 0.989 for a smaller graph size than the standard GNN approach, thus reducing the memory footprint.

Zusammenfassung

Die bevorstehende Aufrüstung des LHC mit hoher Leuchtkraft bringt mehrere Herausforderungen mit sich, vor allem die enorme Zunahme der zu verarbeitenden Daten. Dies erfordert Verbesserungen an den Trigger- und Datenerfassungssystemen des ATLAS-Experiments, einschließlich seiner letzten Stufe, dem Ereignisfilter. Erhebliche Anstrengungen werden in die Berechnung von R&D für den Ereignisfilter investiert, um die Ressourcen innerhalb der Kapazität zu halten; ein möglicher Bereich ist die Nutzung der jüngsten Fortschritte im Bereich des maschinellen Lernens und hochparalleler Architekturen wie GPUs. Am vielversprechendsten sind Algorithmen, die auf graphischen neuronalen Netzen (GNNs) basieren, für die Rekonstruktion von Spuren im Inner-Tracker-Detektor.

GNNs zeigen außergewöhnliche Fähigkeiten bei der Modellierung komplexer Beziehungen innerhalb graphisch strukturierter Daten. In diesem Fall stellt ein Graph die Detektortreffer als Knoten dar; die Kanten, die diese Knoten verbinden, repräsentieren die Möglichkeit, dass die Treffer zu demselben Partikel gehören. Ein GNN wird verwendet, um diese Kanten zu bewerten, um die Wahrscheinlichkeit zu quantifizieren. In dieser Arbeit wird ein Überblick über die Verwendung von GNNs für die Rekonstruktion von Spuren im ITk gegeben; es wurde eine endgültige Spureffizienz von 0,983 erreicht. Die Fähigkeit der Pipeline, sich an hohe Quermomente anzupassen, wird bewertet, und es wird ein iterativer Ansatz zur Spurrekonstruktion durchgeführt. Dadurch wird eine endgültige Spureffizienz von 0,989 bei einer geringeren Graphengröße als beim Standard-GNN-Ansatz erreicht, wodurch der Speicherbedarf verringert wird.

5

Dedicated to...

Contents

1 Introduction **1**

2 LHC and ATLAS **4**
2.1 The Standard Model of Particle Physics 4
2.1.1 Particle Content 4
2.2 The Large Hadron Collider 6
2.2.1 The HL-LHC Timeline 6
2.3 The ATLAS Experiment 7
2.3.1 The ATLAS Detector 8
2.3.2 ATLAS Phase-II Upgrade 9
2.3.3 Tracking in the EF 12

3 Machine Learning **17**
3.1 Neural Networks 17
3.1.1 The modern neural network 17
3.1.2 Multi-Layer Perceptrons 18
3.1.3 Training 19
3.1.4 Optimizers 20
3.1.5 Regularisation and Normalisation 21
3.2 Graph Neural Networks 21
3.2.1 Graphs 22
3.2.2 The General Framework 22
3.2.3 Message Passing 23

4 ATLAS GNN Tracking **26**
4.1 ACORN Pipeline 26
4.1.1 Graph Construction 27
4.1.2 Interaction Network 28

4.1.3 Building Track Candidates 29

5 Baseline Results **33**
5.1 Simulation Data 33
5.2 Baseline 34
5.2.1 Graph Construction 34
5.2.2 GNN 41
5.2.3 Track Reconstruction 44

6 Iterative Approach **51**
6.1 High Momentum Specialised Graphs 52
6.1.1 Metric Learning 54
6.2 Iterative Approach 58
6.2.1 Stage One 59
6.2.2 Stage 2 69
6.2.3 X = 1.5 GeV 71
6.2.4 X = 2 GeV 71
6.2.5 3 GeV 74
6.3 Overall 75

7 Conclusions and Outlook **78**

Chapter 1

Introduction

High energy experimental particle physics aims to probe fundamental nature at the energy frontier. At this frontier, matter is divided into a collection of elementary particles, and their interactions are governed by fundamental forces controlled by a handful of parameters. Most exist only in state for a fraction of a second - and we aim to study them. This is most successfully achieved by accelerating particles at each other such that they collide and interact: this is employed by all particle colliders but most importantly (for this work) by the Large Hadron Collider. The challenge, then, comes in bridging the gap between the objects of study, physical processes occurring at the length- scales of femto metres (or smaller), and the instruments of measurements, some 15+ orders of magnitude larger.

Two aspects predominantly define modern particle physics: first-principle predictions derived from quantum field theories, and a huge quantity of data. The testing of pre-defined models such as that that discovered the Higg's boson in 2012 is no longer our only interest; we wish now to discover the proverbial "needle in the hay stack", the vanishingly rare signatures that may shed light into areas of physics as of yet unknown.

To study the needle one must first discard the hay: particle physics stands tall as an area with a staggeringly successful theoretical framework in the Standard Model, and as such there are vast areas of phenomena that are understood and therefore uninteresting. Triggering is the real-time jettison of these events to keep only those that might contain signatures of interest, and it is a challenging task. Whilst previously, for model-driven searches, the approach has been to keep only certain classes of events, in the era of asking instead "what might theorists be missing" this is no longer appropriate. And

as we look for rare-r and rare-r events (and thus increase the event rate) the trigger must decide faster and faster.

It is an unfortunate truth that scientific research must always be constrained by budget - and, without innovation, computational resources and thus financial demands soar. Rapid growth outside the scientific community in highly parallel computing architectures and the artificial intelligence which make use of them are one innovative area that have recently taken the world of high-energy physics by storm. A particularly promising avenue is the use of geometrical deep learning methods, well suited for the often sparse nature of particle collider data; this thesis aims to investigate one such area, the feasibility of graph neural networks in triggering for the upcoming era of the LHC.

This thesis is structured as follows: *first*, a brief overview of the theoretical and experimental background in Chapter 2; second, the topic of machine learning and neural nets are introduced in Chapter 3; third, the track finding procedure using graph neural networks is introduced and it's capacity tested in Chapter's 4 and 5; and finally, attempts at minimising the typical memory footprint of this procedure is made in Chapter 6.

Chapter 2

LHC and ATLAS

Particle physics has one of the most successful models describing it in the Standard Model (SM); however, it is not a united, comprehensive theory derived from fundamental assumptions. The path to the fundamental structure of nature lies in exploring phenomena the SM does not explain. The observations we are looking for now are mainly present at high energies; as interaction cross sections decrease quadratically with centre-of-mass energies, these interactions are vanishingly rare. Thus we must also observe more collisions. These are the guiding principles behind the future of the most powerful hadron-hadron accelerator in the world, the Large Hadron Collider (LHC), situated at CERN, the largest physics experiment in the world.

2.1 The Standard Model of Particle Physics

The Standard Model is a Yang Mills theory with $SU(3)_C \times SU(2)_L \times U(1)_Y$ gauge symmetry group that encapsulates our knowledge of the fundamental constituents of matter. It describes three out of four of the fundamental interactions in nature: the strong, the weak, and the electromagnetic forces - and to astonishing accuracy. And it does so with only 19 free parameters. However, despite it's overwhelming success, the model still contains open questions.

2.1.1 Particle Content

All elementary particles in the Standard Model, as a QFT, are described by a quantum field. These particles can be divided into *fermions*, spin-1/2

	Flavour	**Mass**	$Q_{\mathbf{el}}$	T_3^L
Leptons	ν_e	$< 2.2\,\text{eV}$	0	$+\frac{1}{2}$
	e	0.511 MeV	-1	$-\frac{1}{2}$
	ν_μ	$< 0.19\,\text{MeV}$	0	$+\frac{1}{2}$
	μ	106 MeV	-1	$-\frac{1}{2}$
	ν_τ	$< 18.2\,\text{MeV}$	0	$+\frac{1}{2}$
	τ	1776.86MeV	-1	$-\frac{1}{2}$
Quarks	u	2.16 MeV	$+\frac{2}{3}$	$+\frac{1}{2}$
	d	4.70 MeV	$-\frac{1}{3}$	$-\frac{1}{2}$
	c	1.27 GeV	$+\frac{2}{3}$	$+\frac{1}{2}$
	s	93.5 MeV	$-\frac{1}{3}$	$-\frac{1}{2}$
	t	173 GeV	$+\frac{2}{3}$	$+\frac{1}{2}$
	b	4.18 GeV	$-\frac{1}{3}$	$-\frac{1}{2}$

Table 2.1: Table of fermions. Masses given approximately. [1]

	Boson	**Interaction**	**Mass**	$Q_{\mathbf{el}}$
Scalar	Higgs H_0	None	125 GeV	0
Vector	Photon γ	QED	$< 10^{-18}\,\text{eV}$	0
	Gluon g	QCD	0	0
	Z	EW	91.2 GeV	0
	$W^\pm$	EW	80.3 GeV	± 1

Table 2.2: Table of bosons. Masses given approximately. [1].

particles that obey Fermi-Dirac statistics, and *bosons*, integer spin particles that obey Bose-Einstein statistics. These are summarised in Tables 2.1 and 2.2.

Fermions are the constituents of matter, and may be further divided into leptons and quarks: leptons are spin-1/2 particles that do not experience the strong force; quarks are spin-1/2 particles that carry colour charge. Bosons with spin-1, *vector bosons*, mediate the fundamental forces that govern particle-particle interactions. W and Z bosons carry the weak force, the photon carries the electromagnetic force, and the gluon carries the strong force. The scalar Higgs boson, with spin-0, provides mass to elementary particles through interaction with the Higgs field.

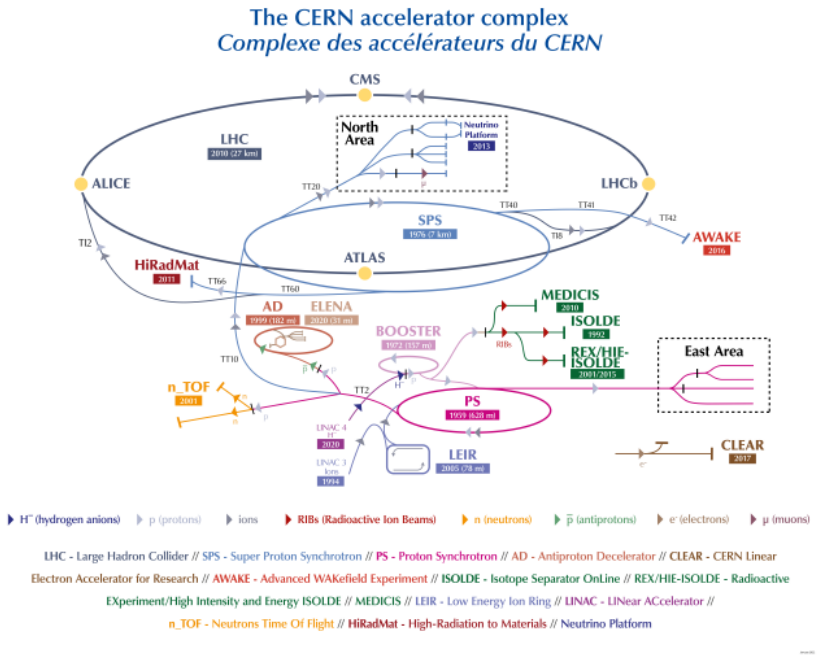


Figure 2.1: The CERN accelerator complex. Figure adapted from [3]

2.2 The Large Hadron Collider

The LHC [2] is a ring-shaped underground synchotron occupying the tunnel previously housing the Large Electron-Positron (LEP) collider, spanning a circumference of 27 km, situated on the French-Swiss border. Operation began (after an accident-prone 2008) in November 2009, with a centre-of-mass collision energy that has been slowly increased to it's current $\sqrt{s} = 13.6$ TeV. The tunnel runs parallel beam pipes which intersect at four collision points around the ring. Each point houses one of the four main experiments of the LHC: ALICE, ATLAS, CMS, and LHCb. A depiction of the complex is shown in Figure 2.1. Rather than a continuous stream, the beams consist of bunches of protons such that collisions occur at discrete intervals - with each bunch-crossing termed a collision *event*.

2.2.1 The HL-LHC Timeline

Since operation began in 2008 the LHC has seen much scientific success, most notably with the detection of the Higgs boson in 2012, the cornerstone of the Standard Model. It has operated consistently on the frontier of high-

Figure 2.2: Baseline upgrade programme to the LHC, as of February 2022. LS refers to a Long Shutdown. Figure adapted from [4].

energy physics (HEP), producing record-breaking proton-proton collisions at first 7, then 8, and now 13 TeV centre-of-mass energy. The inherent setup of the LHC, primarily its radius and bending magnet strength, mean the centre-of-mass energy has now reached its peak; the future of the LHC, with timeline shown in Figure 2.2, is directed now towards increasing it's luminosity. Achieving this goal is at the heart of the technical upgrade to the LHC, to enter the era of the high-luminosity LHC (HL-LHC).

The HL-LHC is scheduled to start operations 2030 with a peak luminosity aim of $\mathscr{L} = 7 \times 10^{34} cm^{-2} s^{-1}$.

2.3 The ATLAS Experiment

The ATLAS Collaboration presented a technical proposal for a *"...General-Purpose pp Experiment at the LHC.."* in 1994 [5]. It was designed to exploit the LHC to full extent, in order to run high precision tests of, among others, QCD, electroweak interactions, and flavour physics [6]. The premier goal, the discovery of the Higgs boson, H, required in particular the ATLAS detector to be sensitive to a wide range of signatures due to the unknown mass of H. Specifications for the ATLAS detector were defined by considering both the set of processes that would further our physics goals and the nature of the LHC. The processes are rare: in proton-proton collisions, QCD jet production cross-sections dominate. The identification of experimental signatures characteristic of the processes is challenging and imposes further demands on the required luminosity and the particle-identification capabilities. The requirements were as follows:

- Detectors to be made of fast, radiation-hard electronics and sensor elements
- Large acceptance in pseudorapidity with almost full azimuthal angle

coverage

- Good momentum resolution and reconstruction efficiency in the inner tracker

2.3.1 The ATLAS Detector

An overview of the ATLAS detector [7] is shown in Figure 2.3. The magnetic configuration is comprised of an inner superconducting solenoid around the inner detector cavity, and large superconducting toroids surrounding the calorimeters. The *Inner Detector*, closest to the interaction point and immersed in a solenoidal magnetic field of 2 T, is contained within a cylinder of length 6.80 m and radius 1.15 m, and measures only the trajectories of charged particles. Surrounding the ID are the *calorimeter systems*: this is responsible for precise energy reconstruction and measurement of all particles other than the muon. It reconstructs particles originating from the hard interaction vertex by initialising a cascade of secondary particles: a particle shower. Particle showers can be categorised as electronic or hadronic, with strongly dependent features, and as such the ATLAS detector has both an electromagnetic (for reconstructing electrons or photons) and hadronic calorimeter (for measuring showers involving the strong interaction). Finally, the ATLAS detector has the *muon spectrometer*: dedicated solely to the measurement of muons. All other particles are assumed to have been absorbed by the preceding components.

Triggering and Data Acquisition

The Trigger and Data Acquisition (TDAQ) system at ATLAS is responsible for all online processing of detector data. It's job is to select and transfer to storage events judged to contain interesting signatures for the physics programme, and to discard all others. In Run 3 [8], it is formed of a hardware-based first level trigger, L1, and a software-based high-level trigger, HLT. The L1 trigger, using information from the muon and calorimeter systems, searches for high p_T muons, electrons, photons and jets, and also for large or missing transverse energy, and reduces event rate from the bunch crossing rate of 40 MHz to 100 kHz. Muons are identified using information from the barrel and end-cap regions of the spectrometer; calorimetry-based selections use information from all calorimeters. Regions of Interest, *RoIs*, regions

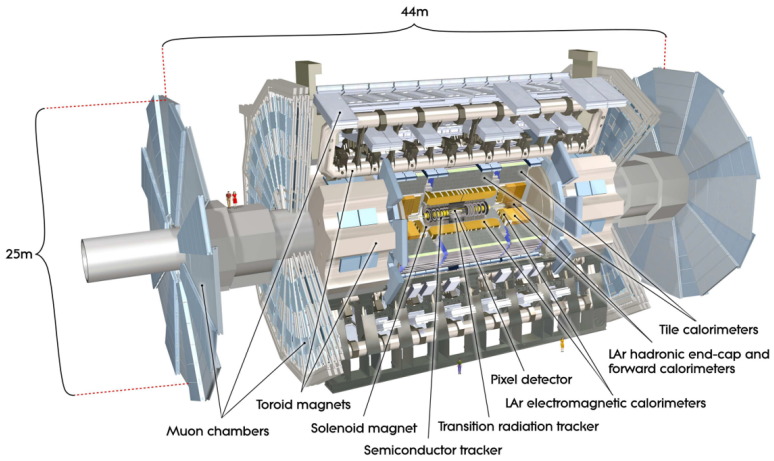


Figure 2.3: Cut-away view of the ATLAS detector, from [6].

within the detector with interesting features, are also identified and passed on. Events passing the L1 trigger selection are transferred to the HLT which use full detector granularity and information with tracking to further reduce event rate to 3 kHz.

2.3.2 ATLAS Phase-II Upgrade

The Phase-II upgrade for ATLAS is required to enable the ATLAS physics program planned for the HL-LHC: namely precision measurements of the properties of the Higgs boson to study electroweak symmetry breaking, improved measurements of SM parameters, and searches for BSM signatures and flavour physics [9]. The nominal luminosity of the HL-LHC, $\mathscr{L} = 5 \times 10^{34} cm^{-2} s^{-1}$, with additional ability to reach $\mathscr{L} = 7.5 \times 10^{34} cm^{-2} s^{-1}$ (operating at the LHC's *ultimate configuration*), requires the experiment to cope with pile-up of up to 200 inelastic collisions per bunch-crossing. This corresponds to a total integrated luminosity in the range of $3000 - 4000 fb^{-1}$ by the end of Run 6, which (at the lower estimate) is an order of magnitude higher than collected prior to the upgrade. The upper estimate of $4000 fb^{-1}$, i.e. the LHC operating at ultimate configuration with pile-up of 200, is what the Phase-II upgrade is designed to facilitate.

The overall upgrade strategy of ATLAS can be divided into four main elements; the ITk Pixel [10] and Strip [11] Detectors, the LAr and Tile Calorimeters, the Muon Spectrometer, and TDAQ [12], [9]. The discussion

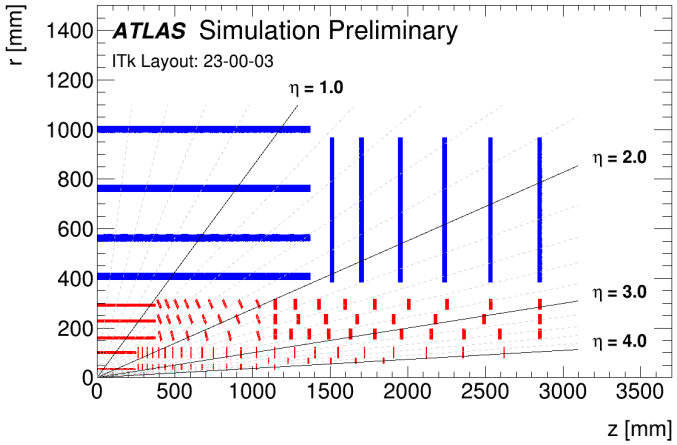


Figure 2.4: Schematic depiction of one quadrant of the ITk for ATLAS Phase-II upgrade. The z-axis is along the beam line with zero being the interaction point. The r-axis is transverse distance from the interaction point. Active elements of the strip detector are shown in blue, active elements of the pixel detector shown in red. Figure from [13].

of the workings of ATLAS below will incorporate the expected upgrades.

Inner Tracker

The anticipated increase in luminosity of the HL-LHC and the radiation damage accumulated during the high centre-of-mass energy of Run 3 render the ATLAS Inner Detector non-functional for future Runs. It will be replaced with a new all-silicon tracker, the Inner Tracker (ITk), to maintain the tracking performance of Run 3 whilst operating with the higher-occupancy environment and coping with the increased integrated radiation dose. A schematic view of one quadrant of the ITk is shown in Figure 2.4. It consists of pixel layers close to the interaction point (depicted in red), and strip layers at larger radii (in blue). The pixel and strip detector modules form a system of cylindrical layers in the central detector region, the *barrel*, and a system of rings beyond, called the *endcap*.

- **ITk strip detector:** the strip sub-detector consists of a four-layer barrel region and one end-cap with six disks on each side. It covers pseudorapidity of $|\eta| = 2.7$ The two inner barrel layers are equipped

with short strips of 24.1 mm length; the two outer barrel layers with long strips of 48.2 mm length. Each barrel layer is 2.8 m long.

- **ITk pixel detector:** the pixel sub-detector consists of a five-layer barrel region and four end-cap ring layers. It covers pseudorapidity of $|\eta| = 4$. In the end-cap region, the positions of the pixel rings along the z-axis are optimised to provide hermetic coverage.

The ITk provides highly granular, hermetic coverage with at least nine points in the barrel region, and 13 in the end caps.

LAr and Tile Calorimeters

The readout electronics of both the LAr and tile calorimeters must be upgraded to cope with the increased radiation, trigger rates, and latencies that of Run 4.

Muon Spectrometer

Primarily the performance of the muon trigger chambers must be improved.

T-DAQ

The Phase-II upgrade of the ATLAS T-DAQ system must facilitate the planned physics program of ATLAS in Run 4 and beyond whilst operating under the conditions of the HL-LHC. Exceptional performance is required to cope with increased pile-up: calorimeter energy resolution is reduced whilst the rate of interesting events is increased. Tracking, a key component of the trigger, becomes more challenging due to the higher hit density in the detector.

The baseline design for Phase-II was a single Level-0 hardware trigger, with input from calorimeter and muon spectrometer systems, with a trigger rate of 1 MHz and a maximum latency of 10 μs; there was potential evolution into a two level trigger that would include hardware-based tracking. This has been updated to reflect refinements to the design of the ITk, with novel approaches to track reconstruction and advances in commercial accelerators also informing the decision. An independent ATLAS committee re-evaluated the future of Event Filter (EF) tracking and recommended *"TDAQ should continue investigating using hardware accelerators to optimize the EF farm.*

The Heterogeneous commodity task force has largely demonstrated proof-of-concept, and a heterogeneous solution (including FPGAs and/or GPUs) could lead to substantial power and cost savings." [12].

The planned T-DAQ Phase II architecture is shown in Figure 2.5. The Level-0 system (in purple) processes muon and calorimeter data at 40 MHz, the frequency of the bunch crossing. This triggers at 1 MHz the read out of all detectors, which send data through the DAQ system to the Event Filter.

The Event Filter performs regional tracking on events in RoIs identified by the L-0 trigger. This is used to perform an initial trigger selection, and full-scan tracking is then run over the entire ITk detector at a rate of 150 kHz, to transfer to permanent storage at a rate of 10 kHz.

2.3.3 Tracking in the EF

Track reconstruction in the Event Filter follows a three step procedure:

- **clustering and space point formation** of hits from the raw ITk data
- **seeding and pattern recognition** takes a subset of hits to find likely track candidates
- **track extension, fitting, and ambiguity resolution** extends track seeds into complete track candidates, utilises algorithms for duplicate removal, fake rejection, and the resolution of ambiguous track candidates, and determines track parameters with a high precision fit

Clustering and space point formation

Hits are formed from the Athena space point formation algorithm [14]. A connected components analysis groups channels with a common edge or corner and where the deposited energy yields charge above a threshold into clusters. These are used to construct three-dimensional objects, *space points*, which represent where the charged particle traversed the active material of the ITk. Each cluster in the pixel sub-detector equates to one space point. Within the strip sub-detector, the procedure is more complex: a charged particle crossing the module outputs two one-dimensional measurements (instead of one two-dimensional measurement as in the pixel) that must be combined into

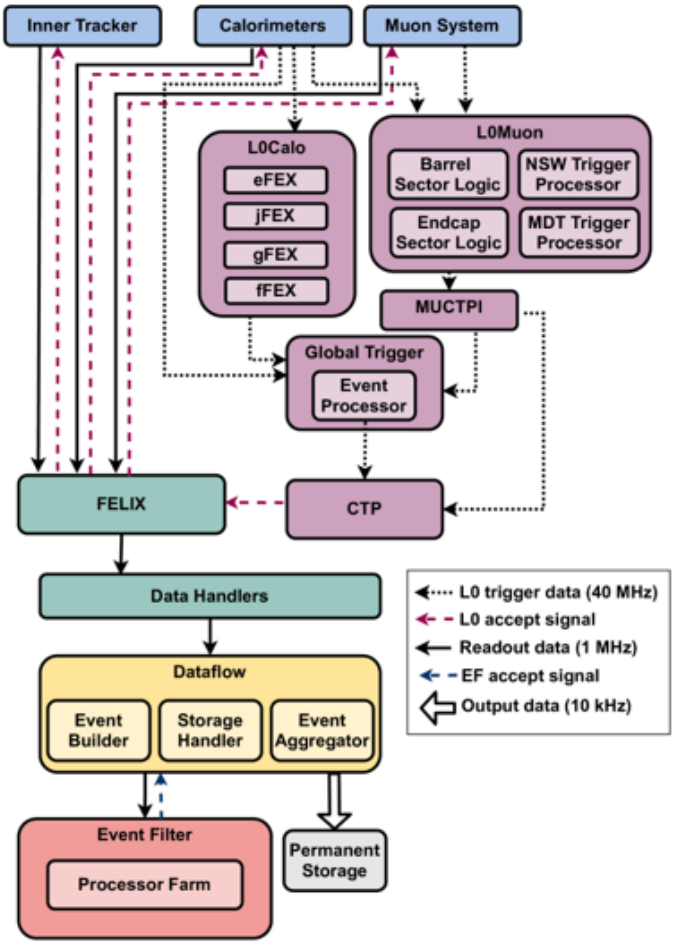


Figure 2.5: A schematic of the TDAQ Phase-II architecture, taken from [12]. The black dotted arrows indicate the Level-0 dataflow from detector systems to the L-0 trigger system; the L-0 trigger decision, indicated by the red dashed arrows, is transmitted to the detectors. Trigger and detector data, shown by the solid black arrows, pass through the Data Acquisition System and into the Event Filter, which reduces event rate to 10 kHz. Events selected by the EF trigger are transferred to permanent storage.

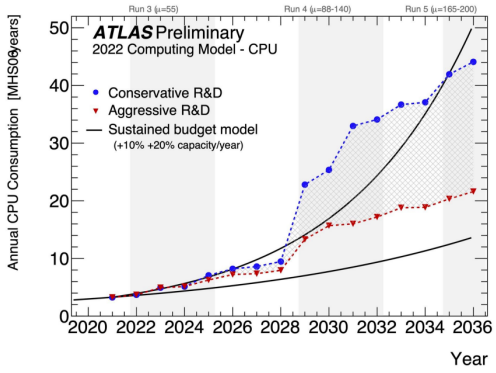


Figure 2.6: Estimated CPU resources required for 2020 to 2036 under different scenarios. The two solid lines indicate improvements in the capacity of hardware annually by 10 and 20%, assuming a sustained budget. The blue dots indicate the conservative R&D scenario, the red dots the aggresive R&D scenario. Vertical shaded bands are periods ATLAS will be collecting data. Figure from [15].

a two-dimensional measurement using the stereo angle between the silicon strips on opposite sides of the module.

Computational Challenges

Even with the upgraded Phase-II detector systems, dealing with the increased event complexity in the HL-LHC era poses a significant computational challenge. All aspects of event reconstruction but, in particular, track reconstruction require significant CPU resources. Projections run to inform resource requirements during the HL-LHC show that, assuming a flat budget, needs will not be met solely by improvements to computing hardware. Figure 2.6 shows resource projections for conservative and aggressive R&D scenarios: future shortcomings are evident unless significant progress is made.

Experience from Run 3 has showed tracking dominates CPU resources, [8]; it also indicates that it scales worse-than-linearly with pile up, as seen in Figure 2.7. Significant effort is therefore being invested into the reduction of computing resources for tracking, seeking improvements to existing algorithms and the development of new ones. A promising avenue is the use of machine learning algorithms that can utilise accelerators such as GPUs and FPGAs. Applications of ML algorithms in high-energy physics have soared in the last decade; in particular the development of geometric deep

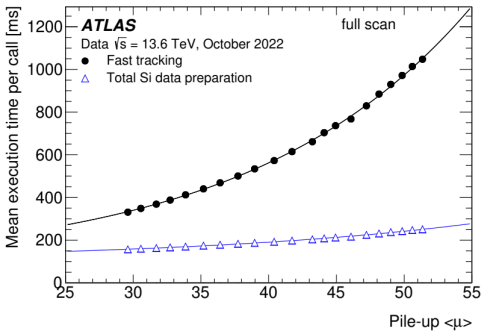


Figure 2.7: Execution time for fast tracking and silicon data preparation time for full scan tracking in jet triggers in Run 3. Figure from [8].

learning, which allow non-Euclidean inputs, are well suited for the sparse samplings common from particle physics experiments. Set- and graph-based architectures have been utilised in particle physics in the last few years for a huge variety of tasks, including jet classification, event classification, vertexing, pile-up mitigation, and, crucially, charged particle tracking (reviewed in [16]). Deep learning as a field offers proven efficacy in HEP tasks: it also offers a natural way to switch to the highly parallel architectures of GPUs and FPGAs, the clear future of computing power.

In particular, graph neural networks (GNNs) have been demonstrated to be effectively utilised in charged particle tracking at the HL-LHC (first by S Farrell et al. [17]), with excellent performance seen in, for example, [18],[19].

Chapter 3

Machine Learning

Machine learning has helped shape experimental particle physics since the 1980s [20]; indeed, the 2024 Nobel laureates in Physics were awarded to two pioneers of the subject. Unlike other experimental techniques, deep learning algorithms extract high-level patterns using lower-level information directly from the data. Here, an overview of the basics of machine learning, and more specifically deep neural networks, is given - with an emphasis on techniques that will be used later in this work.

3.1 Neural Networks

3.1.1 The modern neural network

A neural network can most simply be thought of as a numerically defined fit function with model parameters θ,

$$f_\theta(x) \approx f(x) \tag{3.1}$$

which converts input vector $x \in \mathbb{R}^D$ to some output $f_\theta(x)$.

Neural networks are composed of a set of connected neurons; the strength of connection between two neurons are determined by its *weight*, such that a weight of zero would cut the connection. The input function x is propagated along neurons to reach network output $f_\theta(x)$; most simply, this propagation from neurons $\{j\}$ to neuron i uses the *affine transformation*,

$$h_i = W_{ij} h_j + b_i \tag{3.2}$$

where the output h_j of neuron j is connected to neuron i via weight W_{ij} and the learnable *bias* b_i. A propagation defined exactly as this would allow the neural network to approximate solely linear functions, as the composition of linear functions is always a linear function; non-linearity is introduced to increase network expressivity via an *activation function*, which further processes the propagation output. The most simple activation function is the *rectified linear unit*, ReLU,

$$ReLU(x_j) := \max(0, x_j) = \begin{cases} 0 & \text{if } x \leq 0, \\ x & \text{if } x > 0. \end{cases} \tag{3.3}$$

Other common activation functions include the *Sigmoid*,

$$S(x) = \frac{1}{1 + \exp^{-x}} \tag{3.4}$$

and the *Tanh*,

$$\tanh x = \frac{\exp^{x} - \exp^{-x}}{\exp^{x} + \exp^{-x}}. \tag{3.5}$$

3.1.2 Multi-Layer Perceptrons

The multi-layer perceptron (MLP) is among the most common form of neural networks. Neurons (hence called nodes) are grouped into a set of layers; the nodes of layer i are connected to nodes of layers $i \pm 1$ with the strength of connection determined, as before, by its weight. The layers are separated into the three categories: the *input layer*, with n nodes corresponding to the n-dimensional input vector x_k; the *output layer*, which returns the transformed input data $y_j = f_\theta(x_k)$; and all other interim layers, known as the *hidden layers* - so-called because the computations performed are not immediately visible to the user, and whilst the model parameters can now be tracked it can be difficult to conceptualise the patterns they represent. The structure of a multi-layer perceptron containing only one hidden layer (known as shallow) is shown in Figure 3.1.

The complexity and expressivity of the network are partially determined by it's depth (the number of hidden layers) and width (the number of nodes in each hidden layer). The propagation function (now with some activation function Φ) from layer n to layer $n + 1$, can now be written as

$$h_i^{(n+1)} = \mathbf{\Phi}[W_{ij}^{(n+1)} h_j^{(n)} + b_i^{(n+1)}] \tag{3.6}$$

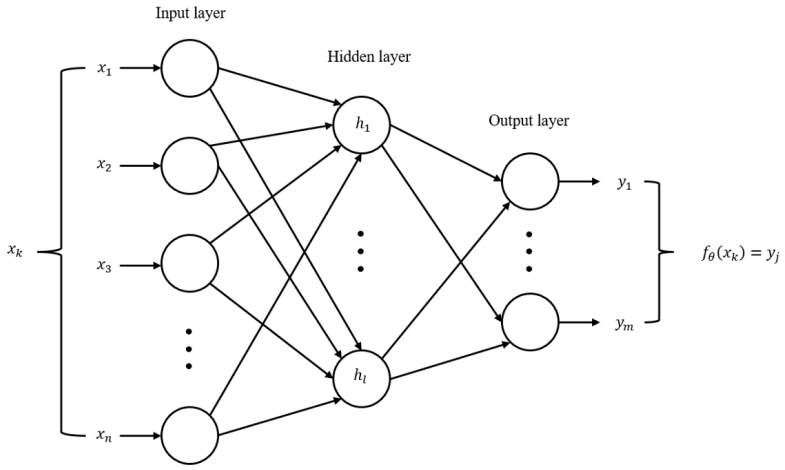


Figure 3.1: A shallow multi-layer perceptron, with n inputs, m outputs, and one hidden layer comprised of l nodes.

Encoders and Decoders

Encoders and decoders are vital components of many larger neural net systems, and are generally formed of MLPs. An encoder maps input data into a high-dimensional latent space representation, capturing abstract features; a decoder transforms this latent space representation into a task-specific output.

3.1.3 Training

The previous sections have introduced the basic concepts of neural networks, as well as some relevant topologies. This section will introduce the concepts behind network learning.

For the purposes here, we are concerned only with *supervised learning*; this occurs when both the input x and the truth $f(x)$ of the training data are known. This is in contrast to *unsupervised learning* where the truth is always unknown.

A training dataset $(x, f)_j$ gives implicit access to the truth $f(x)$; our model parameters θ must be determined such that the our network output $f_\theta(x)$ approximates the truth,

$$f_\theta(x) \approx f(x) \tag{3.7}$$

The discrepancy between these functions is measured using a *loss function*; this is used to update the model parameters θ each iteration, to improve the prediction of the next. The loss function, $\mathscr{L} \geq 0$, is an encapsulation of the network's approximation error; a smaller loss corresponds to a better approximation.

The choice of loss function must be inspired by the task at hand; a simple classification task may use a cross entropy, whilst the classic tool for a regression task is the mean squared error. The loss functions pertinent to this thesis will be discussed later.

The pursuit of a neural network is to minimise the loss by applying the correct configuration of model parameters. This is done iteratively, as follows:

Training Step

Using our training dataset $(x, f)_j$,

1. Forward pass: x_j is propagated forward through the network to produce some output $f(x_j)$; this output is compared to the truth f_j via some loss function.

2. Backward pass: The network is propagated through backward; at each node, the node's parameters are updated to minimise the loss function.

Validation Step

A forward pass is completed using a validation dataset; the loss is computed and logged to check for overtraining. No backward pass is completed.

3.1.4 Optimizers

Optimizers are used to update model parameters during the backward pass using the loss function; typically the optimizer of choice is the *gradient descent* algorithm. One must first calculate the loss functions derivative with respect to that node's parameters; the parameters are now updated by 'taking a step' in the direction of the negative gradient; a naive approach could be

$$\theta_j^{(t+1)} = \theta_j^{(t)} - \alpha \left\langle \frac{\partial \mathscr{L}^{(t)}}{\partial \theta_j} \right\rangle \tag{3.8}$$

The step size (as in the second term of Eq. 3.8) is determined by the *learning rate*, α, a hyperparameter of the model. Consider a high-dimensional loss landscape $\mathscr{L}(\theta)$. Far away from the minimum, the gradients are unreliable and large "steps" more efficiently search the global landscape - and avoid getting trapped in local minima. Approaching the minimum, smaller steps are wanted so as not to miss it. For this reason, the learning rate is often scheduled as an exponential decay.

This walk through loss landscape is inherently unstable, and whether the global minimum has been reached is unknown (in fact in many cases it has not). There are some methods of stabilisation: for instance, *momentum* mixes the loss gradient at the most recent step with gradients from updates before; the vector travelling through parameter space incurs acceleration from the gradient of the loss, and therefore changes in direction are dampened. Another widely used optimizer is the *Adam* optimizer, introduced in [21], which builds on momentum with adaptive learning rates: by adjusting the learning rate for each parameter using estimates of the first and second moments of the gradients, it improves convergence stability.

3.1.5 Regularisation and Normalisation

Normalisation is a technique used to mitigate problems encountered when training neural networks such as over-fitting. Normalisation techniques work to stabilise gradients, ensure inputs to subsequent layers maintain a consistent scale and distribution, and reduce the effect of outliers in the data

- **Batch normalisation** normalises the inputs of each layer within a mini-batch.
- **Layer normalisation** normalises the activations of each sample of a neural net layer: normalisation is thus performed across the features of each sample

3.2 Graph Neural Networks

Much data in the real-world can be represented as graphs; think, for instance, of social networks, molecular structures, or - for this thesis - events in high-energy physics. Until recently, using deep learning models on these datasets required converting them to Euclidean structures such as sequences

or images, and in the process masking some relational information. The rise of geometric deep learning as a sub-field of machine learning has enabled the more traditional structures of neural nets to be adjusted to support non-Euclidean inputs. Graph neural networks (GNNs) are one such area; networks designed to facilitate a graph-like input and exploit it's relational structure. GNNs have been used in many areas of particle physics already.

3.2.1 Graphs

Any set of objects with relations between them can be expressed as a graph, with the objects as nodes and the relations as edges connecting those nodes. These edges can have direction (as for a *directed* graph), or can be directionless (an *undirected* graph). Graphs thus have four features that may want to be used to direct predictions: node-level, edge-level, global, and connectivity. The last, in particular, makes graph representation difficult.

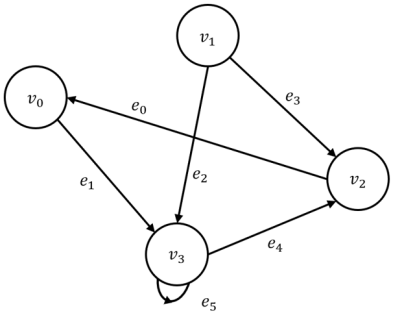
Typically, a graph is represented by an *adjacency matrix*: an $n_{nodes} \times n_{nodes}$ matrix whose (i, j)th element indicates whether the node i is adjacent (i.e. connected) to node j. As most graphs are far from fully-connected, the adjacency matrix is generally sparse and therefore space-inefficient. Adjacency matrices are also not permutation invariant, and for undirected graphs are symmetric, containing redundant information.

One solution to this is to instead represent graphs as a $2 \times n_{edges}$ adjacency list: these describe the connectivity of the edge e_k between nodes i and j as a tuple (i, j) in the kth element of the list. Unlike adjacency matrices, this format is permutation invariant, and in general is a more concise representation. Figure 3.2 shows an example of a directed graph, and both forms of representation.

3.2.2 The General Framework

Fundamentally, GNNs are graph-to-graph mappings that ascertain higher-level information by transforming features whilst preserving connectivity. Similarly to an MLP, a GNN has a layered structure; information between connected nodes is transformed from layer to layer.

Assume we have as our input a graph with n nodes, and input node feature vectors $\{x_1, x_2, ..., x_n\}$. These node features are embedded into a latent space, with each layer of the GNN transforming the embeddings, such



$$A_{matrix} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

$$A_{list} = \begin{pmatrix} 2 & 0 & 1 & 1 & 3 & 3 \\ 0 & 3 & 3 & 2 & 2 & 3 \end{pmatrix}$$

Figure 3.2: A directed graph (left), it's adjacency matrix (top right), and it's adjacency list (bottom right). The edges are not weighted.

that the representation of the ith node in the kth layer is given by $h_i^{(k)}$. The nodes within each layer of the neural net is inherited from the structure of the input graph. Connections are also determined by the input graph: *self* connections exist between the same node on adjacent layers, and *neighbour* connections exist between neighbouring nodes on adjacent layers. These connections ensure that the embedding in layer k of node i in layer $k+1$ is determined by the embedding of both node i and by it's nearest neighbours. This aggregation of information is called *message passing*, and is discussed in the next section. An example of a graph used as the basis for the structure of a GNN is shown in Figure 3.3.

The input layer to the net is simply the input graph, with feature vectors $\{x_i\}$ taken as embeddings $\{h_i^{(0)}\}$. The output layer is dependent on the type of neural net in use.

3.2.3 Message Passing

Message passing, then, is the core mechanism used to propagate information across the graph to collate at each node. This concept was developed as an *interaction network* for physics in [23]. The number of hidden layers (referred to from now as the number of message passing steps) in the GNN dictate what information is gathered from what distance node to update the embedding; the greater the number of message passing steps, the greater the reach across the graph.

Take, as an example, Figure 3.3: in the first hidden layer, node v_0 will collate information from itself and v_2; v_2 will collate information from itself,

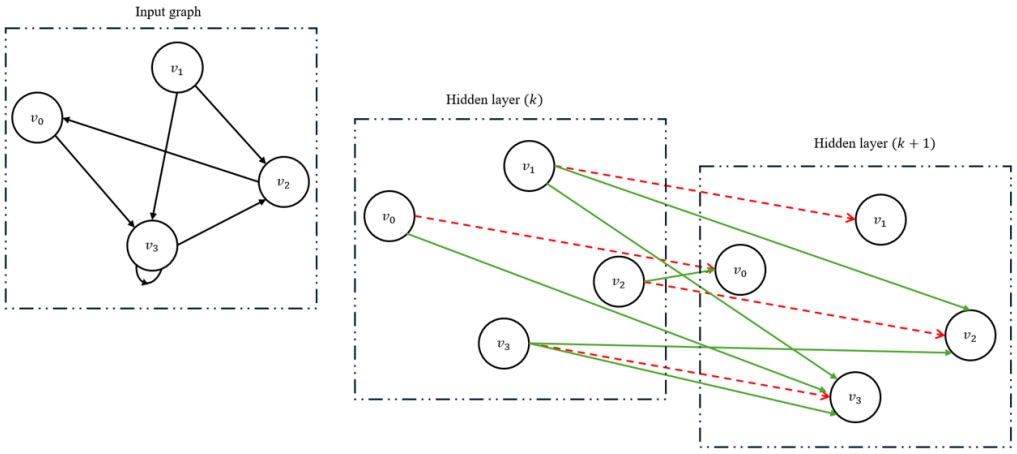


Figure 3.3: The directed graph from Figure 3.2, and an example of the connections between adjacent layers in a GNN architecture. Dashed red arrows show self connections; solid green arrows show neighbour connections. Figure inspired by [22].

v_3, and v_1. Thus, in the second layer, v_0 will receive again information from v_2, but this will now contain information from v_1 and v_3. In that way, in only two message passing steps, information from the entire graph has informed the embedding of node v_0.

Chapter 4

ATLAS GNN Tracking

Here the ACORN track reconstruction framework, developed for charged particle track reconstruction in the ITk detector at ATLAS, will be laid out. The following algorithms aim to reconstruct all non-electron particles produced in the primary vertex of top quark pair production, with transverse momentum $p_T > 1$ GeV, produced at transverse radius $r < 260$ mm and $|\eta| < 4$, that leave at least 3 hits. From here on particles meeting these conditions will be referred to as *target particles*.

4.1 ACORN Pipeline

For each event, a graph is built. Each space point (or hit) is taken as a node, with attached features; edges are constructed between nodes such that an edge connecting two nodes forms a doublet of hits. Pair-wise features can also be associated with each edge. An edge connecting two nodes represents the possibility that the two nodes are successive hits from the same particle; all edges can thus be categorised as true, if that is indeed the case, or fake, if not. By ranking the probability each edge is true using some edge-scoring algorithm and then in some way selecting edges with high scores, the graph can be segmented into components of connected nodes: these are taken as the track candidates. The GNN-based pipeline for track reconstruction is shown in Figure 4.1.

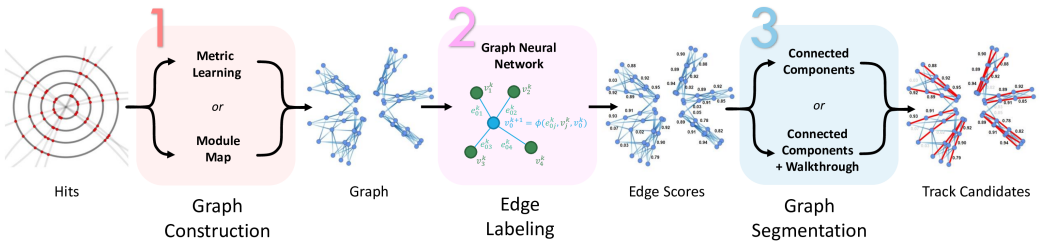


Figure 4.1: Schematic overview of the GNN-based track finding pipeline. Taken from [24].

4.1.1 Graph Construction

An event with pileup $\langle\mu\rangle = 200$ has $\mathcal{O}(10^5)$ space points; a fully connected graph would have $\mathcal{O}(10^{10})$ edges, with the vast majority of edges representing non-physical connections and thus redundant. When building the graph, the choice of which edges to construct is the premier question. To reconstruct all tracks of target particles, all edges connecting successive hits of target particles, *true target edges*, must be built, whereas all other edges are preferably avoided due to time and GPU-memory constraints; additional fake edges can also obscure the tracks we wish to reconstruct.

These requirements are characterised by two performance metrics: edge-wise efficiency, the ratio of true target edges to all true target connections, and edge-wise purity, the ratio of true target edges to all edges present in the graph. An efficiency close to 100% is required to avoid broken track candidates; purity demands thus take a lesser priority.

There are two approaches to graph construction commonly used for track finding: a data-driven approach, the module map, which applies geometric heuristics to determine which space points are likely to belong to the same track (and thus should be connected), and the approach used in this thesis: the metric learning approach.

Metric Learning

Metric learning, [25], is a machine-learning technique that learns a distance function $\tilde{d}(x_i, x_j)$ that quantifies how similar data points (x_i, x_j) are in a feature space. Here, the metric function of the form

$$\tilde{d}(x_i, x_j) = d\left(f(x_i), f(x_j)\right) \tag{4.1}$$

is learnt, where $f : \mathbb{R}^n \to \mathbb{R}^m$ is a multi-layer perceptron that embeds data points (x_i, x_j) into a high-dimensional latent space; the distance metric d is then the euclidean distance within that latent space.

The network is trained using a contrastive hinge loss,

$$\mathcal{L}(x_i, x_j) = w_{ij} y_{ij} \tilde{d}(x_i, x_j) + w_{ij} (1 - y_{ij}) \max[0, m - \tilde{d}(x_i, x_j)] \tag{4.2}$$

where $y_{ij} = 1$ if x_i and x_j are consecutive space points from the same particle and equals 0 otherwise. The hinge loss thus pulls true pairs ($y_{ij} = 1$) ever closer together and pushes fake pairs ($y_{ij} = 0$) at least a margin m away. The weights, w_{ij}, allow the model to enhance the impact of certain training data pairs and mask others.

The trained MLP is used in inference to embed space points into the latent space. An adaption to the k-Nearest Neighbours (kNN) algorithm is then used: an m-dimensional sphere of radius r is centred on each hit and the first k hits within that sphere are connected to it. This thus builds connections in the graph in event-space.

Filter

The metric learning approach creates graphs with $\mathcal{O}(10^7)$ edges. This is an order of 10 greater than graphs created with the Module Map approach, so an additional filtering step to prune fake edges is done, to obtain graphs with $\mathcal{O}(10^6)$ edges. Graph convolution modules embeds the node input and undergoes message passing steps to learn graph-wide features before passing to an MLP that outputs an edge classification score.

The network is trained using a binary cross entropy loss,

$$\mathcal{L}(x, y) = \sum l_n; l_n = -w_n[y_n \cdot \log \sigma(x_n) + (1 - y_n) \cdot \log(1 - \sigma(x_n))] \tag{4.3}$$

where $y_n = 1$ for true edges and equals 0 otherwise, and the weights, w_n, allow the model to enhance the impact of certain training edges and mask others.

4.1.2 Interaction Network

The graph neural network used here takes the trimmed down graph post filter as it's input, and follows an encode-process-decode architecture to output

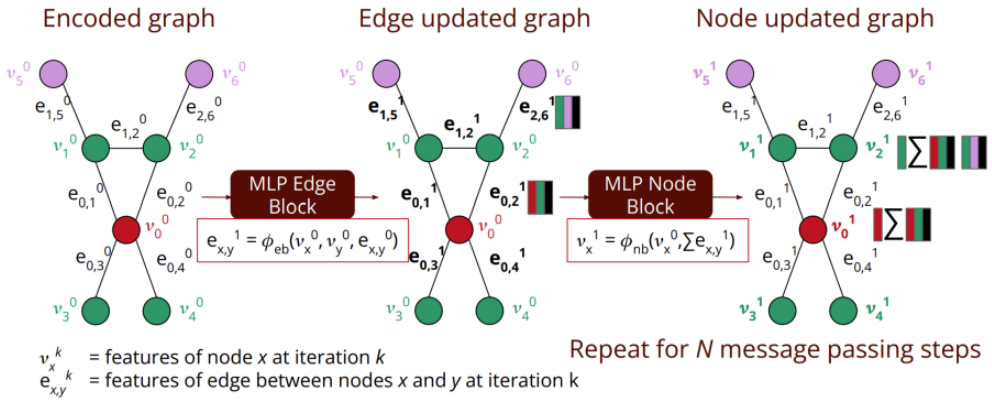


Figure 4.2: A schematic overview of one message passing step in the interaction network. Figure taken from [26].

an edge-classification score ranking the probability an edge is target true. Initially, two encoders (one for node features, such as position, and one for edge features, such as positional differences) embed features into a high-dimensional latent space.

Within the latent space, message passing steps propagate features around the graph, as discussed in Section 3.2. Node latent features are passed to all joining edges; an edge network MLP uses these features to update the latent representation of that edge via some permutation-invariant aggregation with the prior edge feature. The updated edge features are then passed to all connected nodes, where a node network MLP aggregates them with the prior node feature to update it. This sequence comprises one message passing step, and is illustrated in Figure 4.2. Iterations of message passing steps propagate information across the graph, enabling the identification of complex geometric patterns in tracks.

Finally, the latent edge features from the last message passing step are decoded into an edge classification score.

The network is trained with the binary cross entropy loss function.

4.1.3 Building Track Candidates

The scored graph must now be segmented into track candidates.

ConnectedComponents

The most straightforward method is to apply a score threshold and cut all edges beneath that score. Each remaining set of connected nodes, or *connected components*, is taken as a track candidate. This is extremely time efficient and computationally inexpensive, but merges tracks if the threshold is too low and splits tracks if the threshold is too high.

CCandWalk

A more complex method combines the connected components algorithm with a walkthrough algorithm, and requires a directed graph. First, any directed cycles are removed by pointing all edges outwards (using the hit-position of the nodes). An initial low score cut is applied to again create sets of *connected components*. If each node in a connected component has at most one incoming and one outgoing edge (i.e. there is a single path), the component is labelled as a track candidate. If multiple paths exist, a walkthrough algorithm is used.

A topological sort on the graph ascertains that node i appears before node j if a connection $i \rightarrow j$ exists (hence the requirement for an a-cylic graph). This is used to identify nodes without incoming edges: *source nodes*. Given one of these source nodes, the walkthrough algorithm commences.

1. The outgoing edge with the highest score is chosen, given it exceeds some threshold min. The attached node is then added to the track candidate.

2. If any additional outgoing edges have scores greater than some threshold add (where $add > min$), a new track candidate is formed using that edge.

3. The walkthrough continues until no more nodes can be added to the track candidate(s).

If a connected component has multiple track candidates from the walkthrough algorithm, the ambiguity is solved by simply taking the longest track candidate. All nodes associated with a track candidate are removed from the graph; the walkthrough begins with the next source node.

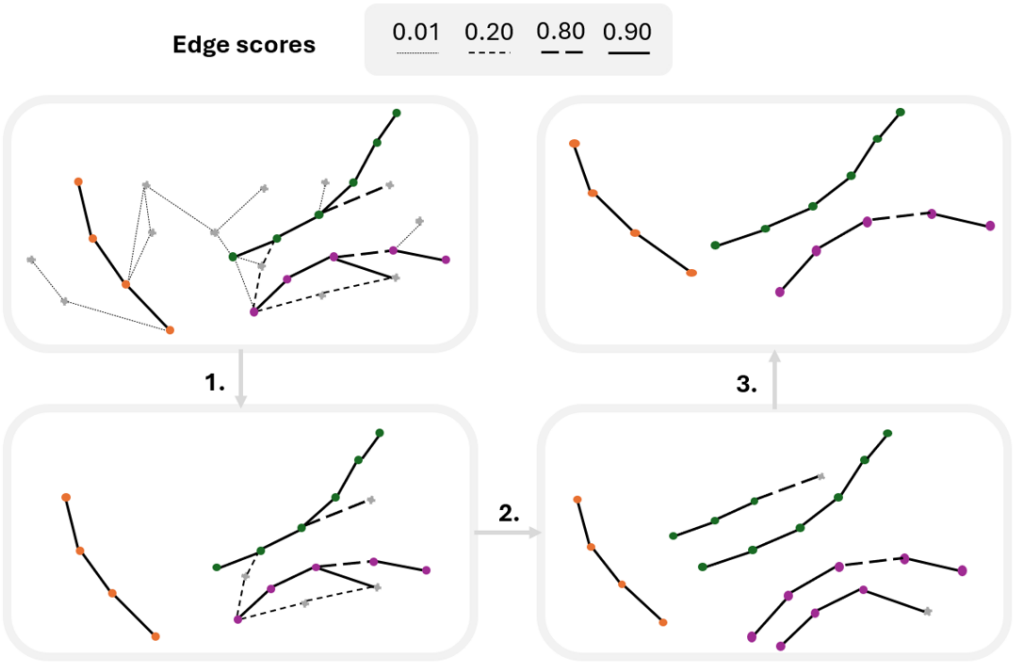


Figure 4.3: A schematic showing an example of the CCandWalk algorithm. Similarly coloured nodes indicate belonging to the same particle; grey spiked nodes do not belong to any particle track. For simplicity, four edge scores have been chosen.

Chapter 5

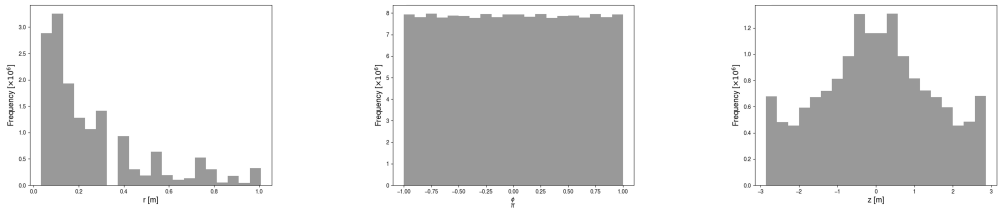
Baseline Results

The results presented hereafter are based on simulated $pp \rightarrow t\bar{t}$ events produced at centre of mass energy $\sqrt{s} = 14TeV$ with a pile-up of $\langle\mu\rangle = 200$. The Geant4-based ITk simulation with ITk layout version 23-00-03 (as shown in Figure 2.4) [27] was used. Evaluations have been performed on Nvidia A6000 GPUs directly running PyTorch in inference mode.

5.1 Simulation Data

To achieve an understanding of the data, histograms illustrating the features of 5000 sample events are shown. Figures 5.1 shows the distributions of the hit positions in cylindrical coordinates. As per common practise in high energy physics, the z-axis is defined along the beam axis. The ϕ distribution follows a uniform distribution as expected. The detector geometry becomes evident in the r and z distributions; the lack of hits in bin $0.325 < r[m] < 0.374$ reflects the gap between the strip and pixel sub-detectors (see Figure 2.4) where there are no active elements; likewise, the sharp peak in hits roughly every 200 mm after demonstrate the location of the strip barrel elements.

Figures 5.2 shows the distributions of the features of the simulated particles. Here, one can identify the criteria defining the target particles, laid out in Section 4. Figure 5.2b illustrates the sample is dominated by particles with $p_T < 1$ GeV, which comprise roughly 87.3% of the sample. Figure **??** shows the number of hits per track; one can see a bump in the number of hits a target track leaves around 9 to 13 hits, which corresponds to the minimum



(a) (b) (c)

Figure 5.1: Histograms of hit positions in cylindrical coordinates r, ϕ, and z respectively.

number of points in the barrel region and end cap in the ITk respectively. Figure 5.2e shows the particle constituents of the data sample. Electrons are excluded from the target criteria as a simplification.

5.2 Baseline

This section will now present the results of the *baseline* process: that is, the method of track finding following exactly Section 4.1. All metrics referred henceforth in this section will refer to the target: edge-wise efficiency, for example, will be the ratio of true *target* edges present in the graph to all true *target* edges.

5.2.1 Graph Construction

Metric Learning

A metric learning net was trained on 1000 events for 170 epochs. The net's architecture was as follows: five fully connected layers (a 3-dimensional input layer, 12-dimensional output layer, and three high-dimensional hidden layers), with a layer normalisation applied after each layer followed by a Tanh activation. The Adam optimiser is used, with the learning rate gradually increasing over a period of 5 epochs to reach a maximum of 0.01, and then decaying. Three node features were inputted: the spacepoint positions in cylindrical coordinates, (r, ϕ, z).

The number of hidden dimensions of the latent space (i.e. the dimensionality of each hidden layer) is a key hyperparameter of the net: the greater the dimensions, the greater the ability of the net to extract abstract features

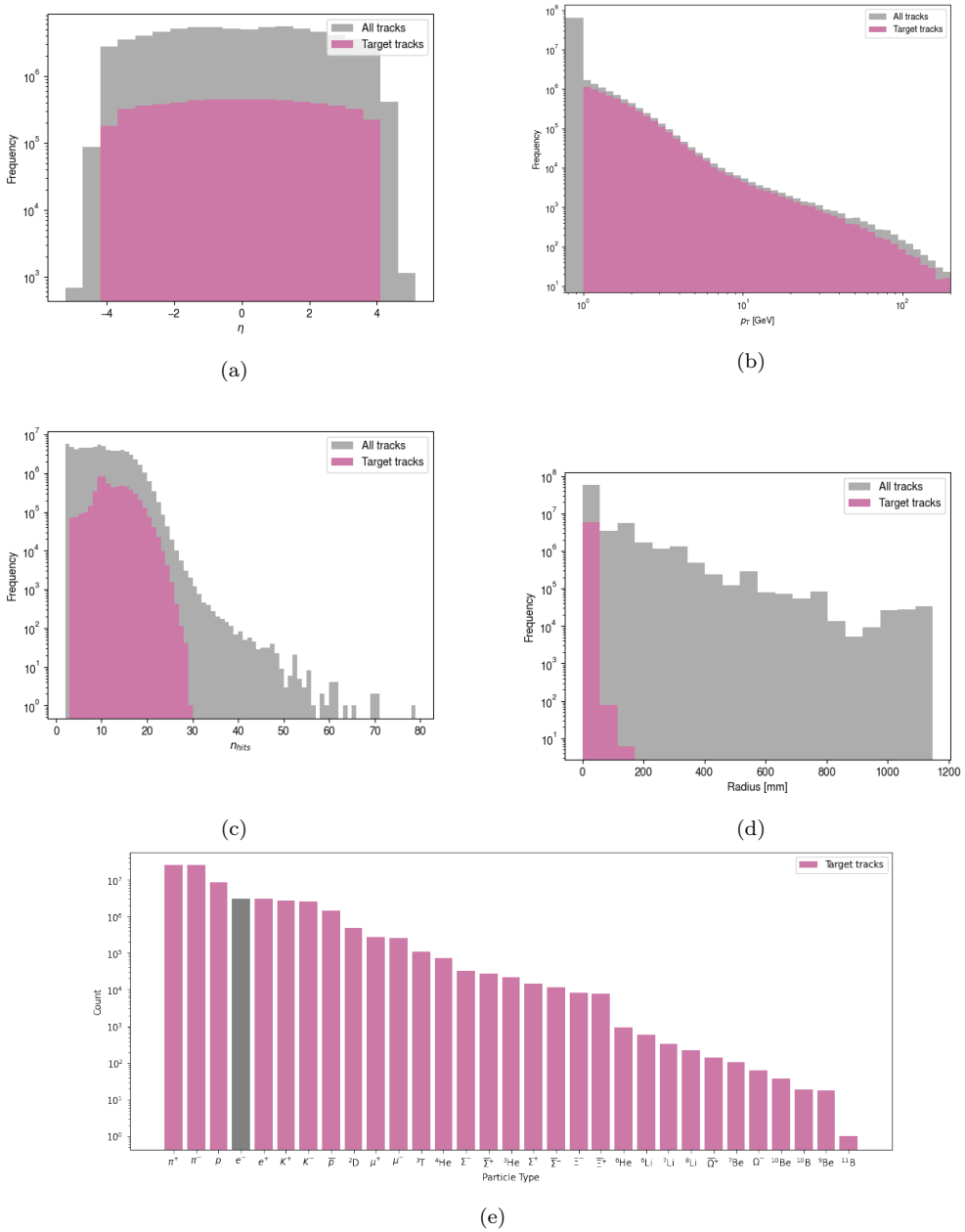
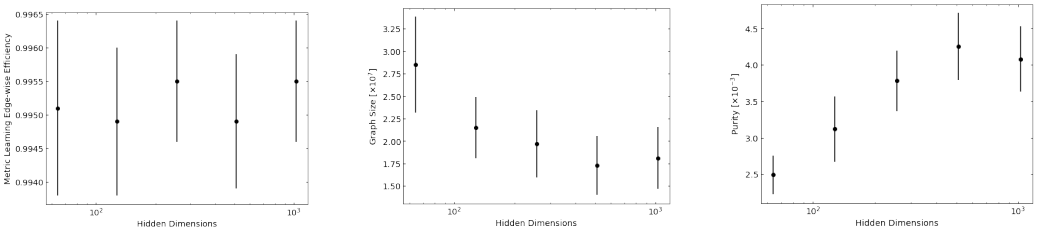


Figure 5.2: Histograms of η and transverse momentum (top row), number of hits and radius of the creation vertex (middle row), and particle type (bottom row) of particles in data sample.



(a) Edge-wise efficiency vs. hidden dimensions

(b) Graph size vs. hidden dimensions

(c) Edge-wise purity vs. hidden dimensions

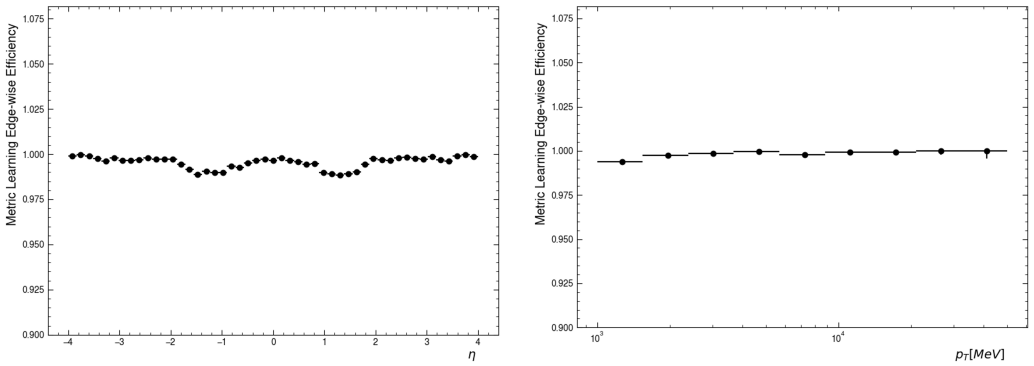
Figure 5.3: Efficiency, graph size, and purity of 300 graphs as a function of hidden dimensions of the metric learning nets that built them.

from the data - but the greater the computational requirements. Too many dimensions and the network can become over-parametrised and performance will stabilise, or even deteriorate: the "curse of dimensionality" can make distances between points meaningless. As such, five nets were trained with [1024, 512, 256, 128, 64] hidden dimensions, but otherwise identical architecture. The same 300 events were sent through each net, with inference radii chosen such that 99.5% efficient graphs were constructed. Figure 5.3 shows the edge-wise efficiency (5.3a), the number of edges per graph (hence called *graph size*, in 5.3b). and edge-wise purity (5.3c) of the graphs constructed by each net. Performance of each net can be measured by the number of superfluous edges built whilst building a constant proportion of target edges: it can thus be seen that performance is maintained for nets with 1024 and 512 hidden dimensions, but deteriorates for smaller nets. A net with 512 hidden dimensions was thus used.

The network was trained using the loss defined in Eq. 4.2; the weights were chosen as follows:

- $w_{ij} = 0$: nodes i and j are successive hits for a non-target particle (*true non-target*)
- $w_{ij} = 1$: nodes i and j are not successive hits for any particle (*fake*)
- $w_{ij} = 3$: nodes i and j are successive hits for a target particle (*true target*)

This emphasises the contribution of true target edges to the loss and masks any contribution from true non-target edges. The total loss was then the



(a) Graph construction efficiency vs. η

(b) Graph construction efficiency vs. p_T

Figure 5.4: Graph construction edge-wise efficiency as a function of η and p_T.

mean of all target contributions summed with the mean of all fake contributions, such that the vast quantities more of fake connections do not overpower the true target. A sweep of weights of true target edges was undertaken to find the optimal value of 3.

The training progress was monitored in real time using the Weights and Biases framework [28]. Training data was split into a train set, validation set, and test set following an 8/1/1 ratio; the train set is used to train the network, the validation set is used to monitor the loss to prevent overfitting on the train set, and the test set is used for final evaluation of the model. A metric calculating the edge-wise purity at 99.5% edge-wise efficiency of the validation set was monitored during training, and training was stopped when this metric, along with others such as the validation loss, plateaued at around epoch 170.

100 events were sent through the trained network with an inference radius of 0.12 on the test dataset to build graphs with a mean efficiency of 0.9955 ± 0.0009 and a mean purity of $(4.08 \pm 0.05) \times 10^{-3}$. The graphs contained $(1.81 \pm 0.35) \times 10^7$ edges. Figures 5.4 shows the graph construction efficiency as a function of η (Figure 5.4a) and p_T (Figure 5.4b). The deterioration in efficiency at $|\eta| = 1$ is because the strip barrel has lower space point longitudinal resolution than the pixel detector. This can be seen in Figure 2.4.

A short study conducted found that the net struggles to construct the final few percent of true target edges. Seven sets of 100 graphs were built

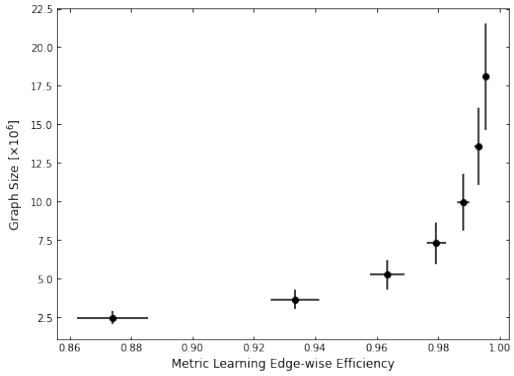


Figure 5.5: Graph size against metric learning edge-wise efficiency for seven sets of graphs built with the same net at varying inference radii.

with inference radii in the interval [0.06, 0.12]: Figure 5.5 shows the size of the graphs against their edge-wise efficiency. One sees a ballooning of graph size to gain the final few percent of edges.

Filter

A filter net was trained on 1000 events for 200 epochs, when monitored metrics plateaued and training was stopped. Thirteen node features were inputted $(r, \phi, z, r^1_{cl}, \phi^1_{cl}, z^1_{cl}, r^2_{cl}, \phi^2_{cl}, z^2_{cl}, \angle\eta^1_{cl}, \angle\eta^2_{cl}, \angle\phi^1_{cl}, \angle\phi^2_{cl})$ into three stacked graph convolutional modules. These are:

- $\mathbf{r}, \phi, \mathbf{z}$: cylindrical coordinates of space points;
- $\mathbf{r^{1,2}_{cl}}, \phi^{\mathbf{1,2}}_{\mathbf{cl}}, \mathbf{z^{1,2}_{cl}}$: for strip space points: cylindrical coordinates of the two strip clusters that form the space point; for pixel space points: a repeat of the space point coordinates;
- $\angle\phi^{\mathbf{1,2}}_{\mathbf{cl}}, \angle\eta^{\mathbf{1,2}}_{\mathbf{cl}}$: the ϕ and η angle respectively of the (for strip space points first and second) cluster relative to the normal of the module it is located on.

All modules applied a graph convolution using the GraphSAGE operator [29], normalised the output with layer normalisation, and then applied a ReLU activation. The three modules had 256, 512, and 1024 hidden layers respectively.

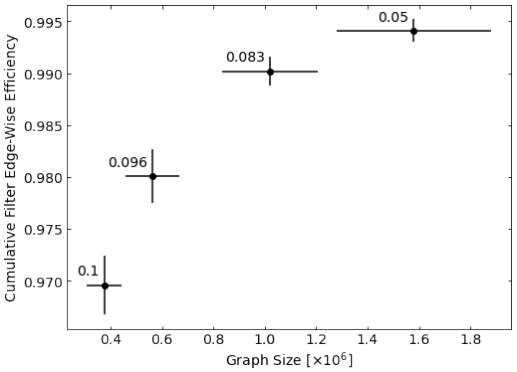


Figure 5.6: Cumulative filter efficiency of four sets of 100 graphs, as a function of graph size. The labels on each point is the score cut applied to prune the graphs.

The SAGEConv updates the latent node features of node i by aggregating latent features from all neighbouring nodes j,

$$x_i' \leftarrow W_1 x_i + W_2 \cdot AGG\{x_j | \forall j \in \mathcal{N}(i)\} \tag{5.1}$$

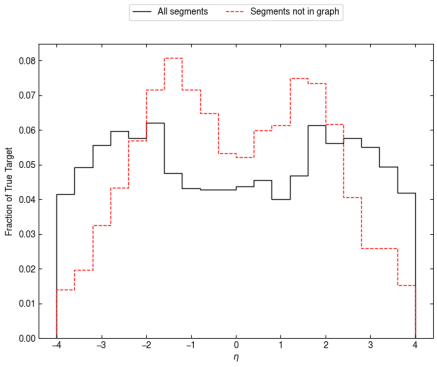
where W are the learnable weight matrices and AGG here is the mean.

The network was trained using the binary cross-entropy loss in Eq. 4.3; the weights were chosen as follows:

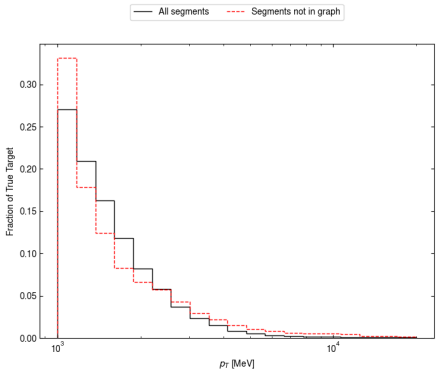
- $w_n = 0$: for *true non-target* edges
- $w_n = 1$: for (*fake*) edges
- $w_n = 10$: for (*true target*) edges

The 100 graphs built to 99.55% efficiency by the metric learning net were sent through this filtering net; edges with scores below four different thresholds were pruned to obtain four sets of graphs with cumulative edge-wise efficiency between [0.97, 1], as shown in Figure 5.12. As for the metric learning net, it is evident that the net struggles with the final few percent in efficiency: the 99.4% efficient graph is more than four times the size of the 97% efficient graph despite containing only $\sim$ 2000 more target edges.

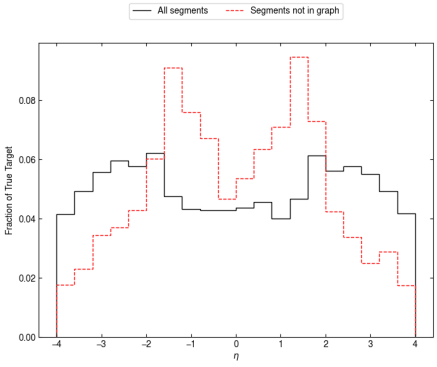
Figure 5.7 shows the η and p_T distributions of all true target edges and those not present in the post-cut graph for the 0.97 and 0.99 post-filter efficient graphs. True target edges not in the graph disproportionately occur at $|\eta| = 1$, a reflection of the situation in Figure 5.4a, and explained similarly



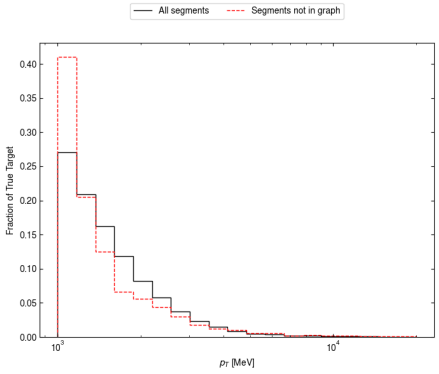
(a) Distribution of η of true target edges, and true target edges not in 97% graph.



(b) Distribution of p_T of true target edges, and true target edges not in 97% graph.



(c) Distribution of η of true target edges, and true target edges not in 99% graph.

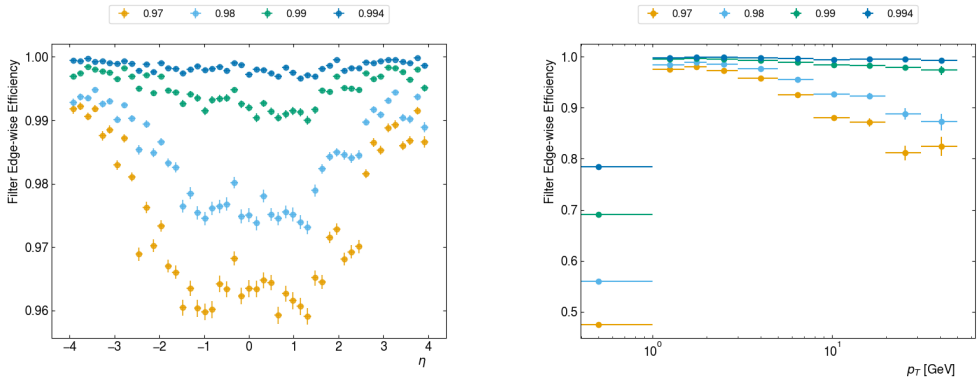


(d) Distribution of p_T of true target edges, and true target edges not in 99% graph.

Figure 5.7: True targets segments in event and missing from 97% and 99% graphs

by the geometry of the ITk; the filter also has difficulties with edges at low p_T (a problem sustained at higher overall efficiencies) and at high p_T (a problem overcome at higher overall efficiencies). Difficulties at high-p_T are due to a lack of statistics in the data sample: low-p_T is presumably reflective of the inherent challenges in connecting tracks with greater curvature.

Figures 5.8 show the edge-wise efficiency of the filter stage as a function of η (5.8a) and p_T (5.8b). It is clear the filter struggles to score correctly target edges occurring mainly in the barrel region of the ITk and at high transverse momentum. The degradation in efficiency at high transverse momentum is explained by the training dataset: high p_T tracks comprise only a small fraction of the input target particles (see Figure 5.2b). The drops in efficiency at $|\eta| = 1$ is a reflection of the situation in Figure 5.4a, and is explained



(a) Filter efficiency vs. η

(b) Filter efficiency vs. p_T

Figure 5.8: Filter edge-wise efficiency as a function of η and p_T for the four sets of graphs shown in Figure . The legend refers to their cumulative efficiencies.

similarly by the geometry of the ITk.

5.2.2 GNN

The GNN edge-scoring net was trained on 2000 events for 200 epochs, when monitored metrics plateaued and training was stopped. The training data were the 99% efficient graphs post-filter. Twelve node features were inputted: previous works have used a homogeneous GNN with three input node features, the space point cylindrical coordinates (r, ϕ, z) as node features, but these have performed poorly in the barrel strip region [18], where two clusters form each space point. An attempt has been made to mitigate this by the inclusion of the cluster positions, such that 12 node features are now used are:

- $\mathbf{r}, \phi, \mathbf{z}$: cylindrical coordinates of space points;
- η: the pseudorapidity of the space point
- $\mathbf{r_{cl}^{1,2}}, \phi_{\mathbf{cl}}^{\mathbf{1,2}}, \mathbf{z_{cl}^{1,2}}$: for strip space points: cylindrical coordinates of the two strip clusters that form the space point; for pixel space points: the single pixel cluster coordinates repeated;
- $\eta_{\mathbf{cl}}^{\mathbf{1,2}}$: the pseudorapidity of the (for strip, first and second) cluster

Edge features inputted are the distance between space points in cylindrical coordinates, $(\Delta r, \Delta\phi, \Delta z)$.

The IGNN follows the encode-process-decode architecture described in Section 4.1.2, with the following architecture:

- The edge and node encoders both use a fully-connected three layer network (with a 3 and 12 - dimensional input layer respectively, a 64-dimensional hidden layer, and a 64-dimensional output layer) with a batch normalisation applied after each layer followed by a ReLU activation

- Eight message passing steps corresponding to eight layers of the interaction network are used. Each step uses two fully-connected three-layer networks, to update the node and edge latent features respectively: the node block, with a 256-dimensional input layer, and 64-dimensional hidden and output layers, with batch normalisation and ReLU activation applied after each layer; and the edge block, with a 384-dimensional input layer, and 64-dimensional hidden and output layers, with batch normalisation and ReLU activation applied after each layer

- An edge decoder takes the final 64-dimensional latent features through a three layer network (with one hidden layer of 64 dimensions) and outputs an edge score.

The network was trained using the loss defined in Eq. 4.3; the weights were chosen as follows:

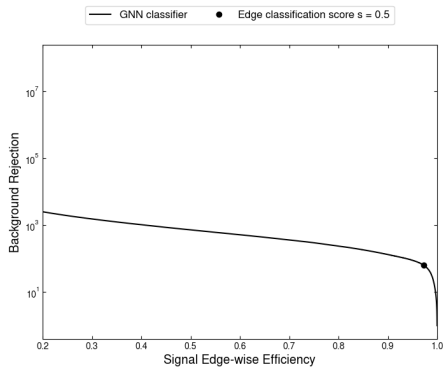
- $w_{ij} = 0$: nodes i and j are successive hits for a non-target particle (*true non-target*)

- $w_{ij} = 0.1$: nodes i and j are not successive hits for any particle (*fake*)

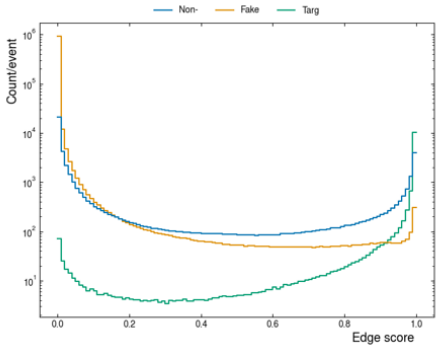
- $w_{ij} = 1$: nodes i and j are successive hits for a target particle (*true target*)

as suggested by [18].

Figure 5.9a shows the GNN edge classification performance after training, evaluated on 0.99 efficient graphs. Background rejection rate is defined as $1/\epsilon_{bkg}$ where ϵ_{bkg} is the fraction of fake edges passing the threshold. Figure 5.9b shows the score distribution on the same 100 99% efficient graphs: most



(a) GNN classification performance, based on GNN edge-wise efficiency and GNN edge-wise fake rate.



(b) Score distribution for 100 0.99 efficient graphs sent through the GNN. Targ- are true target edges; non- are true non-target edges; fake are fake edges.

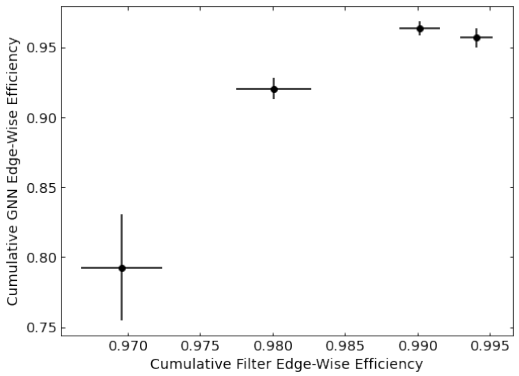
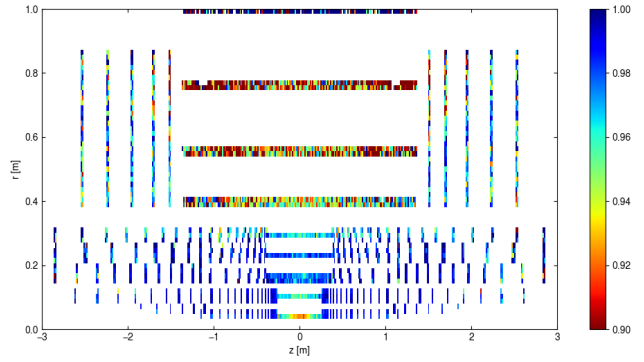


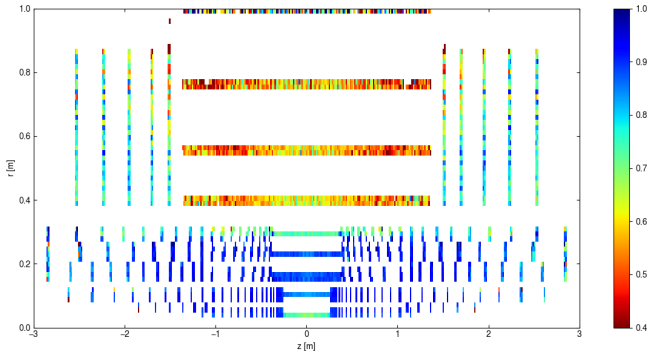
Figure 5.10: Output edge-wise efficiency post-GNN classification with a score cut of $s = 0.5$ against input edge-wise efficiency.

importantly, we see a ramp up of number of target edges as score increases. The flat distribution of true non-target edges is as expected, given they are masked from the GNN loss; the fake edges mostly have low scores as desired. Peaks at either extreme of the score distribution for all categories speaks to some edges of all categories having properties that indicate either true target or fake.

Once trained, the four sets of graphs at [0.97, 0.98, 0.99, 0.994] were sent through the IGNN net. Figure 5.10 shows the cumulative edge-wise efficiency against input edge-wise efficiency for a score cut of $s = 0.5$. The deterioration in performance for the 0.994 input efficient graph is likely because the GNN was trained on the 0.99 input efficient graphs and so performs better on the relatively more sparse graphs.



(a) GNN efficiency in (z, r) plane



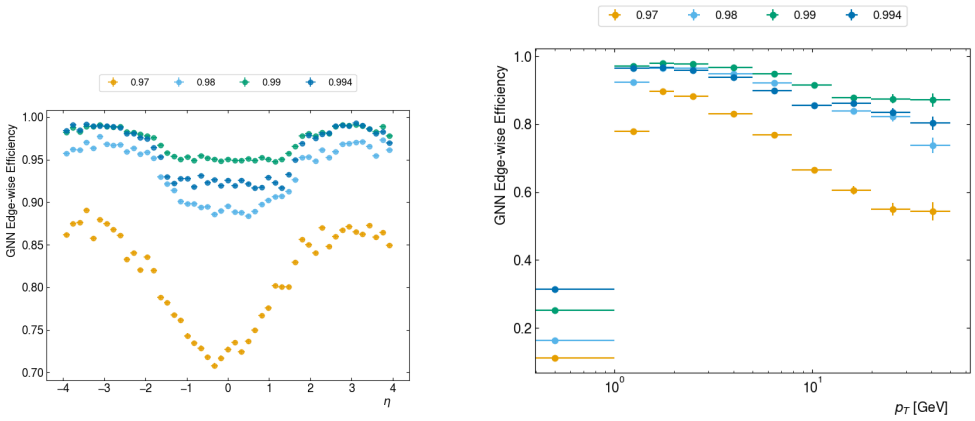
(b) GNN masked purity in (z, r) plane

Figure 5.11: Performance of the GNN edge classifier in the (z, r) plane on the 0.99 efficient graphs with a score cut of $s = 0.5$.

Figure 5.11 shows the performance of the GNN edge classifier in the (z, r) plane at a score cut of $s = 0.5$. Figure 5.11a shows the edge-wise efficiency across the detector; Figure 5.11b shows the masked purity across the detector. The masked purity is defined as the total target edges in graph as a fraction of all edges in graph excluding true non-target edges. This is a useful metric as building true non-target edges is not necessarily a negative. The efficiency is high across the detector, but both metrics perform continue to perform worse in the strip barrel; this is a repeated problem.

5.2.3 Track Reconstruction

Track candidates were built from the four sets of graphs using both the ConnectedComponents (CC) algorithm and the CCandWalk algorithm.



(a) GNN efficiency against η

(b) GNN efficiency against p_T

Figure 5.12: GNN edge-wise efficiency with a score cut of $s = 0.5$ as a function of η and p_T for the four sets of graphs shown in Figure . The legend refers to their cumulative efficiencies post filter.

A track is specified as matched and the associated particle as reconstructed if the track contains at least 3 hits and more than half these hits are shared with the particle, $p_{shared} > 0.5$, according to criteria laid out in [10].

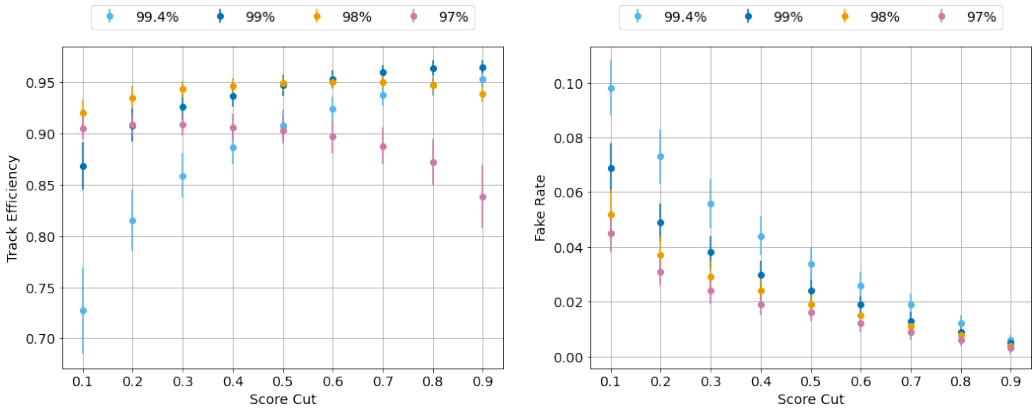
A particle is *duplicated* if it has multiple matched tracks; a track is *fake* if it has more than three hits but has not been matched to a particle.

Connected Components

The CC algorithm was run with score cuts in 0.1 intervals between [0, 1] on the four sets of graphs from Figure 5.12. Figure 5.13 shows tracking efficiency (5.13a), fake rate (5.13b), and duplication rate (5.13c) against score cut of the resulting track candidates built.

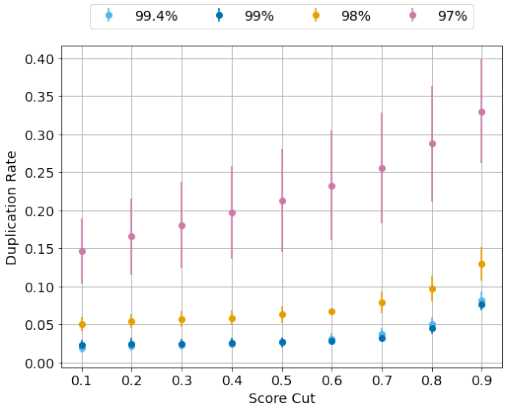
Figure 5.13a shows a maximum tracking efficiency of 0.964 ± 0.007 is achievable using a score cut of 0.8. It also shows for the larger graphs a higher score cut achieves a higher tracking efficiency, which is unintuitive. This is due to the matching criteria that requires the track to share at least half it's hits with the associated particle.

To illustrate this, track candidates built from one event with a CC score cut of 0.5 have been analysed: this produced a track efficiency of 0.947. Figure 5.14 shows the hit distributions of the matched tracks from this event. All tracks have at least 3 hits and share more than half of them with an associated particle.



(a) Track efficiency against score cut

(b) Fake rate against score cut



(c) Duplication rate against score cut

Figure 5.13: Track efficiency, fake rate, and duplication rate of the track candidates formed at various ConnectedComponents score cuts of the four sets of graphs.

Track candidates were built from the same event but now with a CC score cut of 0.1, giving a track efficiency of 0.864. The hit distributions of the particles reconstructed with a CC score cut of 0.5 were taken, and ratios of shared hits and track hits for those particles in 0.1 and 0.5 calculated. Figure 5.15a shows the ratio of shared hit distributions: one can see all particles that were reconstructed for s = 0.5 share at least as many hits with tracks for s = 0.1. Figure 5.15b shows the ratio of track hit distributions: one can see tracks with s = 0.1 have far more hits than tracks with s = 0.5.

In essence, a lower score cut "pollutes" a matched track with superfluous edges so that it no longer shares at least half its hits with the particle.

The increase in duplication rate with greater score cuts seen in Figure 5.13c is a because harsher cuts "break" track candidates. The nature of the CC algorithm means two track candidates sharing hits is impossible - a hit can only belong to one connected component. Multiple tracks associated to one particle are because some edges linking that particle are not making the cut. Recalling the score distribution in Figure 5.9b, one sees increasing numbers of true target edges have scores in the region [0.6, 1.0]: CC cuts greater than 0.6 in particular are likely to cut true target edges and thus potentially break tracks.

The increase in fake rate at lower score cuts seen in Figure 5.13b is because at lower cuts more tracks with at least 3 hits are formed - and many of these are fake.

The variability of these metrics with both score cut and post-filter cumulative edge-wise efficiency is good reason to use the alternative algorithm, CCandWalk.

CCandWalk

The nature of the CCandWalk algorithm makes it inherently more stable, as the walkthrough can seek out the most likely true target edge from a wider range of scores, and will not keep any remaining branching edges that pollute the track. The downside of the algorithm is it is understandably much slower, although progress is being made to speed it up.

Quick scans of varying thresholds of $0.1 < min < 0.4$ and $0.5 > add > 0.8$ found little differentiation in end results. Three thresholds were chosen under suggestions from [18]: an initial cut of any edges with score below s = 0.01 was made, and then the walkthrough commenced with thresholds $min = 0.2$

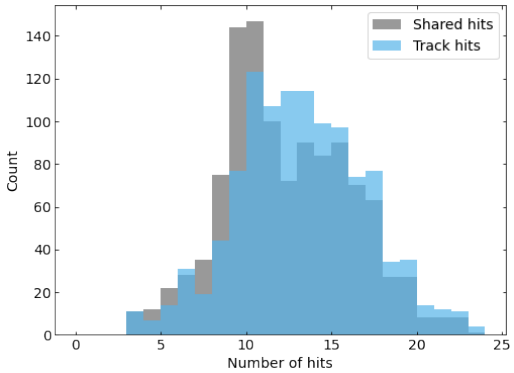
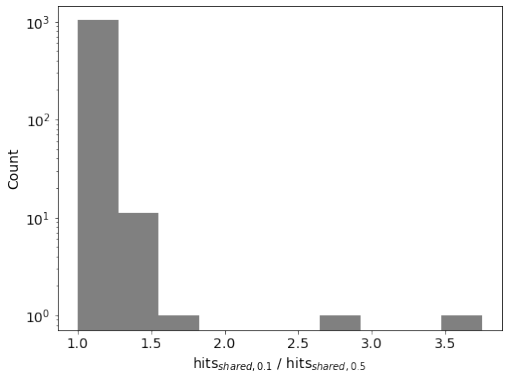
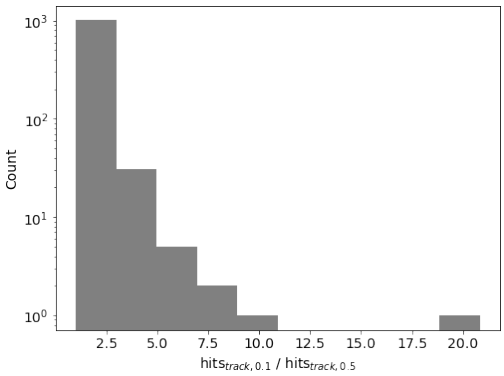


Figure 5.14: Distribution of number of hits of matched tracks (in blue) and number of hits shared by matched tracks and reconstructed particles (in grey) for track candidates built from one graph with the CC algorithm using a cut of s = 0.5.



(a) The ratio of shared hits of particles reconstructed at a score cut of 0.5

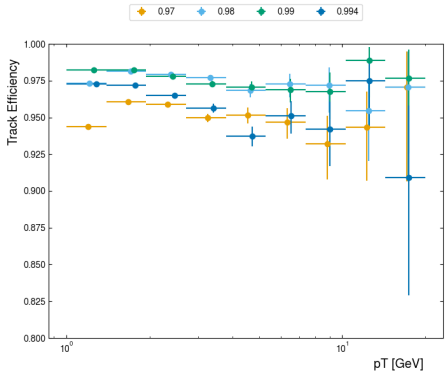


(b) The ratio of track hits of particles reconstructed at a score cut of 0.5

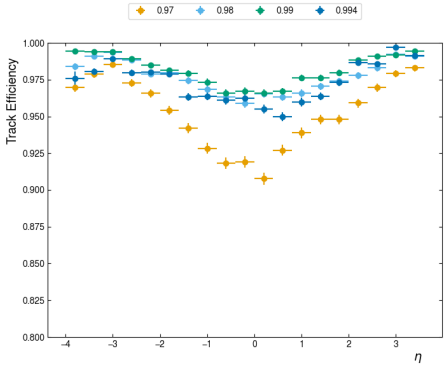
Figure 5.15

Post-Filter Efficiency	Track Efficiency	Fake Rate	Duplication Rate	Graph Size [$\times 10^5$]
0.97	0.960 ± 0.008	0.053 ± 0.006	0.163 ± 0.038	3.75 ± 0.07
0.98	0.977 ± 0.004	0.059 ± 0.008	0.077 ± 0.009	5.63 ± 0.10
0.99	0.983 ± 0.004	0.058 ± 0.005	0.072 ± 0.009	10.2 ± 0.18
0.994	0976 ± 0.006	0.063 ± 0.007	0.126 ± 0.010	15.8 ± 0.30

Table 5.1: Results of track candidates built using CCandWalk from the four sets of graphs, with their post-filter graph size.



(a) Track efficiency vs. p_T. x-values (but not errors) shifted slightly for ease-of-view.



(b) Track efficiency vs η.

Figure 5.16: Track efficiency of the four sets of graphs using the CCandWalk algorithm.

and $add = 0.7$.

Track candidates were built using these thresholds from the four sets of graphs; the results are shown in Table 5.1.

The post-filter 99% efficient graph produced the best track efficiency of 98.3%, with a fake rate and duplication rate of $\mathcal{O}(10^{-2})$. Fitting the tracks would be expected to reduce the amount of fake tracks. Figures 5.16 shows the track efficiency of the four sets of graphs against p_T (5.16a) and η (5.16b). Despite the additional node features added to the GNN, there maintains a clear reduction in track efficiency within the central region ($|\eta| < 2$).

Figure 5.17 shows the track efficiency against graph size for the four sets of graphs.

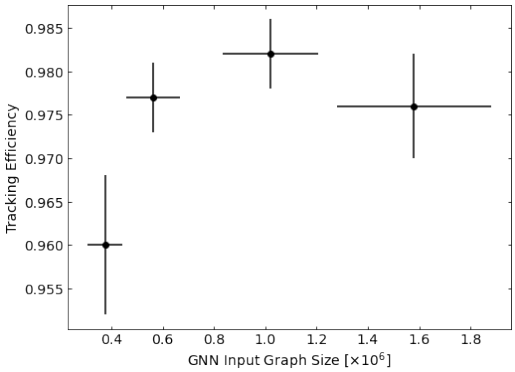


Figure 5.17: Track efficiency vs. post-filter graph size, produced with the CCandWalk algorithm.

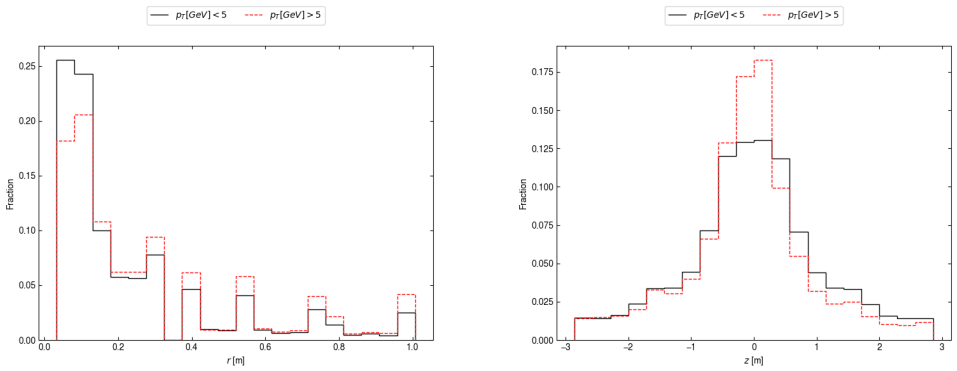
Chapter 6

Iterative Approach

The GNN-based track reconstruction algorithm following the pipeline in Figure 4.1 on ATLAS ITk simulated data for the HL-LHC shows promising results. However, the graph input into the edge scoring GNN, the most computationally intensive area of the pipeline, is very large. It contains 1.02×10^6 edges, which is roughly 95 edges for every true edge, and roughly 940 edges for every target true edge. There seems, then, to be large room for improvement in reducing the number of superfluous edges and making purer graphs. Purer, smaller, graphs would pass more quickly through the pipeline (key for any process in the trigger system); they would require less computational resources (key in the HL-era); and they may even allow less room for error in the edge scoring and track building stage of the pipeline, producing potentially better results.

The large impure graph is partially a result of the rapid increase in graph size required to gain the last few percent of edge-wise efficiency, seen in both the metric learning and filter stage, for example in Figure 5.5. Recall Figure 5.7: here, we saw that the last few percent of edges the 99% efficient graph was still missing occurred disproportionately in the $|\eta| = 1$ region (5.7c) and for edges with low transverse momentum, $p_T \approx 1$ GeV (5.7d).

So: how can one make these challenging areas easier on the net? Perhaps it is possible to train nets that specialise in subsets of these distributions and thus outperform the previous nets. One clear area of potential is to train nets for certain transverse momentum ranges. The radius of curvature of a charged particle travelling through a magnetic field is directly proportional to it's transverse momentum; as such tracks with high transverse momentum should have clearly distinguishable features from tracks with low transverse



(a) Position of hits from particles with $p_T < 5$ GeV, from 100 events.

(b) Position of hits from particles with $p_T > 5$ GeV, from 100 events.

Figure 6.1: Distribution of r, z coordinates as a fraction of total hits.

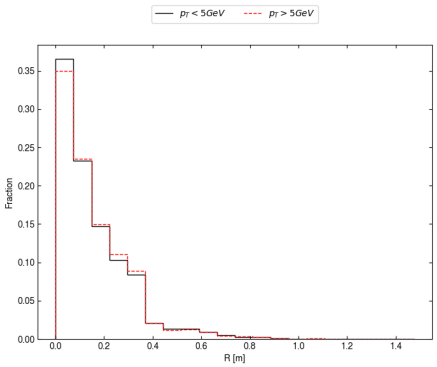
momentum.

Figure 6.1 shows the fraction of total hit positions as a function of cylindrical coordinates r and z for particles with $p_T > 5$ and $p_T < 5$ GeV. It can be seen that particles with greater momentum propagate further into the detector (r-axis) and less far along the beam axis (z-axis). Figure 6.2 shows the separation of consecutive hits in particle tracks along the z-axis (along the beam line, 6.2c), the r-axis (transverse to the beam line, 6.2b) and $R = \sqrt{r^2 + z^2}$, (6.2a). As expected, particles with greater momentum have smaller mean separation along the beam line despite slightly greater total separation.

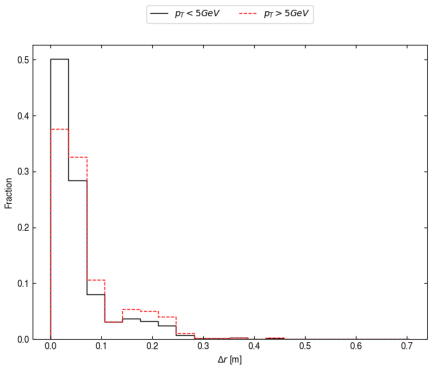
These are some of the features of nodes that could be used by the network to distinguish between high and low momentum tracks. So: the idea is plausible. The question to ask, then, is if a network can be trained to distinguish and build only high (or low) p_T tracks – and if they can, if the specialised nets produce purer and smaller graphs – and thus uses less computational resources.

6.1 High Momentum Specialised Graphs

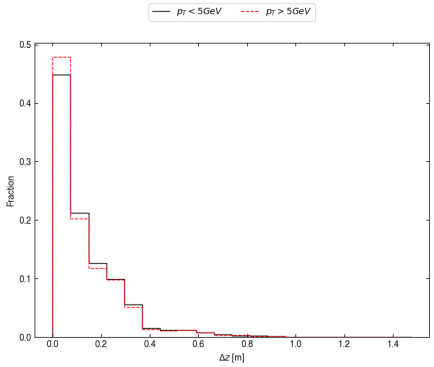
An attempt is first made to build graphs aimed at high momentum particles. Four thresholds for particle momentums were chosen: tracks with momentum greater than [1.5, 2, 3, and 5] GeV, or [14.3, 6.01, 2.90, 0.91, 0.22]% of target particles respectively.



(a) Separation between hits in rz-plane.



(b) Separation between hits in r-axis.



(c) Separation between hits in z-axis.

Figure 6.2: Fractional distribution of the separation between consecutive hits of particles, of particles with $p_T > 5$ GeV and $p_T < 5$ GeV, from 100 events.

6.1.1 Metric Learning

Four metric learning nets were trained on 1000 events for 170 epochs. Target particles were defined (as before) as all non-electron particles produced in the primary vertex of top quark pair production, produced at transverse radius $r < 260$ mm and $|\eta| < 4$, that leave at least 3 hits, but now with transverse momentum $p_T > [1.5, 2, 3, 5]$ GeV. Training was stopped when monitored metrics plateaued, around epoch 170.

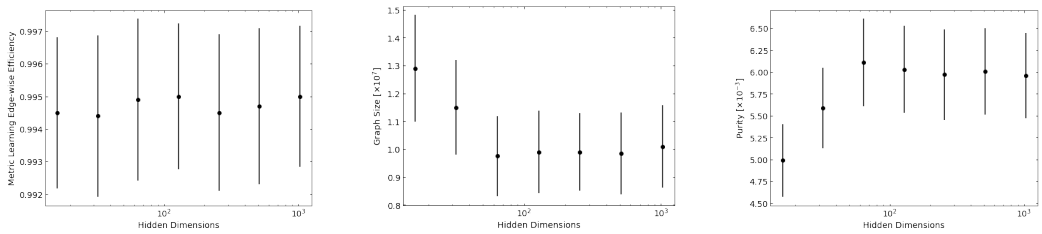
From here, graphs referred to as $p_T > X = [1.5, 2, 3, 5]$ GeV will indicate graphs aimed at building particles with transverse momentum greater than 1.5, 2, 3, or 5 GeV respectively: it will not mean a cut on all particles with momentum lower than X. Similarly, unless otherwise indicated, the edge-wise evaluation metrics efficiency and purity will reference target momentum, i.e. target edges will be defined as those associated with tracks with momentum greater than [1.5, 2, 3, 5] GeV.

Architecture

The architecture of the four nets trained was predominantly the same as for the net in Section 5.2.1: however, a case study was undertaken for reducing the dimensions of the latent space (the hidden dimensions).

Theoretically, a multi-layer perceptron with a greater number of hidden dimensions has a greater ability learn complex patterns by extracting increasingly abstract features from the dataset. The logic follows that for a less complex dataset - for example, one with straighter target tracks - the net has to capture fewer features, and may thus have similar performance at fewer hidden dimensions.

The set of target tracks with $p_T > 3$ GeV were taken as a case study, with seven metric learning nets trained. All followed the same architecture as in Section 5.2.1 but now with [1024, 512, 256, 128, 64, 32, 16] dimensions of the latent space respectively. The same 300 events were sent through each net, with inference radii chosen such that 99.5% efficient graphs were constructed. Figure 6.3 shows the edge-wise efficiency (6.3a), graph size (6.3b), and edge-wise purity (6.3c) of the graphs against the hidden dimensions of the nets that built them. Performance of each net can be measured by the number of superfluous edges built whilst building a constant proportion of target edges: it can thus be seen that performance is maintained for nets with 1024 to 64 hidden dimensions, but deteriorates for smaller nets.



(a) Edge-wise efficiency vs. hidden dimensions

(b) Graph size vs. hidden dimensions

(c) Edge-wise purity vs. hidden dimensions

Figure 6.3: Efficiency, graph size, and purity of 300 graphs as a function of hidden dimensions of the metric learning nets that built them.

This is a remarkable result: recall the investigation into hidden dimensions for the metric learning net with target tracks $p_T > 1$ GeV in Section 5.2.1. There, a deterioration in performance was seen for nets with fewer than 512 hidden dimensions. Every time the number of hidden dimensions is doubled the total number of network parameters are quadrupled, so the smallest well-performing $p_T > 3$ network, with 64 hidden dimensions, has 64 times less network parameters than the $p_T > 1$ network. This means the computational and memory requirements are vastly reduced for the graph construction stage. It also has implications for further along the pipeline: the IGNN apparatus includes multiple MLPs, the edge and node encoders and decoders. If the latent space these MLPs encode into (or decode from) could similarly have reduced dimensionality without loss of performance, then computational requirements would plummet. Indeed, in every aspect of network architecture, the question can be asked: is the depth that is required for the net to capture the complexity of low-momentum tracks needed if one aims only for high-momentum tracks?

This investigation was not taken further within this thesis, and the nets aimed at $p_T > [1.5, 2, 5]$ GeV tracks used 1024 hidden dimensions. However, it stands to reason similar reductions in hidden dimensions should be plausible, and this is something to be confirmed in the future. The net with 64 hidden dimensions was used for $p_T > 3$.

For each metric learning net trained for particles with $p_T > [1.5, 2, 3, 5]$ GeV, 100 events were sent through and graphs built; inference radii were chosen to build graphs with a mean edge-wise efficiency of 99.5%. The following Table 6.1 shows the edge-wise efficiency, graph size, and purity of these four sets of graphs, and the 0.99% 1 GeV set of graphs which from now on will

X [GeV]	Edge-wise Efficiency	Graph Size [$\times 10^7$]	Purity [$\times 10^{-3}$]
1	0.9951 ± 0.0009	1.81 ± 0.35	4.08 ± 0.45
1.5	0.9950 ± 0.0016	1.24 ± 0.23	5.26 ± 0.57
2	0.9951 ± 0.0017	1.13 ± 0.21	5.55 ± 0.58
3	0.9949 ± 0.0031	0.965 ± 0.18	6.11 ± 0.63
5	0.9951 ± 0.0071	0.944 ± 0.18	5.97 ± 0.62

Table 6.1: Results from 100 events passed through metric learning nets.

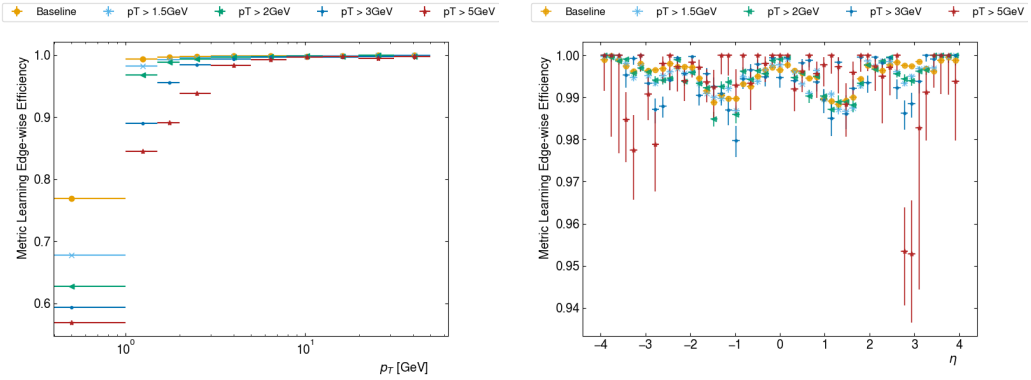
be used as the *baseline* in comparison.

Figure 6.4 shows the edge-wise efficiency of four sets of graphs against transverse momentum (6.4a) and pseudorapidity (6.4b). The edge-wise efficiency in Figure 6.4a calculates the edge-wise efficiency by defining true edges as those associated with particles with $nhits \geq 3$, $|\eta| < 4$, $r < 260$ mm (as usual), but for all transverse momenta, to enable the study of behaviour at low-p_T.

Every non-target edge built, even if true, increases the graph size and obscures the true target tracks, and thus preferably should be avoided; low efficiencies outside of the momentum limits for each graph indicate fewer non-target edges. The perfect scenario for each graph would be 100% edge-wise efficiency for $p_T > X$ and 0% edge-wise efficiency for $p_T < X$: this is, of course, a lofty goal. However, the "step off" in efficiency seen in 6.4a is a clear success. For each graph, an edge-wise efficiency of 99.5% has been achieved for $p_T > X$ GeV. Below that threshold, the edge-wise efficiency begins to decrease, such that the greater the threshold, the lower the efficiency for $p_T < 1$ GeV.

The variation of edge-wise efficiency with η in Figure 6.4b largely follows that of the baseline, with ever greater deviations as the definition of target particle tightens. For graphs $p_T > 5$ GeV, it is largely ignored. This is explained by the lack of hits matching this momentum criteria: Figure 6.5 shows the distribution of particle hits as a function of $\eta \in [-4, 4]$, with the same binning as Figure 6.4b. Each bin has a mean of less than 10 hits; the fluctuations seen in 6.4b are statistical fluctuations.

The reduction in efficiency at low transverse momentum shown in Figure 6.4 should indicate smaller graph sizes for graphs with greater target momentum: Figure 6.6, which shows graph size against number of target segments (segments meaning all edges whether or not in graph), agrees with



(a) Metric learning efficiency vs. p_T

(b) Metric learning efficiency vs. η.

Figure 6.4: Metric learning edge-wise efficiency as a function of η and p_T for the four sets of graphs, $p_T > [1.5, 2, 3, 5]$

.

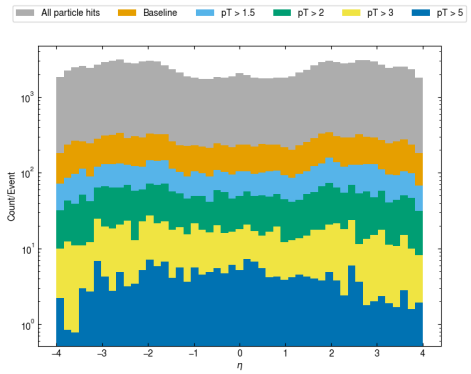


Figure 6.5: Distribution of particle hits against $\eta \in [-4, 4]$.

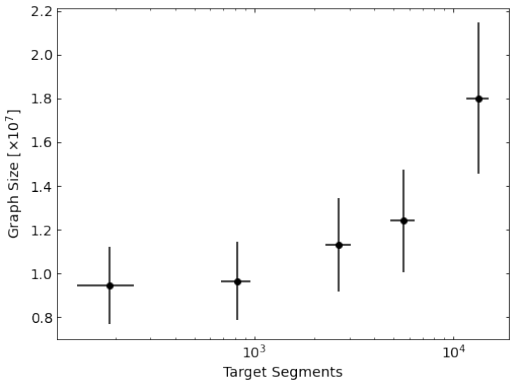


Figure 6.6: Metric learning graph size plotted against the number of segments meeting the changing target definition.

those expectations. As the boundary for target p_T increases, the number of segments that meet that target decreases, and with it the graph size. The graph size decreases with tightening target criteria at a reducing rate, reaching a plateau with the $p_T > 3$ graph roughly the same size as the $p_T > 5$ graph. This is likely due a lack of statistics for the $p_T > 5$ graphs.

It has been clearly illustrated that it is possible to train metric learning nets to recognise a threshold in transverse momentum and construct, with precedence, edges that exceed that threshold. This is the vital step in constructing a momentum-specialised pipeline, as all other stages act on the graph built here, and are reliant on the edges that do (or do not) exist here.

6.2 Iterative Approach

Section 6.1 has illustrated that it is well possible to train specialised nets to construct high p_T edges with precedence. Thus, an iterative approach to track finding is devised. Is it possible to build track candidates from the small, high-momentum graphs, remove the associated data (namely the hits) from each event and then build another set of graphs - this time aimed at the remaining target particles, with $p_T > 1$ GeV? If the second set of graphs are also small, and can produce enough track candidates to reconstruct the vast majority of target particles, this approach may be computationally less demanding for similar performance. A sketch of the process is shown in Figure 6.7: we separate it out into *Stage One*, the first pass through the pipeline to build high momentum candidates, and *Stage Two*, the second

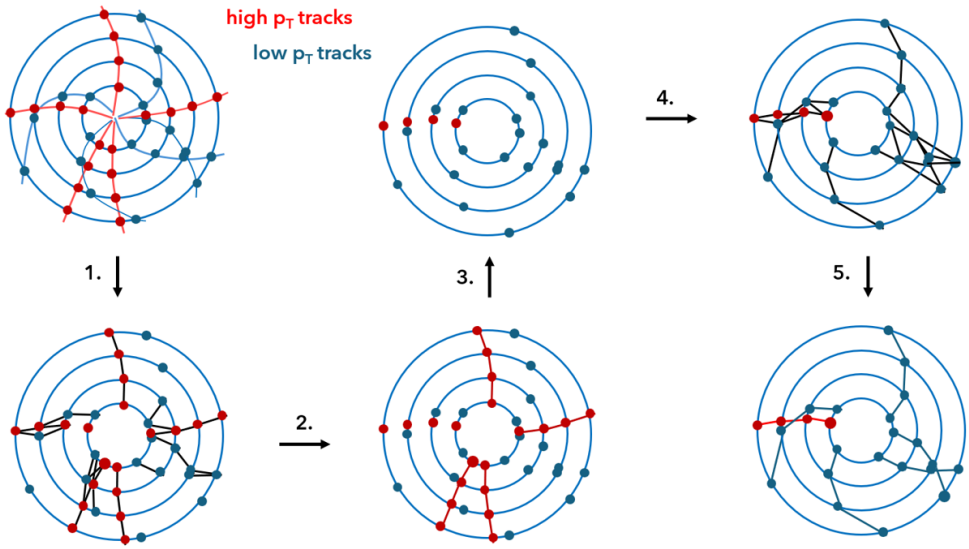


Figure 6.7: The suggested process: 1. Build edges associated with high p_T tracks. 2. Construct track candidates. 3. Remove associated hits. 4. Build remaining graph. 5. Construct track candidates.

pass through the pipeline to build the remaining candidates.

6.2.1 Stage One

Graph Construction: Filter

The initial $p_T > X$ graphs have been constructed, using the nets trained in Section 6.1.1. However, the graphs are still $\mathcal{O}^7$ edges large, and a filtering step is required.

Training a filter to recognise a momentum threshold for edges proved challenging. Initially, a filter was trained on 1000 graphs built to 99.5% efficiency with the $p_T > 2$ metric learning net, to score highly edges likewise with $p_T > 2$ (hence called filter A). This performed poorly: erratic behaviour increased during training, with monitored metrics fluctuating more and more. This is often an indication of overtraining, and here shows that the dataset, with the reduced graph size, is perhaps too small.

Another strategy was attempted: 1000 graphs were built to 99.5% efficiency for $p_T > 1$ (using the metric learning net trained for the baseline in Section 5.2.1). These were used to train a filter to score highly edges with

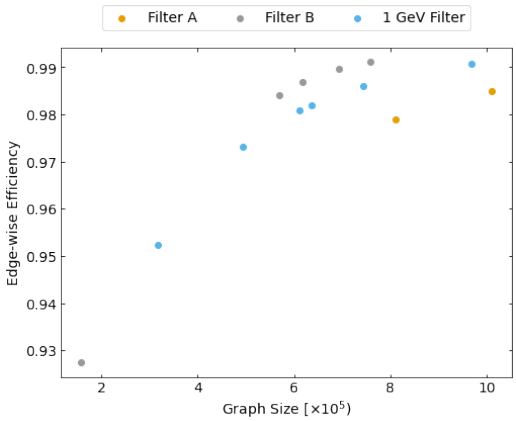
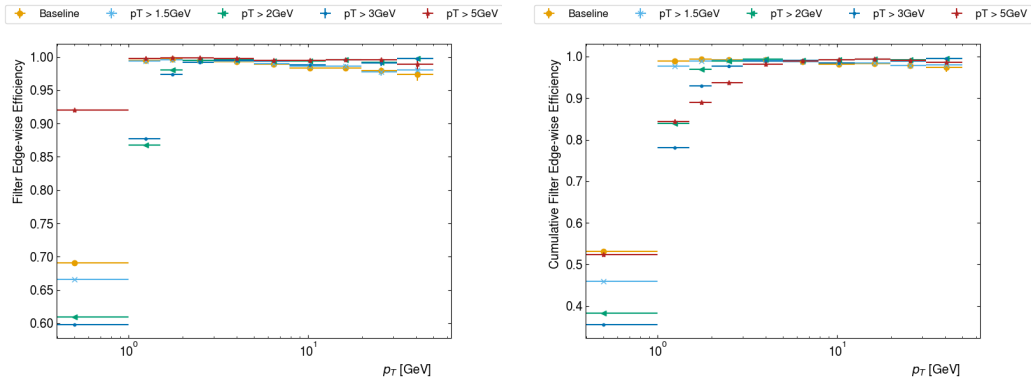


Figure 6.8: Filter edge-wise efficiency against graph size for 100 graphs post varying score cuts in interval [0.07, 0.12]. Graphs were constructed to 99.5% efficiency for $p_T > 2$ and then sent through filters A, B, and the filter trained on 1 GeV graphs in Section 5.2.1. .

$p_T > 2$ (filter B). Monitoring metrics during training showed much more stable behaviour. Both filters had the same architecture as laid out in 5.2.1. 100 events built with the $p_T > 2$ metric learning net were sent through both filters, as well as the filter trained for the baseline, and performance was measured at a variety of score cuts. Filter B performed considerably better than Filter A and also better than the baseline filter, as shown in Figure 6.8, which shows isolated filter edge-wise efficiency against graph size.

A filter for $p_T > 1.5$ graphs was trained following filter B's method, and proved successful. Attempting to train a filter for graphs $p_T > 3$ faced greater challenges. Similar strategies as those for Filters A and B were attempted but now to recognise the threshold $p_T > 3$; monitoring whilst training showed great fluctuations in metrics, and the final performance was poor. This may be due to the lack of target edges at this momentum, and for this reason, and due to the fact training filters is the most time-expensive part of this pipeline (taking at least a week), a filter for 5 GeV was unattempted. Instead, the filter B trained for $p_T > 2$ GeV was used on both sets of graphs: the logic followed that if low-momentum edges were not built initially by the metric learning net, then they could not pass the post-filter score cut regardless.

Four sets of 100 graphs built to 99.5% efficiency for target $p_T > [1.5, 2, 3, 5]$ were sent through the filtering net; edges with low scores were pruned. Figures 6.9 shows the sets of graphs pruned to 99% cumulative edge-wise ef-



(a) Filter efficiency vs. p_T

(b) Cumulative post-filter efficiency vs. p_T

Figure 6.9: The isolated filter and cumulative post-filter edge-wise efficiency as a function of p_T for the four sets of graphs and the baseline.

ficiency post filter, alongside the baseline 99% 1 GeV graphs. Figure 6.9a shows the isolated filter efficiency: the desired "step off" at $p_T < X$ is seen for the $X > 2, 3$ GeV graphs. Figure 6.9 shows the cumulative post-filter edge-wise efficiency, with performance as expected. The step off seen in the metric learning net has been strengthened for the $p_T > [2, 3]$ graphs due to the success of the filter.

Figure 6.10 shows the isolated filter edge-wise efficiency as a function of pseudorapidity for the graphs with target $p_T > [1, 1.5, 2]$ GeV. One can see the degradation of edge-wise efficiency seen in the central region ($|\eta| < 2$) is mitigated for the graphs with greater target p_T. This perhaps a reflection of the statistics: particles with large transverse momentum are found more frequently in the central region than the forward region, as can be seen in Figure 6.11. Conversely, efficiencies drop as $|\eta|$ approaches 4, where there is a reduction in tracks with high p_T (see Figure 6.5). It appears that the change in statistics has shifted the net's challenging areas from low $|\eta|$ to high $|\eta|$. Plots for $p_T > [3, 5]$ are shown in the Appendix.

In the baseline, it was found that a cumulative efficiency of 99% post-filter was required for maximum track efficiency. Here, this expectation does not necessarily follow: if only 95% of high p_T tracks are reconstructed in the first pass of the pipeline, the remaining 5% of high p_T tracks may be reconstructed in the second pass, alongside the low p_T tracks. As, for 1 GeV, the graphs that facilitated 96% of tracks to be reconstructed were 0.4 the size of the graphs that allowed 98.2% of tracks to be built, building graphs

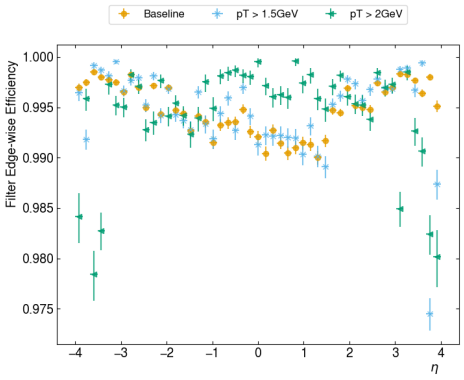


Figure 6.10: Isolated filter efficiency against η of 100 graphs, with cumulative edge-wise efficiency of 99%.

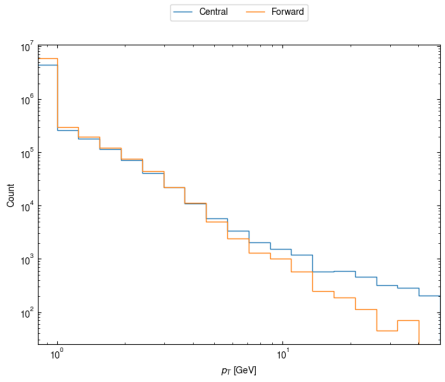
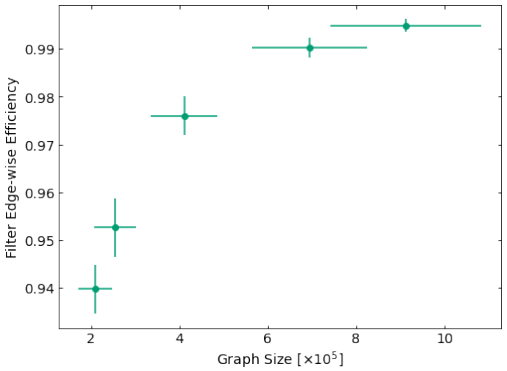
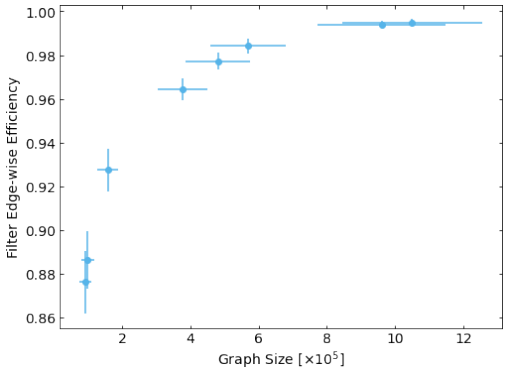


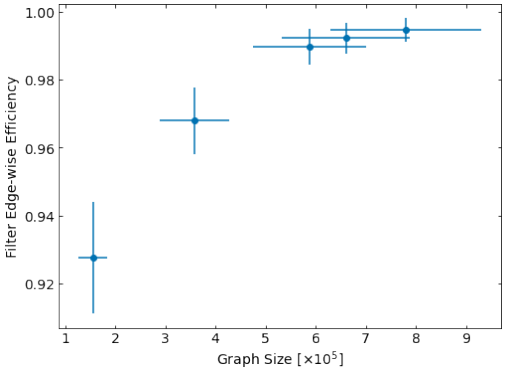
Figure 6.11: The momentum distribution of tracks in the central ($|\eta| < 2$) and forward ($|\eta| > 2$) regions of the ITk.



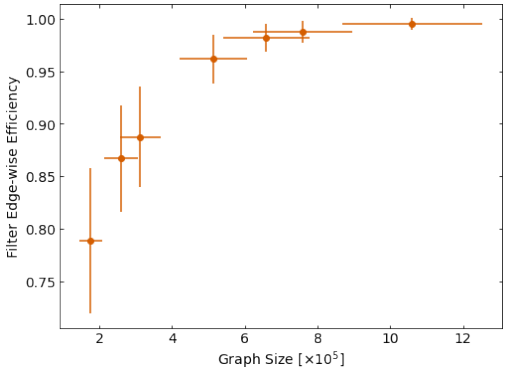
(a) Filter efficiency vs. graph size for graphs $p_T > 1.5$ GeV.



(b) Filter efficiency vs. graph size for graphs $p_T > 2$ GeV.



(c) Filter efficiency vs. graph size for graphs $p_T > 3$ GeV.



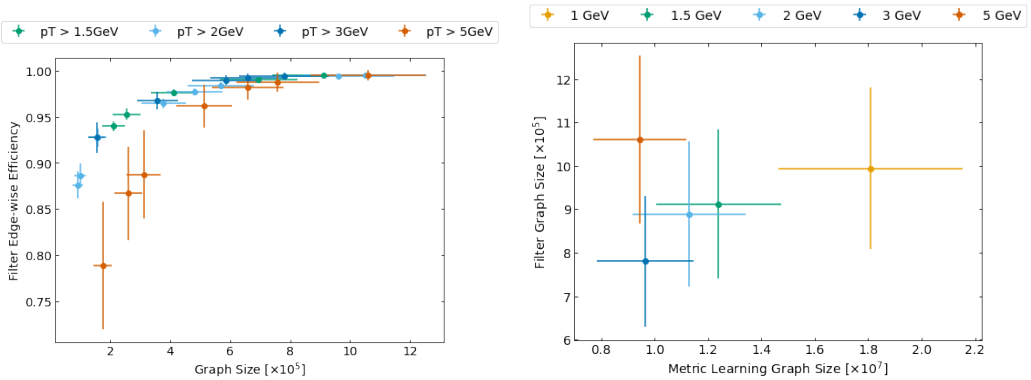
(d) Filter efficiency vs. graph size for graphs $p_T > 5$ GeV.

Figure 6.12: Isolated filter edge-wise efficiency against graph size at varying post-filter score cuts for the four sets of graphs.

with lower cumulative post-filter efficiencies is worth investigating.

Figure 6.12 shows the graph sizes and filter edge-wise efficiencies after pruning edges below varying scores in the interval [0.06, 0.12]. Figure 6.13a shows $p_T > [1.5, 2, 3]$ perform similarly: similar target efficiencies are being achieved for similar graph sizes. As the definition of target is dependent on the graph, a 95% target efficiency for $p_T > 1.5$ in essence means more edges with $p_T > 1$ GeV have been constructed than for a 95% efficient $p_T > 3$ graph. Again, we see a clear deterioration in 5 GeV.

Figure 6.13b shows the graph size of the sets of graphs at 99.5% efficiency after metric learning and 99% efficiency after a post-filter cut. It can be seen that (with the exception of $p_T > 5$) the stricter the target criteria, the smaller the graph. However, the scale of the reduction in graph size



(a) All Figures 6.12 plotted side by side for comparison.

(b) Graph sizes of the four sets of graphs and baseline post filter against post metric learning.

Figure 6.13

after metric learning, shown in Figure 6.6, is not sustained post-filter. This is unsurprising considering the difficulties of training the high-momentum filters, and suggests that greater gains may be possible if a better filter training strategy was found. It is also perhaps worth emphasising at this point the logic mentioned above: namely that due to the two stage process a lower post-filter cumulative efficiency than for the baseline might prove acceptable.

GNN

A GNN edge-scoring net was trained on 2000 graphs for 230 epochs, when monitored metrics plateaued and training was stopped. The training graphs were $p_T > 2$ graphs, with a post-filter threshold cut to give 98.5% cumulative edge-wise efficiency. The IGNN follows the same architecture as described in 5.2.2.

Figure 6.14 shows the GNN edge classification performance after training with the performance of the baseline GNN from 5.2.2 for comparison. We see the $p_T > 2$ net consistently outperforming the baseline, until rapid convergence at high signal efficiency.

Figure 6.15 shows the GNN edge classification performance, evaluated on the four sets of the graphs $p_T > [1.5, 2, 3, 5]$ at 99% input edge-wise efficiency. As expected, the GNN performs best on the $p_T > 2$ graphs, but performs well on $pT > [1.5, 3]$ graphs. Performance on $p_T > 5$ is unacceptable due to insufficient statistics.

Varying post-filter score cuts were taken for the four sets of graphs to

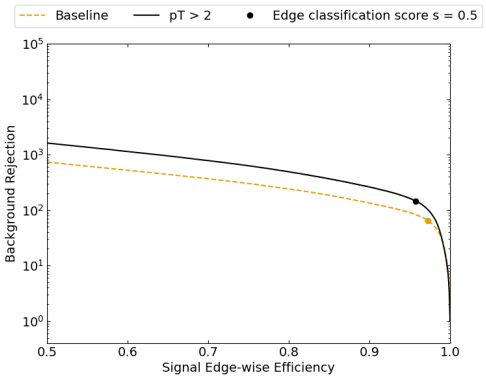


Figure 6.14: GNN classification performance, based on GNN edge-wise efficiency and GNN edge-wise fake rate.

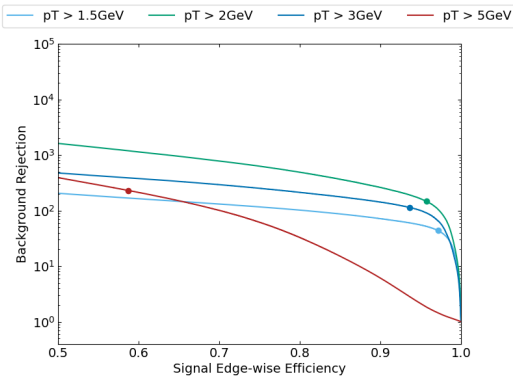
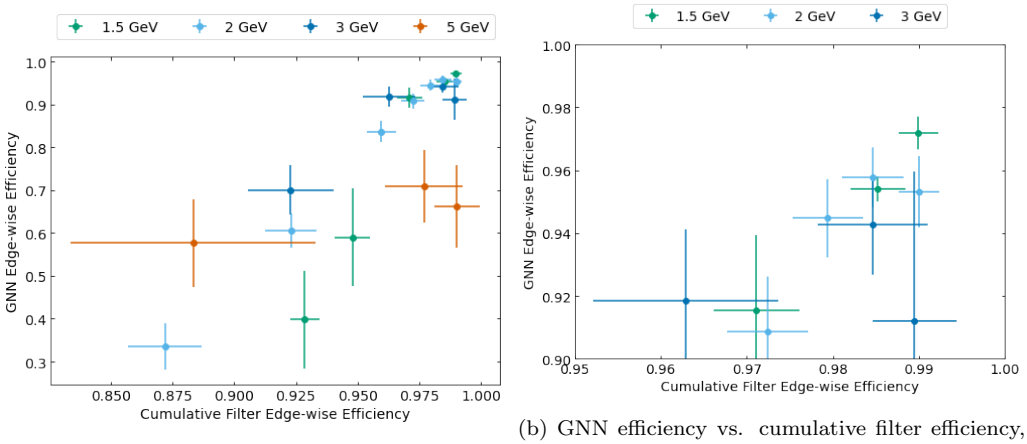


Figure 6.15: GNN classification performance, based on GNN edge-wise efficiency and GNN edge-wise fake rate, on the four sets of input graphs $p_T > [1.5, 2, 3, 5]$ at 99% target edge-wise efficiency. GNN trained on $p_T > 2$ 98.5% graphs. Marker shows edge classification score at s = 0.5.



(a) GNN efficiency vs. cumulative filter efficiency

(b) GNN efficiency vs. cumulative filter efficiency, zoomed.

Figure 6.16: GNN edge-wise efficiency at a cut of s = 0.5 against post-filter cumulative edge-wise efficiency for the four sets of graphs $p_T > [1.5, 2, 3, 5]$, at varying filter score cuts.

give cumulative edge-wise efficiencies within the interval [0.8, 1]. Figure 6.16 shows these efficiencies against the isolated GNN edge-wise efficiency, calculated with a post-GNN score cut of $s = 0.5$, with Figure 6.16b zoomed in to the region of interest, the high-efficiency interval. The reduction in GNN efficiency for highest filter efficiency for the $p_T > 2$ GeV graphs is unsurprising, as the GNN was trained on the 98.5% graphs. The reduction in GNN efficiency for the most efficient post-filter $p_T > 3$ GeV graphs is more puzzling: perhaps it reflects a limit on the graph impurity the GNN can effectively sift through.

Track Reconstruction

The ConnectedComponents algorithm for track reconstruction is henceforth deemed too variable and set aside. Track candidates were built using the CCandWalk algorithm from each set of $p_T > [1.5, 2, 3, 5]$ graphs with highest post-GNN edge-wise efficiency (see Figure 6.16). A track is matched and the associated particle is reconstructed according to the same criteria laid out in 5.2.3.

Table 6.2 shows, for target particles with $p_T > 1$ GeV, the final track efficiency, fake rate, and duplication rate, alongside the post-filter cut graph size. Figure 6.17 shows the track efficiency against p_T. One can see the

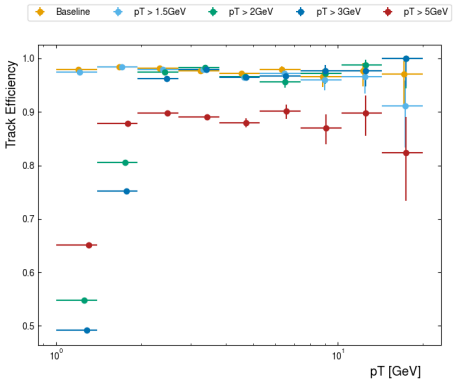


Figure 6.17: Track efficiency against p_T for the sets of graphs. x-values (but not errors) shifted slightly to view data points.

X	Track Efficiency		Fake Rate	Duplication Rate	Graph Size [$\times 10^5$]
	$p_T > 1$ GeV	$p_T > X$ GeV			
1	0.983 ± 0.004	-	0.058 ± 0.005	0.072 ± 0.009	10.2 ± 1.86
1.5	0.979 ± 0.004	0.977 ± 0.007	0.062 ± 0.006	0.097 ± 0.011	9.12 ± 1.71
2	0.637 ± 0.016	0.979 ± 0.008	0.062 ± 0.008	0.130 ± 0.024	6.97 ± 1.11
3	0.608 ± 0.016	0.980 ± 0.019	0.049 ± 0.008	0.104 ± 0.019	5.87 ± 1.13
5	0.715 ± 0.019	0.912 ± 0.076	0.024 ± 0.005	0.155 ± 0.023	7.59 ± 1.37

Table 6.2: Results from building track candidates using CCandWalk algorithm for $p_T > X$ graphs.

step off in efficiency for $p_T > [2, 3]$ GeV graphs has been carried through to the final stage: these graphs also perform better at the highest p_T than the baseline. It is also seen that the $p_T > 1.5$ graph performs very similarly to the baseline graph at low p_T, which suggests the difference in trajectories between particles with $p_T = 1$ GeV and $p_T = 1.5$ GeV may be too similar.

It has been demonstrated that it is highly possible to choose a threshold of minimum p_T and develop a track reconstruction pipeline that targets it, and that this can produce similar target track efficiencies with not only smaller graphs but also slimmer networks. However, the objective is to reconstruct particles with $p_T > 1$ GeV: this will be met in the following Section.

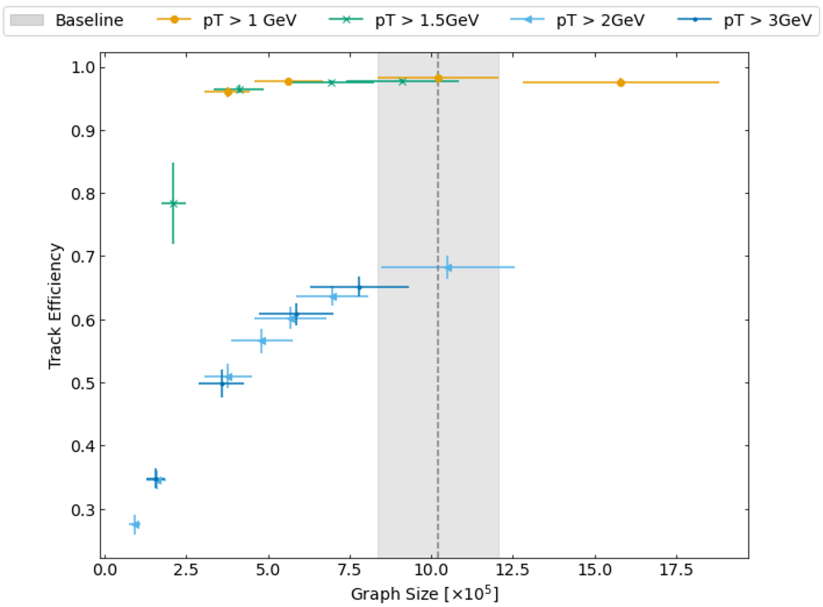


Figure 6.18: Track efficiency, $p_T > 1$ GeV target tracks, vs. post-filter graph size

Iteratively

The process (as shown in Figure 6.7) now should proceed to Stage 2. Two reasons indicate a high track efficiency in Stage 1 is not necessarily the best way forward for Stage 2: firstly, the large increase in graph size for the last few percent of efficiency gained, (as seen in, for example, Figures 5.5, 6.13a); secondly, the continual struggle the $p_T > 5$ GeV graphs showed with limited target statistics. It is also for this reason that the 5 GeV graphs have at this point been discontinued.

In light of this, for each X, track candidates were built from a variety of smaller graphs with lower edge-wise efficiencies for each of the three $p_T > [1.5, 2, 3]$ sets. Stage 1 track efficiency against graph size is shown in Figure 6.18. As the objective is to reduce graph size, there is no point in taking graphs larger than the baseline. A variety of graphs from Stage 1 were chosen to proceed to Stage 2: these are shown in Table 6.3.

Label	X	Track Efficiency		Duplication Rate	Fake Rate	Graph Size [$\times 10^5$]
		$p_T > 1$ GeV	$p_T > X$ GeV			
A	1.5	0.784 ± 0.064	0.811 ± 0.070	0.461 ± 0.037	0.069 ± 0.015	2.10 ± 0.38
B	1.5	0.965 ± 0.006	0.972 ± 0.008	0.143 ± 0.024	0.059 ± 0.007	4.11 ± 0.76
C	2	0.346 ± 0.015	0.869 ± 0.024	0.219 ± 0.029	0.036 ± 0.008	1.58 ± 0.29
D	2	0.602 ± 0.018	0.977 ± 0.009	0.129 ± 0.027	0.054 ± 0.008	5.69 ± 1.11
E	3	0.349 ± 0.016	0.903 ± 0.042	0.197 ± 0.029	0.033 ± 0.008	1.55 ± 0.29
F	3	0.498 ± 0.022	0.970 ± 0.022	0.143 ± 0.024	0.043 ± 0.008	3.58 ± 0.69

Table 6.3: Results from Stage 1 of the sets of graphs chosen to continue to stage 2.

6.2.2 Stage 2

In Stage 2, hits associated with all track candidates built in Stage 1 are removed from the event. Associated truth information is also removed, such that the evaluation during Stage 2 does not reflect the absence of particles that have already been reconstructed. The trimmed down events are then passed through the pipeline once more to build remaining track candidates. Nets used were those trained for the baseline, as now the target particles are the same (i.e. $p_T > 1$ GeV). Indeed, brief studies training metric learning nets on reduced events showed a slight deterioration in performance, perhaps due to the reduction in target statistics.

For each interval X:$[1.5, 2, 3, 5]$, two graphs were chosen: one with a high and one with a low Stage 1 track efficiency, in order to identify a good Stage 1 tracking efficiency and Stage 2 post-filter efficiency for maximum final track efficiency and minimum combined graph size.

Figure 6.19 shows the hits associated with track candidates built in Stage 1 and thus removed for Stage 2.

The naive expectation would be for a linear increase in hits with track efficiency; in actuality, tracks with lower p_T (that are more likely to be constructed in the $p_T > 1.5$ graphs than the the $p_T > [2, 3]$) graphs tend to leave more hits (as seen in Figure 6.20) - so a greater-than-linear increase in hits would be expected. The 1.5 GeV set of graphs labelled (i) does not match this expectation because of the quality of it's tracks. This set of graphs has the greatest duplication rate of all the graphs plotted. The criteria for a reconstructed track, laid out in Section 4, notes that a track and associated

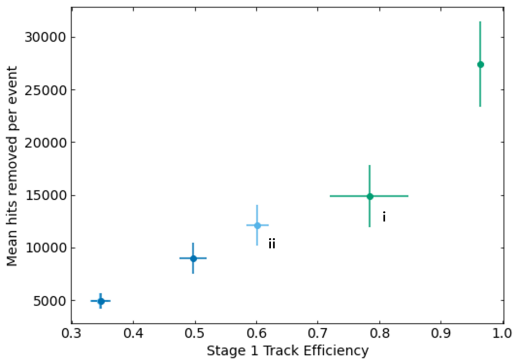


Figure 6.19: Hits removed vs stage 1 track efficiency. (i) and (ii) labelled for discussion.

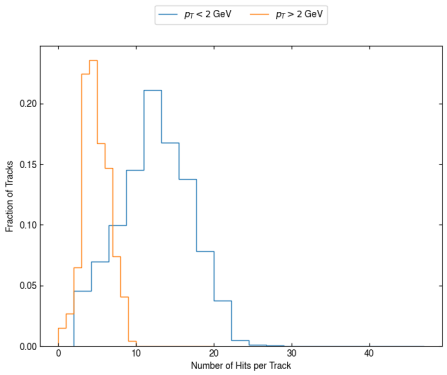


Figure 6.20: Number of hits per tracks for $p_T > 2$ GeV and $p_T < 2$ GeV.

particle must shared at least half the track's hits - but the particle has no limit to extra hits. The poor *target* track efficiency and high duplication rate both result from a poorly scored graph. The lack of hits for (i) despite high *overall* track efficiency, then, is because many hits from a reconstructed particle are not part of the matched track and are therefore not removed. This is confirmed by studying the ratio of hits associated with tracks and hits associated with reconstructed particles: for (i), it is 33.5%, whereas for (ii), it is 60% (this is for all pT not $pT > 1$).

Each X = [1.5, 2, 3] will now be discussed briefly, highlighting some pertinent distributions from Stage 1 and proceeding quickly through Stage 2, before an overall summary of the final results to conclude.

6.2.3 X = 1.5 GeV

Here the results for the set of graphs with momentum threshold $X = 1.5$ GeV will be discussed.

Two sets of graphs, A and B (as from Table 6.3) were continued to Stage 2. Figures 6.21a and 6.21b show the Stage 1 p_T distribution of the edge-wise efficiency and track efficiency respectively. One can see the typical drop in edge-wise efficiency at high-p_T seen also in the 1 GeV filters, due to a lack of statistics here.

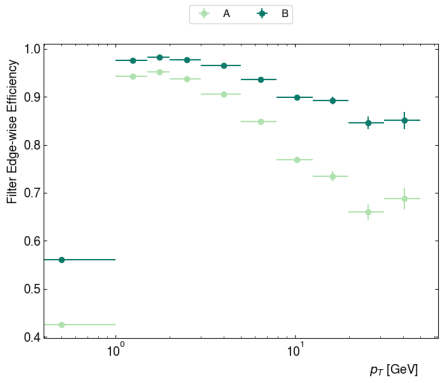
The corresponding hits removed are shown in Figure 6.21c. The reduced events were sent through the metric learning and filter net trained in Section 5.2.1. Graph size against post-filter cumulative efficiency is shown in Figure 6.21d, with post-filter score cuts of [0.07, 0.08, 0.09, 0.10, 0.11] made for each set of graphs. One can see greater post-filter efficiencies at smaller graph sizes for A over B: perhaps surprising, given that B contains fewer hits. The true target edges graphs B must contain for high edge-wise efficiency are those not part of Stage 1 track candidates, only 3.5% of target tracks. Graphs A, on the other hand, must contain edges associated with the 21.6% of target tracks not reconstructed in Stage 1. A common theme so far has been nets struggling when statistics are too low: recall the initial plateau in reduction of graph size for nets training on $p_T > 5$, Figure 6.6. Entering Stage 2, only 0.1% of nodes in B are from target particles; in contrast, 0.8% are from target particles in A. Of course, the final Stage 2 track efficiency and thus the edge-wise efficiency here must be greater for graphs A than B in order to achieve an acceptable combined track efficiency.

Post-filter score cuts were chosen such that the combined graph size of Stages 1 and 2 was smaller than the baseline, 1.02×10^6. These are identified by the X markers in Figure 6.21d.

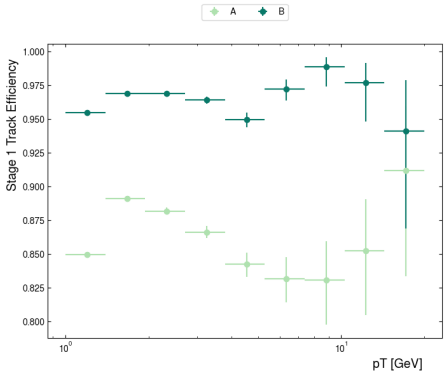
These graphs were sent through the edge-scoring IGNN trained in 5.2.2, and then tracks were built using the CCandWalk algorithm. Figure 6.21 shows Stage 2 track efficiency (6.21e) and combined track efficiency (6.21f) against transverse momentum. A summary of results is shown in Table 6.4.

6.2.4 X = 2 GeV

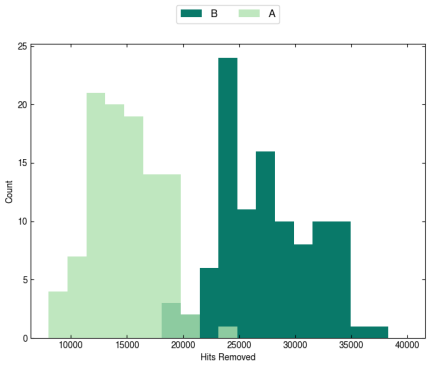
Two sets of graphs, C and D (as from Table 6.3), were continued to Stage 2. Figure 6.22a shows the Stage 1 post-filter edge-wise efficiency and Figure 6.22b the track efficiency of both sets as a function of p_T.



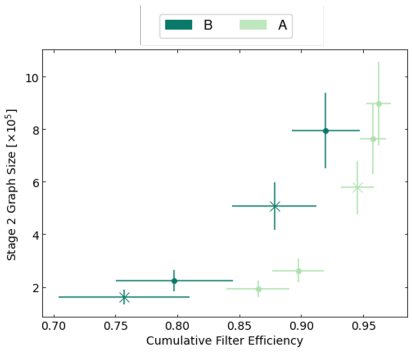
(a) Isolated filter edge-wise efficiency vs. p_T



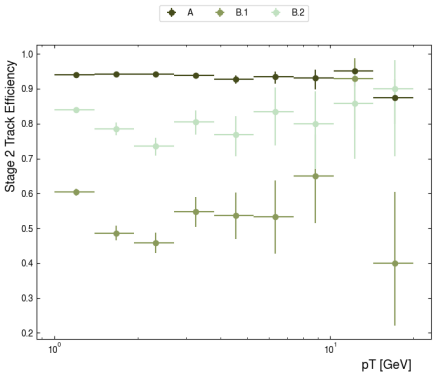
(b) Stage 1 track efficiency vs. p_T



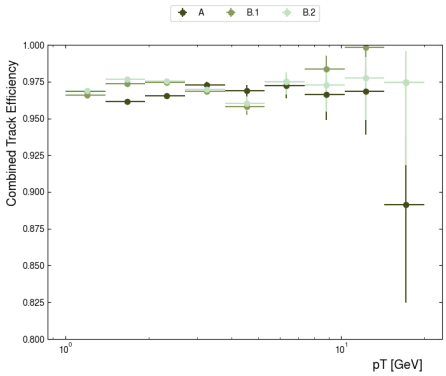
(c) Hits removed per event for Stage 2 of graphs X = 1.5.



(d) Graph size against post-filter cumulative efficiency at varying score cuts of sets of graphs A and B in Stage 2.

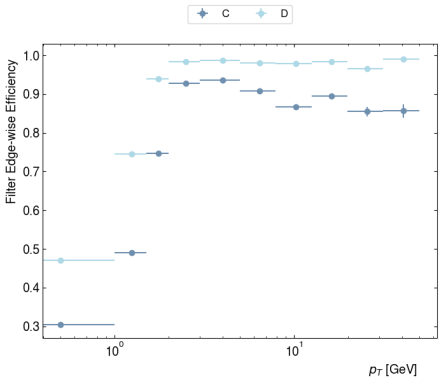


(e) Stage 2 track efficiency vs. p_T

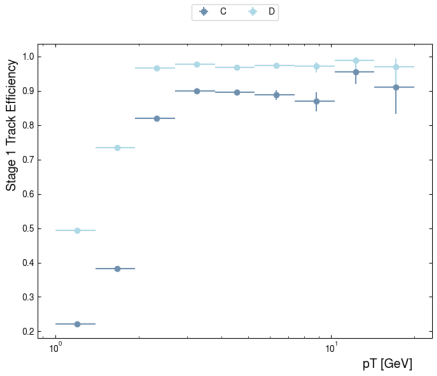


(f) Combined track efficiency vs. p_T

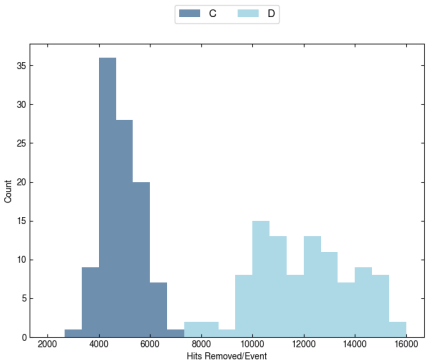
Figure 6.21: A summary of graphs A and B through Stage 1 and 2.



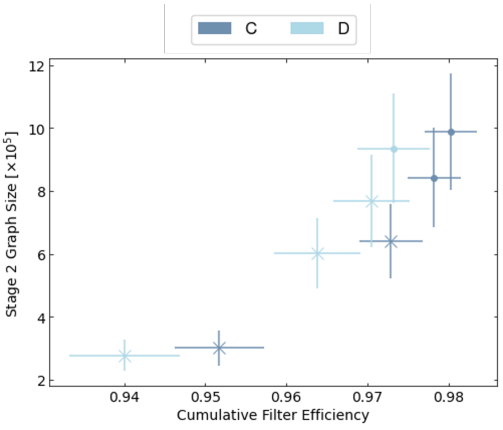
(a) Isolated filter edge-wise efficiency vs. p_T



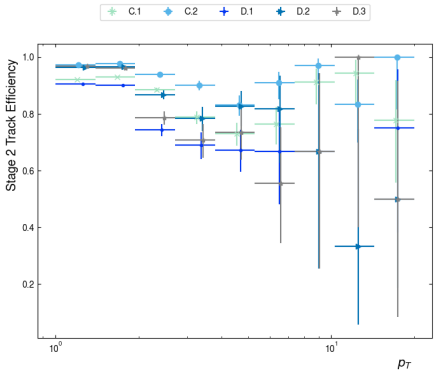
(b) Stage 1 track efficiency vs. p_T



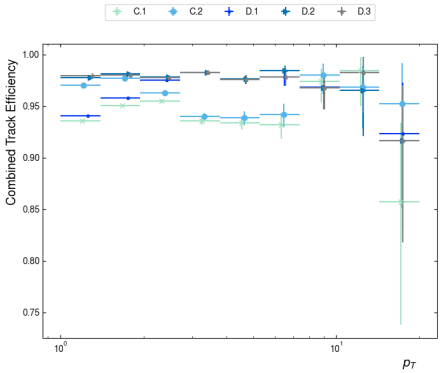
(c) Hits removed per event.



(d) Stage 2 graph size against post-filter cumulative efficiency at varying score cuts.



(e) Stage 2 track efficiency vs. p_T. x-values (but not errors) shifted for ease of view.



(f) Combined track efficiency vs. p_T. x-values (but not errors) shifted for ease of view.

Figure 6.22: A summary for graphs C and D through Stage 1 and 2.

Label	Combined Track Efficiency	Combined Graph Size [$\times 10^5$]
A	0.984	7.88
B.1	0.973	5.71
B.2	0.987	9.81

Table 6.4: Final results for sets of graphs X = 1.5 GeV

The corresponding hits removed from each set are shown in Figure 6.22c. These reduced events were sent through the metric learning and filter net trained in 5.2.1. Graph size against post-filter cumulative efficiency is shown in Figure 6.22d, with post-filter score cuts of [0.07, 0.08, 0.09, 0.10] made for each set of graphs.

Cuts identified by the X marker in Figure 6.22d were chosen to continue through the pipeline by considering how many tracks were left to build and how large a graph was acceptable: thus three sets of graphs continued through the pipeline. A summary of these is in Table 6.5.

These were sent through the edge-scoring IGNN trained in Section 5.2.2, and then tracks were built using the CCandWalk algorithm. The Stage 2 track efficiency and combined track efficiency as a function of p_T are shown in Figures 6.22e and 6.22f respectively; a summary of results is shown in Table 6.5

One can see the large error bars for graphs stemming from C in Stage 2. In general, ...

Label	Combined Track Efficiency	Combined Graph Size [$\times 10^5$]
C.1	0.955	4.58
C.2	0.984	7.98
D.1	0.964	8.46
D.2	0.991	11.71
D.3	0.986	13.38

Table 6.5: Final results, X = 2

6.2.5 3 GeV

The two sets of graphs, E and F as from Table 6.3, were continued to Stage 2; Figures 6.23 and 6.23b shows the post-filter edge-wise and track efficiency in

Label	Combined Track Efficiency	Combined Graph Size [$\times 10^5$]
E.1	0.933	5.10
E.2	0.988	8.41
F	0.989	9.77

Table 6.6: Summary of X = 3 GeV graphs.

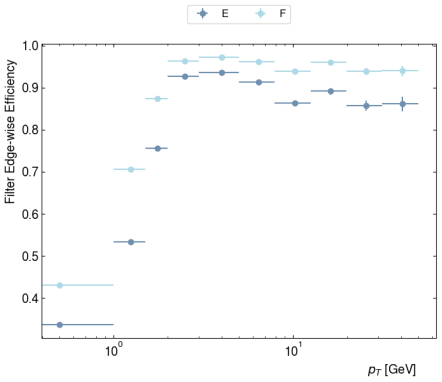
Stage 1 respectively, with corresponding hits removed shown in Figure 6.23c. These reduced events were sent through the metric learning and filter net trained in 5.2.1. Stage 2 graph size against cumulative post-filter edge-wise efficiency is shown in Figure 6.23d, with post-filter score cuts in the interval [0.07, 0.10] made for each set of graphs. Graphs indicated by the X marker were scored by the IGNN and then tracks were built using the CCandWalk algorithm. Figures 6.23e and 6.23f show the Stage 2 and combined track efficiency as a function of p_T respectively; a summary of results is shown in Table 6.6.

6.3 Overall

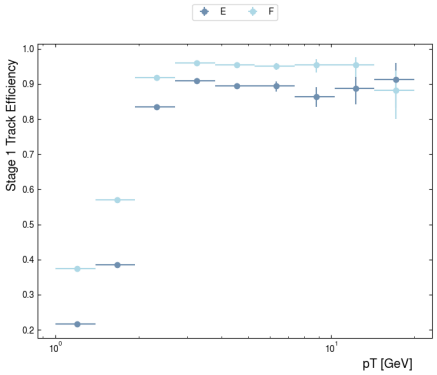
Table 6.7 shows all results together for quick reference. For all X, routes have been found that have culminated in greater track efficiencies and smaller graphs than the baseline; however, this is often at the cost of very large duplication rates (and slightly higher fake rates). $p_T > 3$ produces the route with highest combined track efficiency, 0.989 ± 0.003; this route has a combined graph size smaller than the baseline.

Figure 6.24 shows the graph sizes of all graphs with final track efficiencies better than 0.983, the baseline. Graphs to the left of the grey bar also have smaller combined graph sizes.

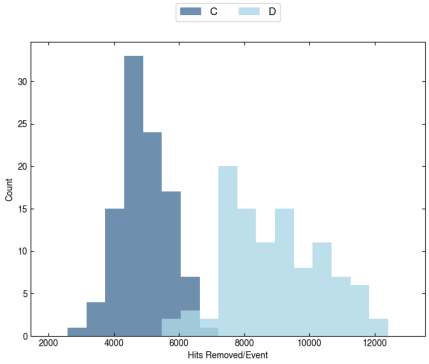
The desired route should have, in both Stage 1 and Stage 2, high track efficiencies within a p_T interval and low track efficiencies outside of it. Mid-range track efficiencies indicate a graph that is ambiguously scored: this leaves room for the CCandWalk algorithm to be misled, or for track candidates to be broken.



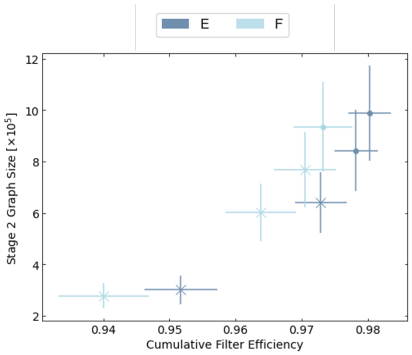
(a) Isolated filter edge-wise efficiency vs. p_T



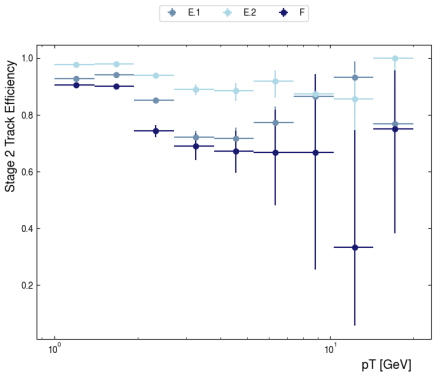
(b) Stage 1 track efficiency vs. p_T



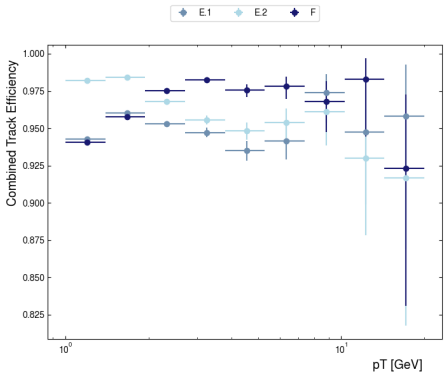
(c) Hits removed per event.



(d) Caption



(e) Stage 2 track efficiency vs. p_T



(f) Combined track efficiency vs. p_T

Figure 6.23: X = 3

X	Track Efficiency	Graph Size [$\times 10^5$]	Duplication Rate	Fake Rate
1	0.983	10.2	0.072 ± 0.009	0.058 ± 0.005
1.5	0.984	7.88	0.398 ± 0.06	0.086 ± 0.018
1.5	0.973	5.71	0.121 ± 0.09	0.106 ± 0.025
1.5	0.987	9.81	0.123 ± 0.08	0.103 ± 0.013
2	0.955	4.58	0.258 ± 0.066	0.070 ± 0.011
2	0.984	7.98	0.185 ± 0.031	0.067 ± 0.011
2	0.964	8.46	0.265 ± 0.045	0.085 ± 0.014
2	0.991	11.71	0.130 ± 0.030	0.065 ± 0.012
2	0.986	13.38	0.158 ± 0.030	0.076 ± 0.011
3	0.933	5.10	0.232 ± 0.038	0.035 ± 0.009
3	0.988	8.41	0.130 ± 0.020	0.066 ± 0.006
3	0.989	9.77	0.174 ± 0.018	0.074 ± 0.010

Table 6.7: Overall summary of combined results

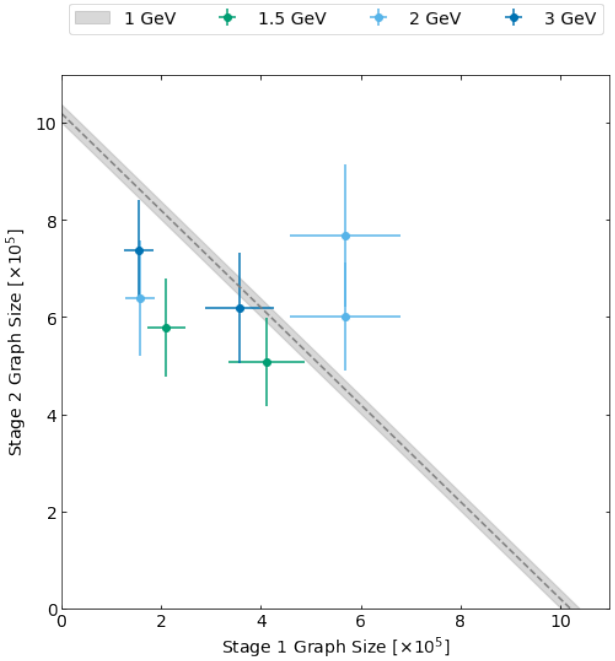


Figure 6.24: The graph sizes of all sets of graphs with final track efficiencies greater than or equal to 0.982.

Chapter 7

Conclusions and Outlook

The objective of this Master's thesis was to assess the capacity of graph neural networks to act as a method of track finding within the Event Filter, and to investigate a potential path of memory-footprint reduction. The high-luminosity era of the LHC necessitates an upgrade to the ATLAS TDAQ system that must take advantage of new innovations to meet requirements; graph neural networks are a powerful tool that appear inherently suited to the task and take advantage of highly parallel computing hardware.

The general capacity of graph neural networks to act as a method of track finding has been assessed in Chapter 5. A track efficiency of 98.3 % is achieved using the baseline method; this requires a graph containing 1.02×10^6 edges. Particles not reconstructed are primarily those with high transverse momentum and from the barrel region of the ITk. A theme of the GNN pipeline's ability to reconstruct tracks within the ATLAS ITk is poor performance within the strip barrel. A future avenue to mitigate this might be to train individual networks for the strip and pixel sub-detectors; this may allow the network to learn the differing clustering features to greater effect.

The baseline track efficiency is competitive but uses a large, impure graph. The ability of a GNN-based track finding pipeline to specialise to tracks with transverse momentums greater than a threshold is assessed, to investigate potential routes that result in smaller graphs. Metric learning nets, in particular, are very suited to this task, with a step off in edge-wise efficiency below the p_T threshold resulting in a sharp reduction in graph size. There is some difficulty in encouraging the filtering nets to preferentially score high-p_T edges: this slightly mitigates the scale of the reduction in graph size. The IGNN edge scoring net performs well and facilitates a final track effi-

ciency that is high over the p_T threshold and low elsewhere, particularly for $p_T > [2, 3]$ GeV.

The iterative process of building Stage 1 track candidates, removal of associated hits, and building of Stage 2 track candidates, can produce a final track efficiency of 0.989 ± 0.003, for a combined set of graphs that are smaller than the single set of graphs used to build 0.983 of tracks. In this way, the approach can be deemed a success: however, it is at the expense of a greater duplication rate. This is primarily the result of broken tracks due to ambiguously-scored graphs: an avenue for exploration in the future would be a post-pipeline algorithm that could potentially stitch these tracks back together.

Throughout this thesis, a repeating theme has been poor performance at high-p_T due to a lack of statistics. This could be addressed by using a sample with a greater number of high-p_T particles. The step off in edge-wise efficiency seen below the specified p_T threshold is gradual: the $p_T > 1.5$ GeV graphs reconstruct almost as many tracks in the $1 > p_T > 1.5$ GeV region as the $p_T > 1$ GeV graphs do. For this reason, it is beneficial to use a higher threshold - but very quickly, training problems due to lack of statistics are encountered - for example, throughout the $p_T > 5$ GeV graphs. It would be interesting to investigate a well-trained pipeline at a higher threshold than 3 GeV.

Finally, to truly verify whether this approach is feasible for the Event Filter, it must be compared to the state-of-the-art method, the combinatorial Kalman Filter. For this, one must implement the iterative approach into a tracking framework such as Athena.

Bibliography

[1] Particle data group. https://pdg.lbl.gov/2024/tables/contents$_t ables.html$.

[2] P. Bryant L. Evans. Lhc machine. *JINST*, 3, 2008.

[3] The ATLAS Collaboration. The cern accelerator complex, layout in 2022. https://cds.cern.ch/record/2800984, 2022.

[4] Emma Ward. Lhc and hl-lhc timeline for atlas website. http://cds.cern.ch/record/2652466, 2018.

[5] The ATLAS Collaboration. Technical proposal for a general purpose pp experiment at the large hadron collider at cern. 1994.

[6] The ATLAS Collaboration. The atlas experiment at the cern large hadron collider. 2008.

[7] The ATLAS Collaboration. The atlas experiment at the cern large hadron collider. *JINST*, 3, 2008.

[8] The ATLAS Collaboration. The atlas trigger system for lhc run 3 and trigger performance in 2022. *JINST*, 19, 2022.

[9] The ATLAS Collaboration. Technical design report for the phase-ii upgrade of the atlas tdaq system.

[10] The ATLAS Collaboration. Atlas inner tracker pixel detector: Technical design report. *ATLAS-TDR-030*, 2017.

[11] The ATLAS Collaboration. Technical design report for the atlas inner tracker strip detector. *ATLAS-TDR-025*, 2017.

[12] The ATLAS Collaboration. Technical design report for the phase-ii upgrade of the atlas trigger and data acquisition system - event filter tracking amendment. 2022.

[13] The ATLAS Collaboration. Expected tracking and related performance with the updated atlas inner tracker layout at the high-luminosity lhc, 2021.

[14] The ATLAS Collaboration. Atlas run 3 charged particle track seed finding performance. *ATLAS-TDR-025*, 2023.

[15] The ATLAS Collaboration. Atlas software and computing hl-lhc roadmap.

[16] J. Vlimant J. Shlomi, P. Battaglia. Graph neural networks in particle physics. 2020.

[17] S. Farrell *et al.* Novel deep learning methods for track reconstruction. *Connecting the Dots*, 2018.

[18] The ATLAS Collaboration. Atlas itk track reconstruction with a gnn-based pipeline. *ATL-ITK-PROC-2022-006*, 2022.

[19] The ATLAS Collaboration. Physics performance of the atlas gnn4itk track reconstruction chain. *The European Physical Journal Conferences*, 295, 2023.

[20] B. Denby. Neural networks and cellular automata in experimental high energy physics. *Computer Physics Communications*, 49:429–448, 1988.

[21] J. Ba D. Kingma. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

[22] Charu C. Aggarwal. *Neural Networks and Deep Learning*. Springer, Cham, Switzerland, 2018.

[23] P. Battglia et. al. Interaction networks for learning about objects, relations and physics. *arXiv:1612.00222*, 2016.

[24] The ATLAS Collaboration. Track finding performance plots for a graph neural network pipeline on atlas itk simulated data, 2022.

[25] F. Karray B. Ghojogh, A. Ghodsi and M. Crowley. Spectral, probabilistic, and deep metric learning: Tutorial and survey. 2022.

[26] S. Dittmeier. Track reconstruction for the atlas phase-ii event filter using gnns on fpgas., 2024.

[27] The ATLAS Collaboration. Expected tracking and related performance with the updated atlas inner tracker layout at the high-luminosity lhc. *ATL-PHYS-PUB-2021-024*, 2021.

[28] Weights and biases. : https://wandb.ai/site.

[29] j. Leskovec W. Hamilton, R. Ying. Inductive representation learning on large graphs. *arXiv:1706.02216*, 2017.