Deep Learning Applications for Particle Physics in Tracking and Calorimetry

Alexander Schuy

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2023

Reading Committee:
Shih-Chieh Hsu, Chair
Gordon T. Watts
Scott Hauck

Program Authorized to Offer Degree:
Physics

University of Washington

Abstract

Deep Learning Applications for Particle Physics in Tracking and Calorimetry

Alexander Schuy

Chair of the Supervisory Committee:
Shih-Chieh Hsu
Physics Department

This thesis presents an in-depth exploration of advanced deep learning applications in particle physics, particularly in the context of tracking, calorimetry, and energy reconstruction within High Energy Physics (HEP) experiments. It encompasses three studies, each underscoring the potential of deep learning to address the increasing computational demands posed by more powerful particle accelerators and their complex datasets. The first study highlights the application of Graph Neural Networks (GNNs) in the Exa.TrkX project for efficient particle tracking. The second study focuses on the DeepCalo model, a multi-modal deep learning model incorporating FiLM layers, dense layers, and convolutional layers, adept at processing ECAL data, tracks, and high-level scalars. This study specifically demonstrates the novel implementation of DeepCalo on Field-Programmable Gate Arrays (FPGAs) for low-latency applications in particle physics experiments. The third study evaluates the use of Sparse Point-Voxel Convolutional Neural Networks (SPVCNN) for clustering energy deposits in hadronic showers, showcasing its potential for real-time data analysis in high-energy environments. Collectively, these studies not only exhibit the adaptability and computational efficiency of deep learning models in HEP but also indicate their critical role in managing the computational load of future high-luminosity experiments, which may be vital to addressing some of the most pressing challenges in modern particle physics, such as understanding dark matter.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to express my heartfelt gratitude to the University of Washington, particularly the Physics Department's staff and faculty, for their unwavering support and guidance throughout my academic journey. Their dedication to excellence in education and research has been instrumental in shaping my intellectual growth and personal development.

I am profoundly thankful to my research advisor, Professor Shih-Chieh Hsu, for his exceptional mentorship, invaluable insights, and encouragement through challenging times. Professor Hsu's unwavering kindness and compassion, as expressed in his everyday actions, have been a constant source of inspiration and support. Whether in our meetings, his teaching, or other departmental and outreach activities, his dedication to nurturing not only the academic growth but also the well-being of those around him is exceptional. I am deeply grateful for the privilege of working under his mentorship and for the impact he has had on my life and career.

I would also like to extend my gratitude to my mentor, Professor Scott Hauck, for his guidance, patience, and willingness to share his time and expertise.

I am deeply appreciative of the entire community at the University of Washington, as I have had the privilege of learning from and collaborating with many talented individuals who have contributed to my growth as a researcher and as a person.

Lastly, I would like to thank my family and friends for their unwavering support and encouragement throughout this journey. Your belief in me has been a source of strength and motivation.

I am truly grateful for the opportunities and experiences I have had at the University of Washington, and I look forward to building upon the strong foundation I have gained here.

Thank you once again for your support and guidance.

This material is based upon work supported by the National Science Foundation under Grant No. 1934360 (Collaborative Research: Advancing Science with Accelerated Machine Learning - ASAML), 2117997 (HDR Institute: Accelerated AI Algorithms for Data-Driven Discovery - A3D3) and 2110963 (Accelerating Searches for Beyond the Standard Model Physics and the ATLAS Pixel Detector).

Chapter 1

INTRODUCTION

1.1 Theory

Particle physics is the branch of science dedicated to the study of the fundamental constituents of matter and their interactions. It seeks to answer profound questions about the nature of the universe, from the smallest scales of subatomic particles to the vast cosmos itself. Rooted in centuries of scientific inquiry, our current understanding of this field is encapsulated in the Standard Model, a theoretical framework built in the past century that, while immensely successful, still holds unresolved mysteries. This section describes the current state of the Standard Model and discusses some of the current challenges in the field.

The Standard Model succinctly describes three of the four known fundamental interactions that govern the behavior of particles: the electromagnetic, weak, and strong forces. The electromagnetic interaction operates between electrically charged particles. The weak interaction is responsible for processes such as beta decay in atomic nuclei. The strong interaction holds quarks together within protons, neutrons, and other hadrons. Notably absent is gravity, the fourth fundamental interaction, which remains outside the current scope of the model.

Within the Standard Model, particles are organized into quarks, leptons, and bosons. The six quarks are split into three generations: the first comprising the up and down quarks, the second consisting of the charm and strange quarks, and the third including the top and bottom quarks. Similarly, leptons come in three generations: the first has the electron and electron neutrino, the second contains the muon and muon neutrino, and the third features the tau and tau neutrino. Bosons, on the other hand, act as force carriers. The photon mediates the electromagnetic interaction, gluons mediate the strong force, and the W and Z

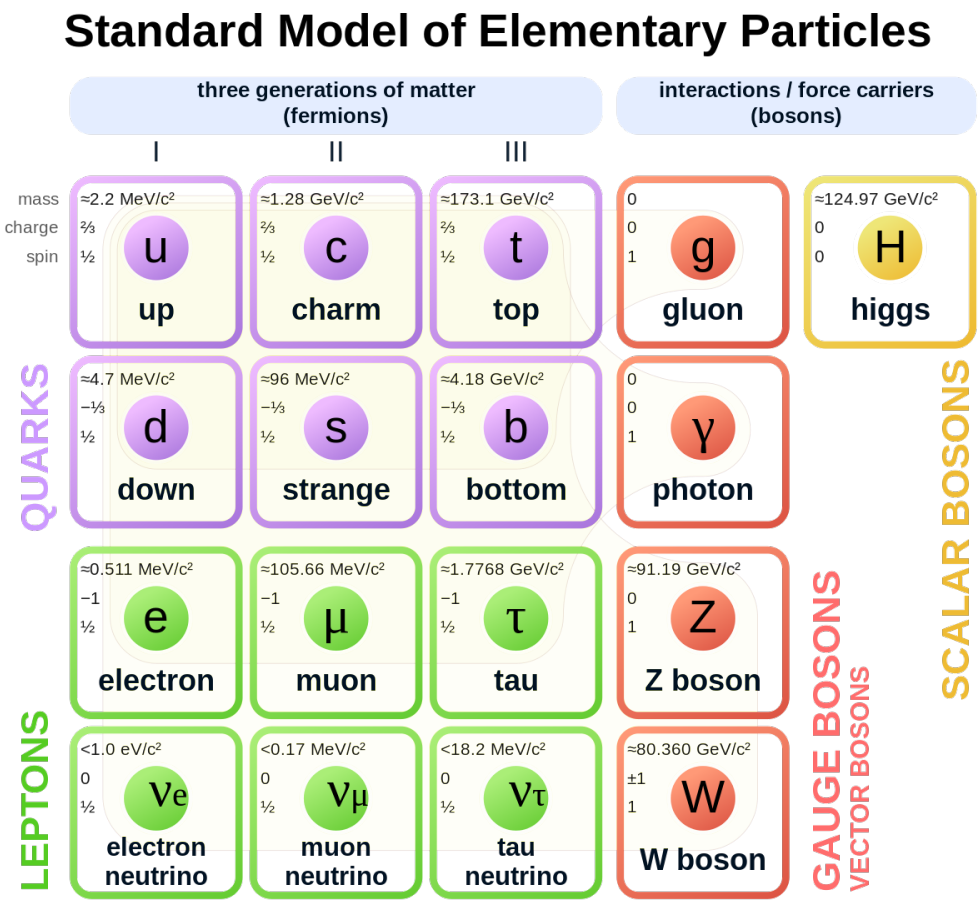

Figure 1.1: Comprehensive visualization of the standard model of particle physics [1]. This diagram showcases the fundamental particles, including quarks, leptons, and bosons, grouped by their respective types. Additionally, they are organized according to their generation of matter.

bosons mediate the weak force. The Higgs boson, a unique scalar particle, imparts mass to particles via the Higgs mechanism.

A fundamental quantum property distinguishing these particles is spin. This intrinsic form of angular momentum is foundational to quantum mechanics. Scalar particles, like the Higgs boson, possess zero spin. Fermions, which encompass both quarks and leptons, have half-integer spins, such as $+\frac{1}{2}$ or $-\frac{1}{2}$. This spin property causes fermions to adhere to the Pauli exclusion principle. In contrast, bosons have integer spins, like 0 or 1, allowing multiple bosons to coexist in the same quantum state. Notably, the gauge bosons (photon, W, Z, and gluons) in the Standard Model are vector particles with a spin of 1.

Among the fundamental particles described above, the Higgs boson stands out due to its unique significance and the monumental effort dedicated to its discovery. The Higgs boson is the result of excitations of the Higgs field. The Higgs field is a scalar field with two electrically charged and two neutral components, which together form a complex doublet of the weak isospin SU(2) symmetry. Long theorized, the Higgs boson was postulated in the 1960s as an essential part of the Higgs mechanism, a process through which particles acquire mass [11][12][13][14][15]. The mechanism operates within the framework of spontaneous symmetry breaking (SSB) in the electroweak sector. In the absence of the Higgs mechanism, the fundamental gauge bosons of the weak force—the W and Z bosons—would remain massless, contradicting experimental observations.

While the theoretical framework behind the Higgs mechanism was well-established, proving the existence of the associated boson proved elusive for decades. It wasn't until 2012 that a breakthrough came. Using the Large Hadron Collider (LHC) at CERN (see Fig. 1.2), the world's most powerful particle accelerator at the time, experiments conducted by the ATLAS and CMS collaborations independently confirmed the existence of a new boson consistent with the properties of the Higgs boson [16][17]. This discovery represented the final piece of the puzzle, solidifying the Standard Model's depiction of the subatomic world.

However, despite the monumental achievement in the discovery of the Higgs boson, the Standard Model is not the ultimate answer to every question in particle physics. Several

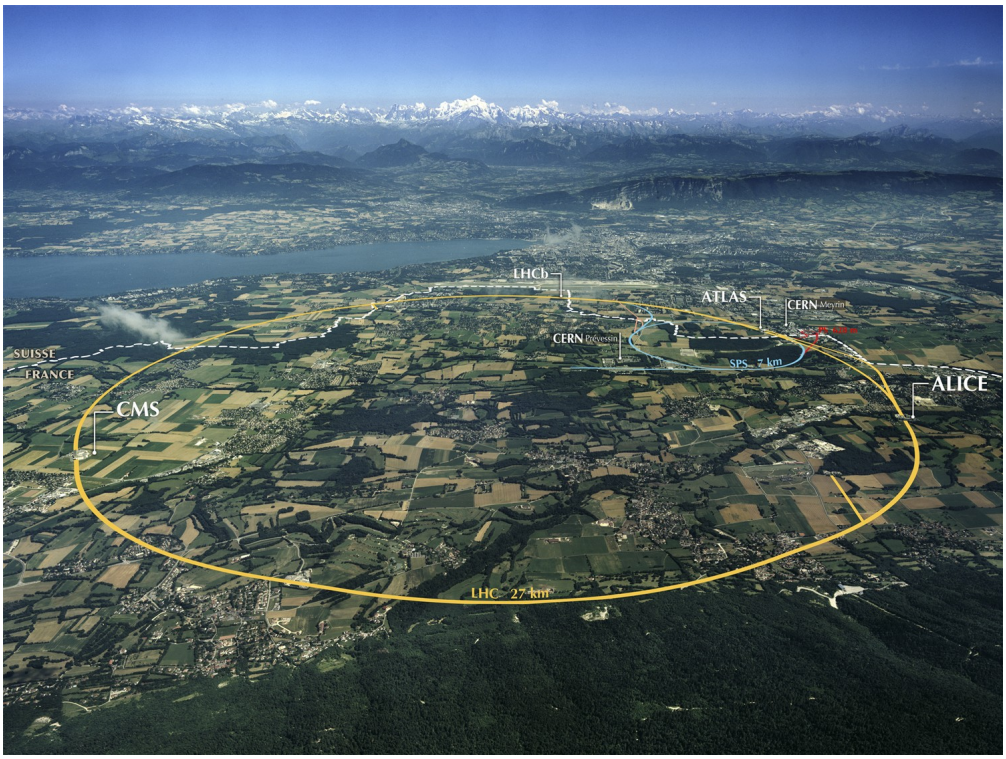

Figure 1.2: Aerial view of the LHC and its associated experiments [2]. The LHC is the world's most powerful particle accelerator in operation, capable of accelerating bunches of protons along a 27 km ring to an energy of 6.5 TeV each. At this energy, protons travel around the ring 11245 times per second. Each bunch contains roughly 100 billion protons. Bunches travelling in opposite directions around the ring are intentionally collided at experiment locations, resulting in approximately 1 billion proton-proton collisions per second [3].

phenomena remain unexplained, including the nature of dark matter, the reasons behind the masses of neutrinos, and the noticeable absence of gravity from the model.

Dark matter, for instance, remains one of the most enigmatic subjects in both particle physics and cosmology. While it doesn't emit, absorb, or reflect light, its presence is inferred through its gravitational effects on visible matter and the cosmic microwave background radiation. Observations from astrophysics, such as the rotation curves of galaxies and the lensing of light around massive objects, provide robust evidence for the existence of dark matter. Yet, its particle-based nature remains a mystery. Various theoretical particles, such as Weakly Interacting Massive Particles (WIMPs) and axions, have been proposed as candidates, but so far, no direct detection in particle experiments has been achieved.

Neutrinos, too, present a conundrum. While they were originally thought to be massless in the early formulations of the Standard Model, experiments like those observing solar neutrinos and atmospheric neutrinos have demonstrated that neutrinos oscillate between different flavors, implying that they possess non-zero masses. The exact mechanism and scale of neutrino masses, however, remain an active area of investigation, with neutrinoless double-beta decay experiments and cosmological observations offering potential insights.

Moreover, the Standard Model does not incorporate gravity, the most familiar force to us. While the other three forces have quantum descriptions in the model, gravity, described by General Relativity, stands apart. Unifying gravity with the other forces in a consistent quantum theory is one of the holy grails of theoretical physics.

Exploring higher energy and intensity scales might offer solutions or at least clues to these mysteries. Just as the discovery of the Higgs boson required accelerators to reach unprecedented energy levels, new phenomena may lie hidden, awaiting discovery at even greater energies. The High-Luminosity Large Hadron Collider (HL-LHC), an upgrade of the LHC, is set to delve into these higher energies and intensities, potentially shedding light on the aforementioned challenges. By colliding particles at such scales, it is hoped that signs of new particles or interactions might emerge, offering clues to the universe's deeper workings and guiding the next steps in our pursuit of knowledge.

1.2 Detector

Particle physics has always been intrinsically linked to the evolution of detection techniques. Historically, the progress in understanding the universe's smallest constituents has gone hand in hand with the development of increasingly sophisticated detectors. Early experiments in the field relied on cloud chambers and bubble chambers to observe charged particle tracks. As our knowledge grew, so did the necessity for more advanced apparatuses. Detectors allow physicists to test the predictions of theoretical models by directly observing and measuring particle interactions.

Controllable experiments, as are accomplished by pairing particle accelerators with particle detectors, provide a useful complement to more passive approaches, such as astrophysics experiments, which typically observe the remnants of high-energy cosmic rays as they interact with Earth's atmosphere or other experimental apparatuses. While these experiments provide invaluable insights into cosmic events and high-energy interactions in the universe, the unpredictable and uncontrollable nature of these cosmic events limits our ability to make precise measurements and study specific phenomena in detail. In contrast, particle physics experiments produce and study particle interactions in a more controlled environment (although the results of individual collisions are highly stochastic) and in larger volume. This allows for detailed studies of particular interactions and the systematic exploration of theoretical predictions.

At their core, detectors are motivated by a fundamental need: to determine the plausibility of physical models based on recorded events, and to make precise measurements related to those models. When high-energy particles, such as protons, are accelerated and collided, they can produce new particles not typically present or observable under normal conditions. The goal of the detector is to accurately record the properties of these final state particles. By cataloging a large number of such high-energy collisions, rare and novel phenomena can be observed, furthering our grasp of particle physics.

The basic setup for many of these experiments is relatively consistent. Particles, like

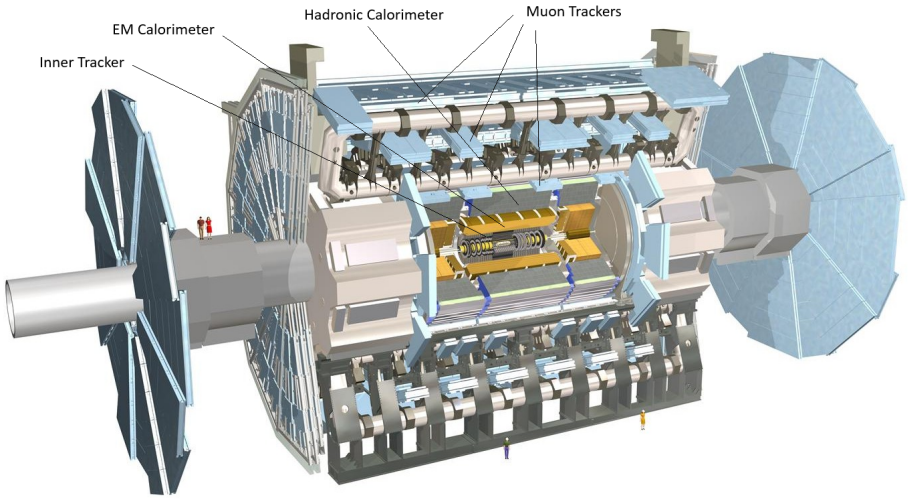

Figure 1.3: Simple illustration of key components of ATLAS detector [4]. Other detectors generally follow a similar design.

protons in the case of the LHC, are accelerated to near the speed of light and brought into collision. These collisions may produce a slew of new particles ('final state particles'), which subsequently travel through the detector. As these particles pass through various components of the detector, their interactions produce signals that are converted into data. This data, post-processing, offers information about the properties of the particles involved in the collision.

Understanding the properties of the particles resulting from these collisions is foundational to deciphering the underlying physical processes that took place. By analyzing the energy, momentum, charge, and type of the final state particles, physicists can infer the interactions that led to their creation. These observed interactions can be compared against predictions from existing theories, such as the Standard Model. Any discrepancies between observation and prediction might indicate new, previously unaccounted-for phenomena. Furthermore, the detailed study of these collisions can also help in refining current models or in setting limits on parameters within those models. For Beyond the Standard Model (BSM)

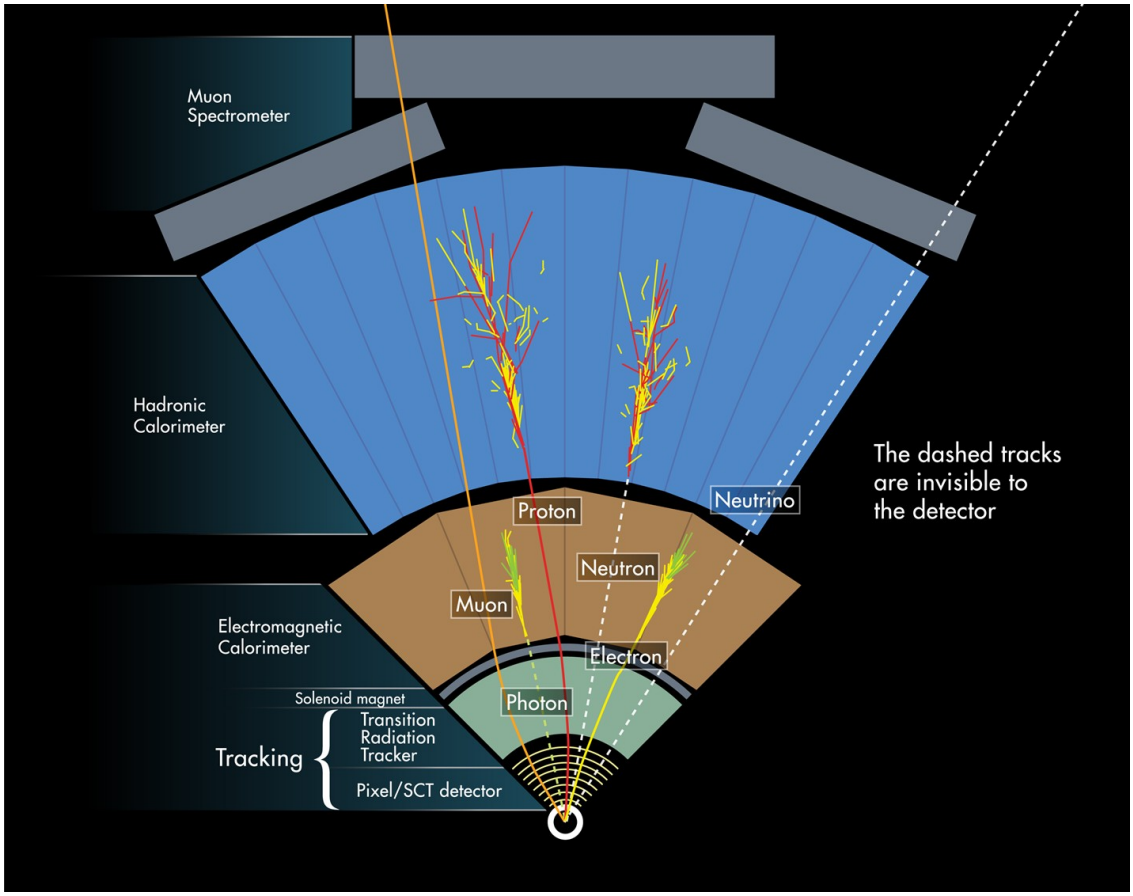

Figure 1.4: Cross-sectional representation of the ATLAS detector [5]. Illustrates the regions of the detector that contribute to measuring the properties of common particle types.

theory, this is particularly crucial. If BSM particles exist, they might be produced directly in these collisions or influence the interactions in subtle ways. By examining the particle properties and the ways they interact, we might find clues that lead to groundbreaking advancements, potentially reshaping our understanding of the universe's fundamental workings.

Typically, close to the interaction point–the point at which, e.g., the protons are collided–we install a tracking detector or 'inner tracker'. This device detects charged particles as they pass through, allowing for the reconstruction of their paths or "tracks." By applying a known external magnetic field, the curvature of these tracks can be used to determine the particle's

momentum, which can be precisely determined, especially for lower momentum particles– high momentum particles have a smaller curvature in their tracks, which increases the relative uncertainty. To do its job well, the tracker must provide fine spatial resolution to accurately trace particle paths and minimize error.

Further out, calorimeters come into play. These devices measure the energy of particles by absorbing them. Calorimeters can be further divided into electromagnetic calorimeters, which primarily measure electrons and photons, and hadronic calorimeters, designed to measure the energy of hadrons. When a particle enters a calorimeter, it initiates a cascade of secondary particles and interactions, which produces a measurable signal proportional to the incoming particle's energy.

1.2.1 Tracker

Charged particle tracking is indispensable in High Energy Physics (HEP) for many applications, including particle identification and kinematics, vertex finding, lepton reconstruction, and flavor jet tagging. Fundamental to particle tracking is the pattern recognition algorithm. This algorithm is tasked with associating a list of 2D or 3D position measurements, termed hits or spacepoints, from a tracking detector to a set of particle track candidates, commonly known as tracks. Conceptually, a track is a sequence of spacepoints linked by the pattern recognition to a specific charged particle.

The physical basis of trackers lies in their ability to detect the passage of charged particles through specific materials. When a charged particle traverses a medium, it ionizes the atoms in its path, leaving behind a trail of ionized electron-hole pairs. This ionization trail acts as the signature of the particle's path and forms the primary mechanism through which trackers operate.

Trackers are typically constructed from materials with three key properties: a) high sensitivity to ionizing radiation, b) rapid response times to quickly detect passing particles, and c) stability under the intense radiation environment close to the collision point. Silicon is the material of choice for many modern trackers due to its excellent charge collection

properties and its ability to be fabricated into fine-pitch devices, allowing for precise tracking. In some trackers, thin layers of silicon are assembled in a pixel or strip configuration, enabling the precise measurement of a particle's position in two dimensions.

The resultant 'hits' or 'spacepoints' are essentially the localized regions in the tracker where a particle has passed and caused ionization. The exact position of these hits is determined by the readout electronics associated with the tracker, which converts the ionization signals into digital data. The spatial resolution of these hits is crucial as it directly impacts the precision with which the particle's path, and hence its momentum, can be reconstructed.

A typical HEP offline tracking algorithm has four stages: spacepoint formation, track seeding, track following, and track fitting. The spacepoint formation stage amalgamates the raw data of the detector readout cells into clusters. From these clusters, the spacepoint 3D coordinates and their respective uncertainties are deduced. This procedure is indispensable as the subsequent stages depend on accurate spacepoint information.

The second phase, known as track seeding, is oriented towards establishing the initial trajectory of a charged particle. It combines the spacepoints into doublet or triplet seeds. Each seed provides valuable information about the particle's pathway: its direction, point of origin, and, in some cases, its curvature. Uncertainties associated with these values are also gleaned from the seeds. The seeds essentially form the basis for constructing an uninterrupted track, capturing the initial parameters that are later extended and refined by the subsequent stages.

Track following, the third phase, builds upon the foundational seeds. It extrapolates the trajectory defined by the seeds and seeks additional spacepoints that align with this extrapolated path. This is crucial as high-energy collisions in particle accelerators can result in a myriad of particles. These particles, in their high-speed journey, leave behind a trail of spacepoints in the tracking detector. To map the exact pathway of a particle, it becomes imperative to associate as many spacepoints as possible with its trajectory. This stage ensures that tracks are extended to their full length, incorporating all the relevant spacepoints they intersect.

Finally, the process culminates with track fitting. Sometimes combined with track following, this phase fits a trajectory through the track spacepoints. The fitting procedure is often achieved using advanced mathematical techniques, optimizing the fit based on certain criteria, like the least squares method. This stage serves a dual purpose. Firstly, it evaluates the quality of the track, ensuring it's consistent with the expected trajectory of a particle. Secondly, it extracts the particle's physical and kinematic properties. Vital attributes such as charge, momentum, point of origin, and others are determined at this stage.

It is worth noting the importance of efficiency in these algorithms. Each stage must possess high efficiency, ideally identifying more than 90% of the charged particles within a given fiducial volume. The rationale behind this stringent requirement is clear: to ensure unbiased physics results. The fidelity of the entire tracking algorithm is contingent on the precise and comprehensive reconstruction of particle trajectories. Incomplete or inaccurate tracking can skew results, thereby obscuring potential discoveries or misrepresenting known phenomena. Consequently, the continual refinement and validation of these tracking algorithms are paramount.

Recognizing these challenges, the particle physics community has turned its gaze to more innovative methodologies, particularly the realm of machine learning. Machine learning, with its ability to discern patterns from vast datasets, offers a promising avenue for enhancing tracking precision and efficiency. Among the myriad of techniques within machine learning, deep learning has emerged as a particularly potent tool. Deep learning models possess the capability to process low-level information, such as raw detector hits, and learn intricate features directly from the data without extensive handcrafted feature engineering. This capability dovetails well with the requirements of tracking, where low-level detector signals could encode the subtleties of particle interactions.

The integration of deep learning into tracking systems is not merely a theoretical proposition. Numerous experiments are actively exploring and implementing these techniques, revealing substantial improvements in both speed and accuracy. The adaptability of deep learning, coupled with its innate prowess in handling vast data streams, makes it an indis-

pensable ally in the quest to unravel the mysteries of the universe at the very frontiers of energy and luminosity.

1.2.2 Calorimeters

In addition to applying ML for tracking, recent efforts have been made to apply ML algorithms to calorimeter clustering. Calorimeters are designed to capture and measure the energy of incident particles, and when a particle interacts with the calorimeter, it typically creates a "shower" as it deposits energy, resulting in measurable energy in many sensitive elements of the calorimeters ("cells"). Clustering algorithms are designed to link multiple cells together, such that the sum of the cells' energy measurements approximates the energy deposited by the original particle. Traditionally, these are domain algorithms that perform loops over calorimeter cells, considering adjacency and energy patterns, potentially involving multiple steps. In the HL-LHC era, it is unclear if these traditional algorithms can achieve adequate physics performance while satisfying computing constraints. Because of this, recent efforts have been made to create ML-based clustering algorithms that can take advantage of modern hardware acceleration. For example, a graph neural network for clustering that also comprises a noise filter has demonstrated good performance for the CMS upgrade High Granularity Calorimeter (HGCAL) [18], which has much finer spatial resolution than the current CMS calorimeters. ATLAS has also developed a GPU-based porting of topological clustering for calorimeters, which can execute clustering about 4 times faster than CPU-based clustering and could be deployed in their online high-level trigger [19].

Full data processing is typically performed about once per year, occupying hundreds of thousands of CPU cores for several days. Per-event latency could be as high as about 3 seconds for HL-LHC events, which could be dramatically reduced by running tracking on GPUs. Similarly, the CPU-based component of the trigger will have to process over 500,000 events per second. Executing this component of the trigger takes on the order of 100,000 CPU threads, so each thread should process an event in roughly 200 ms. The algorithms run in the trigger step are reduced in complexity relative to offline processing in order to

improve latency. Adding computational power through the use of GPUs and ML algorithms would improve the overall complexity of the trigger system allowing for algorithms that more closely replicate the offline computing system.

1.3 Deep Learning

Deep learning (DL) is a rapidly developing subfield of machine learning (ML) and artificial intelligence (AI) that has given rise to physics-relevant techniques such as classification, tagging, noise reduction, event reconstruction, event simulation, and anomaly detection. There are two aspects of particle physics that make it unique, or at least highly atypical, as compared to other fields where machine learning is applied. Firstly, its governance by quantum mechanics introduces complexities and probabilistic behaviors not found in classical systems, impacting the type of data and the nature of predictions required from deep learning models. Secondly, particle physics benefits from advanced simulation tools capable of generating highly accurate synthetic data, essential for training DL models.

The LHC data is characterized by its high dimensionality and large volume, which traditional high-energy physics (HEP) approaches tried to manage by reducing the data's dimensionality through multiple processing stages. These stages were applied to both individual collision events and collections of events. In contrast, the integration of deep learning allows for direct analysis of this complex data, leveraging the field's unique aspects. This direct approach helps in more effectively capturing the subtleties and nuances inherent in low-level data, illustrating how HEP tasks can be efficiently reformulated as machine learning problems.

Ideally, a learning algorithm would find the function that optimizes L over the space of the observed data X to a low dimensional space of a desired target label Y , which optimizes some metric of our choosing. An essential goal in machine learning is generalization—the ability of the model to perform well on data which was not used in training. Failure in this task is called overtraining. There are a vast array of techniques to avoid overtraining (or overfitting) that can all be considered forms of regularization. Regularization techniques like

dropout were key to advances in image recognition with deep learning.

Initially, the term deep neural networks referred to neural networks with many hidden layers, and it was used to differentiate such networks from shallow neural networks, which had only one hidden layer. For many years, it was argued that using a shallow network was not a restriction, because of the theoretical analysis that demonstrated that any function can be approximated by a shallow network. However, an effective shallow network may require an enormous number of nodes in the hidden layer, and in practice, shallow neural networks often failed to discover useful functions from high-dimensional data sets. More generally, deep learning can refer to a broad class of machine learning methods emphasizing hierarchical representations of the data and modular, differentiable components. Not only do these deep networks have more expressive capacity, but also the layers can be interpreted as building up a hierarchical representation of the data. In natural images, for example, the first layers learn low-level features like edges and corners, the middle layers learn midlevel features like eyes, and the final layers learn high-level features like faces. The processes that produce particle physics data naturally lead to compositionality and hierarchical structured data. For instance, a typical event at the LHC is composed of jets, jets are composed of hadrons, hadrons lead to tracks and calorimeter clusters, tracks are composed of hits, and calorimeter clusters are composed of calorimeter cells. The analogy also extends to higher levels with groups of particles forming resonances in a cascade decay. For these reasons, one might anticipate deep learning to be particularly effective at the LHC.

1.3.1 Convolutional Neural Network

Convolutional Neural Networks (CNNs) have revolutionized the field of machine learning, particularly in the realms of image and video recognition, natural language processing, and other areas requiring the analysis of large data sets. At their core, CNNs are a class of deep neural networks, most commonly applied to analyzing visual imagery. They employ a mathematical operation called convolution, which is a specialized kind of linear operation. CNNs are known for their ability to automatically and adaptively learn spatial hierarchies

of features from input images or video.

The primary advantage of CNNs lies in their ability to process data in its raw form, particularly images. This efficiency stems from two main characteristics. Firstly, the weights of the convolution operations in CNNs remain fixed as they move across different locations in the image. This approach, known as weight sharing, assumes equivariance with respect to translation, meaning that if the input image is shifted, the features detected by the convolution will also shift in the same way. Secondly, this weight sharing mechanism significantly reduces the number of trainable parameters within the network, facilitating the construction of deeper, more complex networks without a corresponding explosion in computational requirements. CNNs are thus highly efficient for tasks like image classification and object detection. They achieve this efficiency through a layered structure of neuron collections, each processing different portions of the input image, referred to as receptive fields. These layers enable CNNs to capture and exploit the spatial and temporal dependencies present in image data, a crucial advantage when analyzing visual information.

In the context of particle physics, particularly in the study of calorimetry, CNNs have found significant applications. Calorimetry in particle physics involves measuring the energy of particles, where detecting and analyzing the spatial distribution of energy deposits can be crucial. The ability of CNNs to efficiently process and analyze complex spatial data makes them highly suitable for this task. For instance, in high-energy physics experiments, such as those conducted at the Large Hadron Collider (LHC), CNNs can be used to analyze the data from calorimeters to identify and classify different types of particle collisions.

However, a unique challenge arises in the context of particle physics data, which is often globally sparse yet locally dense. This means that while the overall data set has a small fraction of meaningful values (sparse), the regions with signals are densely sampled. This characteristic poses a significant challenge for standard CNNs, as the sparsity of the data can lead to inefficiencies and difficulties in learning relevant features.

This challenge necessitates the discussion of specialized techniques, such as sparse convolutions, which will be the focus of Chapter 3. Sparse convolutions are a variation of the

traditional convolution operation in CNNs, designed to handle data that is sparse in nature effectively. They adapt the convolution process to focus on the non-zero values in the data, thereby improving efficiency and performance in scenarios where data sparsity is a significant concern, as is often the case in particle physics applications.

1.3.2 Graphical Neural Network

Graphs, as a foundational mathematical and data representation structure, have been used for centuries to elucidate relationships between distinct entities. In their basic form, graphs consist of nodes (representing objects) and edges (representing pairwise relationships between these objects). The appeal of graphs, particularly in the world of data representation and analysis, stems from their innate ability to capture intricate relationships and dependencies among objects. This ability ensures that the subtleties and nuances inherent to complex data sets are not lost in translation, allowing for a more accurate and in-depth analysis.

Graph data structures are common to many abstract problems in computer science, such that algorithms operating on them have broad applicability. Notable applications include social networks, computer network analysis, natural language processing, circuit design, transportation and logistics, and biological and chemical modeling such as protein folding.

Of course, another realm where the utility of graph structures has become evident is particle physics. Particle physics experiments generate a myriad of data, often with diverse and variable structures. Traditional representations, such as vector or grid-like structures, tend to fall short in fully encapsulating the richness of this data. For instance, one might need to zero-pad these structures or lose vital information to make them fit into a predetermined size. On the other hand, graphs offer flexibility, accommodating data of variable sizes without the need for such modifications. This ensures that the raw essence of the data, and its interconnected relationships, remain preserved.

Furthermore, particle detectors often produce sparse and heterogeneous data. Translating this kind of data into image-based representations can be a challenging endeavor–although methods exist, which will be discussed next–often necessitating the projection of the data

into a form that might not be its most natural or informative state. Here again, the flexible nature of graphs proves beneficial. With graphs, there is no necessity to impose an artificial ordering scheme, unlike sequence-based representations. This translates to a representation that is more true to the underlying physical phenomena, be it energy deposits in a detector, individual physics objects such as tracks or missing energy, or even nuanced data like individual or groups of particles. Graphs, therefore, can capture the heterogeneous nature of particle-physics data, providing a holistic view that is both detailed and cohesive.

In recent years, advances in computational methodologies have led to *graph neural networks (GNNs)*. GNNs are, as the name implies, neural networks designed to operate on graph-structured datasets. The most common form follows a "graph-in, graph-out" approach, meaning that the underlying structure of the graph is unchanged by the neural network; instead, the features of the nodes or edges are transformed."Dynamic" approaches that adjust the graph structure, often by pruning edges, have also been created, but are not relevant to any of the studies in this thesis, and so will not be discussed further.

Central to GNN operations is precisely *how* the node/edge features should be transformed. The most common method is the pairwise message-passing paradigm. This concept can be likened to nodes in a graph engaging in a conversation. During each iteration, termed a 'layer' in GNNs, each node amasses information from its adjacent nodes. Typically, this involves aggregating features from neighbors, possibly undergoing transformations along the way, and subsequently melding them with the node's inherent features to yield an updated profile. This process of local transformation, given a node and its neighbor's features, is similar to the operation of a convolutional network.

Basic Mathematics of Message Passing in GNNs

Let $G = (V, E)$ be a graph where V is the set of nodes and E is the set of edges. Each node v_i has an associated feature vector $\mathbf{x}_{v_i}$.

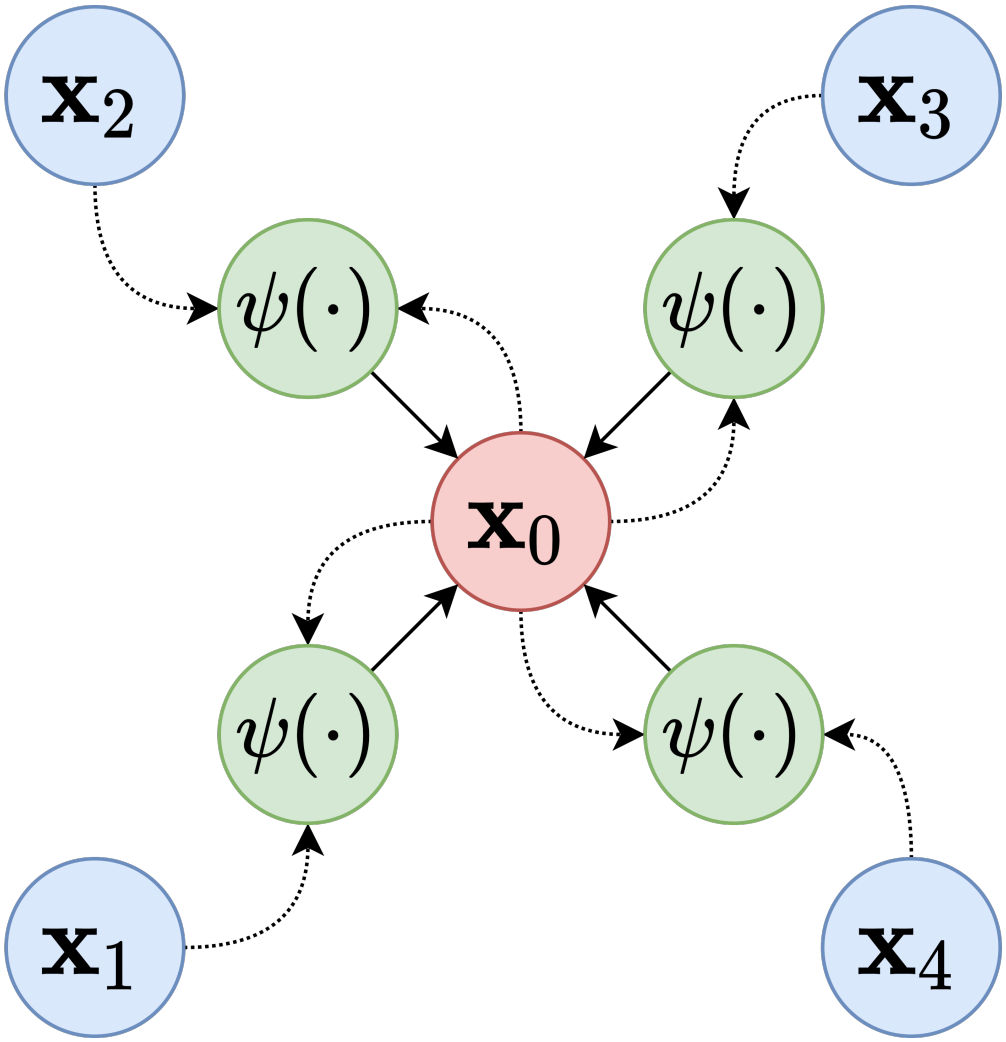

Figure 1.5: Representation of message passing for a single node x_0 with neighbors x_1, x_2, x_3, x_4 [6]. ϕ aggregates information from each pair of neighbors, which is then used to update the information stored at node x_0.

The message-passing mechanism can be formalized as:

$$\mathbf{m}_{v_i}^{(t)} = \sum_{v_j \in \mathcal{N}(v_i)} \psi(\mathbf{x}_{v_i}, \mathbf{x}_{v_j}, \mathbf{w}^{(t)}) \tag{1.1}$$

Where:

- $\mathcal{N}(v_i)$ represents the neighbors of node v_i (i.e., the set of nodes that are connected to v_i by a single edge.
- ψ is an aggregation function, parameterized by weights $\mathbf{w}^{(t)}$ for the t^{th} layer.
- $\mathbf{m}_{v_i}^{(t)}$ is the aggregated message for node v_i at layer t.

Typically, ψ includes a multi-layer perceptron (MLP) that operates on each $\mathbf{x}_{v_i}$ independently, learning an *embedding*–often to a space with a different dimensionality.

After the messages are computed, the node representations are updated as:

$$\mathbf{x}_{v_i}^{(t+1)} = g(\mathbf{x}_{v_i}^{(t)}, \mathbf{m}_{v_i}^{(t)}, \mathbf{u}^{(t)}) \tag{1.2}$$

Where g is an update function, and $\mathbf{u}^{(t)}$ are the weights for the t^{th} layer's update function. g is also typically an MLP. More complicated methods exist, including edges and global features, but the basic premise remains unchanged.

As with a convolutional neural network, one can reason about what is sometimes referred to as the *receptive field* of the graph network. After n layers, or *message passing iterations*, the current node feature vector $\mathbf{x}_{v_i}^{(n)}$ is a function of the input feature vectors $\mathbf{x}_{v_i}^{(0)}$ of each node that is at most n steps from it in the graph. As such, increasing n increases the amount of information from the graph that can be incorporated into the final output at each node. This is a typical hyperparameter and imposes an implicit constraint that input graphs must share some similarity in their connectedness for the network to perform well.

Utilizing GNNs in the realm of particle physics offers clear potential. With the complexities of particle interactions and the intricate datasets generated by particle experiments,

traditional neural networks often struggle to harness the complete spectrum of information available. GNNs, on the other hand, can naturally assimilate the relational intricacies, providing deeper insights into particle behaviors and interactions.

However, the deployment of GNNs in particle physics is not without challenges. For instance, the scalability of GNNs to massive particle datasets, and the computational intensity of deeper GNNs with multiple layers of message-passing, are aspects that require attention. Additionally, while the 'graph-in, graph-out' approach preserves the intrinsic structure of data, it necessitates careful consideration of how the output graph is interpreted, especially in cases where the desired output might not naturally be represented as a graph.

Furthermore, the robustness and generalization capabilities of GNNs, especially when applied to diverse particle datasets with varying characteristics, is a topic of active research. Variants of GNNs, such as those incorporating attention mechanisms, have been proposed to address some of these challenges, offering adaptive weighting of node relationships based on their significance.

As the field of particle physics moves towards more complex experiments, generating larger and more intricate datasets, the importance of suitable data representation and analysis tools will rise. GNNs, with their inherent ability to capture relationships and dependencies, look to be a useful tool in this journey.

1.4 Thesis Structure

As discussed in the above introduction, deep learning techniques will prove critical in the next generation of particle physics experiments, which could be crucial to understanding outstanding problems in our fundamental understanding of physics, such as dark matter. This thesis presents a comprehensive exploration of deep learning applications in particle physics, with a particular focus on tracking and calorimetry. It encapsulates a series of three studies that demonstrate the versatility and efficacy of deep learning techniques in addressing complex challenges in these domains.

This thesis is laid out as follows. Each study is split into a separate chapter. It is

aesthetically appealing to start at the center of the detector and move outwards in our discussion. As such, the second chapter covers the application of GNNs to tracking with the ExaTrkX project. The third chapter covers the use of CNNs to regress the energy of electrons/photons with DeepCalo. The fourth chapter covers the application of sparse CNNS to hadronic calorimetry clustering with SPVCNN. Finally, the fifth chapter is the conclusion.

Chapter two of the thesis presents the advancements of the Exa.TrkX project in applying geometric learning methods, specifically metric learning and graph neural networks, to particle tracking in High Energy Physics (HEP). The chapter focuses on the Exa.TrkX tracking pipeline, which organizes detector measurements into track candidates and subsequently filters them. Initially developed using the TrackML dataset, a simulation of a tracking detector inspired by the Large Hadron Collider (LHC), the pipeline's applicability extends to other detectors, including the DUNE Liquid Argon Time Projection Chamber and the CMS High-Granularity Calorimeter. This paper details the latest enhancements necessary for evaluating the Exa.TrkX pipeline's physics and computational performance on the comprehensive TrackML dataset. This evaluation is a preliminary step towards the pipeline's validation with actual data from the ATLAS and CMS experiments. The chapter highlights that the pipeline attains tracking efficiency and purity on par with existing production tracking algorithms. Importantly, it underscores the pipeline's potential for future HEP applications, particularly noting its substantial gains from GPU acceleration and the near-linear scalability of its computational demands in relation to the number of particles in an event.

Chapter three presents the integration of the DeepCalo deep learning model into the CERN Large Hadron Collider's particle energy reconstruction workflow. Addressing the high data generation rates and dynamic conditions of experiments, this chapter introduces the first automated workflow using hls4ml for implementing DeepCalo on Field-Programmable Gate Arrays (FPGAs). By optimizing dataflow in hls4ml and applying quantization-aware training for model compression, the chapter demonstrates an efficient, fully-on-chip implementation of large Convolutional Neural Network models while preserving model quality. The results

showcase a significant improvement in processing speed, with an inference latency of 1.34 milliseconds per 5 images on the Xilinx Alveo U50 FPGA, significantly faster than the existing GPU-based system. This achievement meets the stringent latency requirements of the Level-1 Trigger in particle physics experiments and marks a substantial advancement in computational performance compared to conventional CPU and GPU systems.

Chapter four explores the application of Sparse Point-Voxel Convolutional Neural Networks (SPVCNN) for clustering energy deposits from hadronic showers in the Compact Muon Solenoid detector at the Large Hadron Collider. Addressing the challenge posed by the stochastic nature of hadronic interactions, the chapter demonstrates how SPVCNN, trained with a modified object condensation loss, effectively groups cell deposits into clusters while filtering noise. The study reveals that SPVCNN performs comparably to traditional topological cluster-based methods in various scenarios, including those with pileup. Additionally, it highlights the potential of SPVCNN for GPU acceleration, underscoring its suitability for high-level trigger applications with millisecond latency, which is particularly relevant for the upcoming High-Luminosity Large Hadron Collider (HL-LHC), indicating a significant step forward in efficient and accurate calorimetry.

The three studies presented in this thesis, while focusing on distinct aspects of particle physics, are united by several key themes that underscore the transformative potential of deep learning in this field. First, each study highlights the power of deep learning in managing and interpreting the complex, high-dimensional, and often sparse datasets characteristic of high-energy physics experiments. The use of Graph Neural Networks (GNNs) for particle tracking, Convolutional Neural Networks (CNNs) for energy reconstruction, and Sparse Point-Voxel Convolutional Neural Networks (SPVCNN) for calorimetry clustering, all exemplify the ability of deep learning models to extract meaningful insights from intricate data structures.

Secondly, a common thread across these studies is the emphasis on computational efficiency, particularly through GPU acceleration. This is crucial in particle physics where the volume of data and the need for swift analysis necessitate high computational performance.

Each chapter demonstrates how deep learning models can be optimized for speed and scalability, a feature that is especially beneficial for real-time data processing in particle physics experiments.

Moreover, these studies highlight the adaptability of deep learning models to different detectors and experimental scenarios, from the LHC's CMS detector to the DUNE Liquid Argon Time Projection Chamber. This versatility showcases the potential of deep learning approaches to be applied across various experimental setups in particle physics, enhancing their utility and impact.

The potential of these models for real-time analysis in experimental settings is another significant aspect. The DeepCalo implementation on FPGAs, for example, meets the stringent latency requirements of Level-1 Triggers in particle physics experiments, demonstrating the practicality of deep learning models in real-world, high-speed experimental environments.

Additionally, the scalability of these models is evident in their applicability to the upcoming challenges of the High-Luminosity LHC. As particle physics experiments evolve to higher luminosities and complexities, the demand for sophisticated data analysis tools like those presented in these studies will increase, marking these deep learning models as vital for future advancements in the field.

In conclusion, these chapters collectively showcase the significant role that deep learning can play in the next generation of particle physics experiments. Their ability to handle complex data, combined with computational efficiency and adaptability, positions deep learning as a key driver for future research and experiments in particle physics. The insights and methodologies developed in these studies could be crucial in unraveling some of the outstanding mysteries in physics, such as the nature of dark matter, potentially leading to groundbreaking discoveries and a deeper understanding of the fundamental forces and particles that constitute our universe.

Chapter 2

PERFORMANCE OF A GEOMETRIC DEEP LEARNING PIPELINE FOR HL-LHC PARTICLE TRACKING

The Exa.TrkX project has applied geometric learning concepts such as metric learning and graph neural networks to HEP particle tracking. Exa.TrkX's tracking pipeline groups detector measurements to form track candidates and filters them. The pipeline, originally developed using the TrackML dataset (a simulation of an LHC-inspired tracking detector), has been demonstrated on other detectors, including DUNE Liquid Argon TPC and CMS High-Granularity Calorimeter. This chapter documents new developments needed to study the physics and computing performance of the Exa.TrkX pipeline on the full TrackML dataset, a first step towards validating the pipeline using ATLAS and CMS data. The pipeline achieves tracking efficiency and purity similar to production tracking algorithms. Crucially, for future HEP applications, the pipeline benefits significantly from GPU acceleration, and its computational requirements scale close to linearly with the number of particles in the event.

For this project, the author's personal contributions were mainly focused on studying the embedding and fixed-radius NN and resiliency to noise and misalignment.

2.1 Introduction

Charged particle tracking plays an essential role in High-Energy Physics (HEP), including particle identification and kinematics, vertex finding, lepton reconstruction, and flavor jet tagging. At the core of particle tracking there is a pattern recognition algorithm that must associate a list of 2D or 3D position measurements from a tracking detector (known as *hits* or *spacepoints* in literature) to a list of particle track candidates (or *tracks*. A *track* is defined as a list of spacepoints associated by pattern recognition with a charged particle).

The number of particle track candidates varies significantly from one experiment setup to another. For example, in a High-Luminosity LHC (HL-LHC) [20] collision *event*, due to the *pile-up* of multiple proton-proton collision per bunch crossing, there are typically 5,000 charged particles and 100,000 spacepoints, about 50 % of which are associated to particles of interest.

A typical HEP offline tracking algorithm [21, 22, 23] has four stages: spacepoint formation, track seeding, track following, and track fitting. The spacepoint formation stage combines the detector readout cell raw data in clusters from which the spacepoint 3D coordinates, and their uncertainties, are determined.

Track seeding combines spacepoints in *doublet* or *triplet seeds*. Each seed provides an initial track direction, origin, and possibly a curvature, with associated uncertainties.

Online tracking algorithms may use different pattern recognition algorithms[1] to create and filter track seeds and candidates, but share the same high efficiency requirements. Online application also have stringent computing requirements (e.g. latency $O(10)$ μs for LHC triggers).

The computational cost of current tracking algorithms grows worse than linearly with beam intensity and detector occupancy, as demonstrated in Figure 2.2. Given the order-of-magnitude increase for beam intensity at HL-LHC, charged particle pattern recognition algorithms might well limit the discovery potential of HL-LHC experiments.

Over the last two decades, tracking computational challenges arising from the increased number of combinations have been addressed by tightening fiducial regions for charged particles, developing highly optimized tracking algorithms [22, 23], and even optimizing the geometry of tracking detectors. These optimizations brought order-of-magnitude gains in tracking computational performance with limited impact on physics. While these efforts continue [28], it is unlikely that another order of magnitude can be gained through incremental optimization without impacting physics performance. Furthermore, given the com-

[1] including Hough transforms [24, 25] and cellular automata [26, 27]

putational complexity and iterative nature of current track following and filtering algorithms, it is challenging to run them efficiently on data parallel architectures like GPUs.

The TrackML challenge [7] jump-started the application of deep learning pattern recognition methods applied to HEP tracking. The HEP.TrkX pilot project [29] proposed the use of graph networks to filter track doublet and triplet seeds [30]. Building on that work, the Exa.TrkX project [31] has demonstrated the applicability of *Geometric Deep Learning* (GDL) methods [32] – specifically metric learning and Graph Neural Networks (GNN) – to particle tracking [33]. GDL is concerned with learning representations of data that have complex geometrical relationships and no natural ordering, like detector spacepoints. GDL models are computationally regular, naturally parallel and therefore well-suited to run on hardware accelerators.

This work describes new developments that enabled the first study of the computing and physics performance of the Exa.TrkX pipeline on the entire TrackML detector at HL-LHC design luminosity, a step towards the validation of the pipeline on ATLAS and CMS data.

2.1.1 Related Work

Early on, the Hep.TrkX pilot project attempted to assign and regress track parameters to single spacepoints using image processing models. Subsequent attempts to estimate track parameters using image processing and recurrent networks showed promising results [34] in a simplified environment. A similar realization of the method is reported in [35] where a model processing image from successive pixel detector layers is used to produce tracklets, seeds to classical pattern recognition. The method yields superior seeding efficiency for tracks within jets in dense environments. The concept of using LSTM [36] to supplement the Kalman Filter method for track following developed by HEP.TrkX [34, 37, 30] was later found in one of the promising solutions of the accuracy phase [38] of the TrackML challenge. The task of particle tracking was addressed with a hit-to-track assignment method using gated recurrent unit [39] (GRU), producing promising result in sparse environments [37]. This approach was constrained computationally due to the use of recurrent models.

Ref. [40] applies the track finding approach developed in Ref. [41] to the whole detector by exploiting a new data-driven graph construction method and large model support in Tensorflow [42]. Ref. [43] applies a similar GNN model to the task of particle-flow reconstruction. The model has a classification objective, followed by a partial regression of generator-level particle candidate kinematics. The method performs at least as well as a classical particle-flow algorithm in HL-LHC-like collision conditions. As part of the Exa.TrkX project, graph networks are used for LArTPC track reconstruction [44]. Ref. [45] explores the opportunity to implement Exa.TrkX-inspired graph networks on FPGAs. Starting from the input stage of the Exa.TrkX pipeline, Ref. [46] studies the impact of cluster shape information on track seeding performance. In Ref. [47], metric learning is used to improve the purity in spacepoints buckets formed using similarity hashing. With the advent of quantum computer of increasing size came the development of quantum machine learning techniques, also applied in particle physics [48]. In particular, inspired by the use of GNN for charged particle tracking of the Exa.TrkX team, quantum graph networks have been tested on the same problem [49].

2.2 Methodology

2.2.1 Input Data

This study is based on the TrackML dataset that uses a Montecarlo simulation of top quark pair production from proton-proton collisions at the HL-LHC. To simulate the effect of event pileup and produce realistic detector occupancy, a Poisson random number (with $\mu = 200$) of QCD "minimum bias" events are overlaid on top of the $t\bar{t}$ collisions.

The TrackML detector is a set of concentric cylindrical layers of pixelated sensors (the *barrel*) complemented by a set of circular disks (the *endcaps*) to ensure nearly 4π coverage in solid angle, as pictured in Figure 2.1. Figure 2.3 shows the spatial distribution of the spacepoints of a typical event. One notable feature of this dataset is the inclusion of "noise" spacepoints, added as a proxy for various low-momentum particle interactions and detector

effects which would otherwise require more expensive and detailed simulations.

2.2.2 The Geometric Deep Learning Pipeline

This work updates the methodology previously presented in Ref. [33] to a fully-learned pipeline, where both graph construction and graph classification are trained. This section describes the pipeline (represented schematically in Figure 2.4) used to obtain the results in § 2.3.

The pipeline currently used to reconstruct tracks from a pointcloud of spacepoints requires six discrete stages of processing and inference. These broadly consist of a preprocessing stage, three stages required to construct a spacepoint graph, and two stages required to classify the graph edges and partition them into track candidates. Each stage is trained independently (due to memory constraints) on the output of the previous stage's inference.

First, the dataset is processed into a format suitable for model training. This includes calculating directional information and summary statistics from the charge deposited in each spacepoint, i.e. the *cell features* in Figure 2.4. These values are appended to the cylindrical coordinates of each spacepoint to form an input feature vector to the pipeline. To apply a graph neural network to this set of data, it is necessary to arrange them into a graph. One can apply various geometric heuristics to define which spacepoints are likely to be connected by an edge (i.e. belong to the same track), but a useful technique is to train a model on the geometry of connected tracks. Thus, our second stage is to train an Embedding Network – a multi-layer perceptron (MLP) which embeds each spacepoint into an N-dimensional latent space. The graph is constructed by connecting neighboring spacepoints within a radius $r_{\text{embedding}}$, in the latent space. We train this embedding with a pairwise hinge loss, to encourage spacepoints that belong to the same track to be close in the embedded space, according to the Euclidean metric. This allows for a highly efficient edge construction, since we do not rely on any heuristics of the detector geometry that may lead to missed edges.

The edge selection at this stage is close to 100% efficient but $O(1)$% pure, with a graph size of $O(10^5)$ nodes and $O(10^7)$ edges (the purity-efficiency trade-off can be tuned with the

choice of $r_{\text{embedding}}$). Before running training or inference on the memory-intensive GNN, we filter these edges down with another MLP. The input to this third stage is the concatenated features on either side of each edge. That is, the Filter Network is a binary classifier applied to the set of edges. Constraining edge efficiency to remain high (above 96%) leads to much sparser graphs, of $O(10^6)$ edges.

The fourth stage of the pipeline is the training and inference of the graph neural network. The results presented in this work are predominantly obtained from the Interaction Network architecture, first proposed in Ref. [50]. This varietal of GNN includes hidden features on both nodes and edges, which are propagated around the graph (called "message passing") with consecutive concatenations along edges and aggregations of messages at receiving nodes. In the final layer of the network, a binary classification is obtained for each edge as true or fake, and trained on a cross-entropy loss.

The final stage of the TrackML pipeline involves task-specific post-processing. If our goal is track formation, we can place a threshold on the edge scores produced by the GNN and partition the graph into connected components. If our goal is track seeding, we can directly sample the classified edges for high likelihood combinations of connected triplets, or convert the entire graph to a *triplet graph* and train this on a second GNN to classify the triplets. A triplet graph is formed by taking all edges in the original (*doublet*) graph and assigning them as nodes in the new triplet graph. The nodes in this triplet graph are connected if they share a hit in the doublet graph. Applying a GNN to this structure produces highly pure sets of seeds as shown in Ref. [33].

Many of these techniques are common to other applications being explored in the Exa.TrkX collaboration. The pattern of nearest-neighbor graph-building and GNN edge classification has shown its potential for neutrino experiments [44] and CMS High Granularity Calorimeter [41]. Indeed, these applications build on the TrackML pipeline and extend it, for example by adding the particle type as an edge feature.

2.2.3 Feature Set

The input dataset includes both spatial coordinates and highly granular pixel cluster shape information. Graph construction (the second pipeline step in Figure 2.4, that includes learned embedded space model and edge filter model) appears to benefit significantly from the cluster shape information, approximately doubling the purity for a held fixed high efficiency. The summary cluster shape statistics include the number of channels and the total charge deposited, as well as local and global representations of the cluster as a high-level feature vector. Details about the calculation of this feature vector as well as a thorough exploration of the effect of cluster shape information on seeding performance are provided in Ref. [46]. Cluster shape information does not appear to improve the performance of the GNN, and in fact seems to degrade it. This suggests that the width of the GNN hidden layers is not great enough to capture the functional relationship of cluster information between nodes. Scaling to a width that properly explores this question would require more memory than available on the Nvidia A100 GPUs used for this study.

Depending on the final goal of the pipeline, further features can be included in the loss calculation in order to bias the model towards desired regions. For example, if our aim is to maximize the TrackML score (described in Ref. [7]) — a weighting function s_i that places more importance on a spacepoint i from a longer and higher p_{T} track, and in the first and last sets of detector layers – we can weight-up true edges by this function, normalized to have a mean of weight = 1. To measure the performance of models trained to this goal, we introduce a *weighted* purity measure. Weighted purity is defined as a function the TrackML weights w_{ij} and the truth $y_{ij} \in \{0, 1\}$ of each edge connecting spacepoint i and spacepoint j,

$$\text{Purity}_{\text{weighted}} = \frac{\sum_{ij} w_{ji} y_{ij}}{\sum_{ij} w_{ij}}, \tag{2.1}$$

$$w_{ij} = \begin{cases} \frac{1}{2}(s_i + s_j), & \text{if } y_{ij} = 1 \\ 1, & \text{if } y_{ij} = 0 \end{cases} \tag{2.2}$$

We see significant improvements in this metric when validating on the weighted model: the Embedding Network improves from a weighted purity of $1.7\% \pm 0.2\%$ to $2.0\% \pm 0.3\%$, while the Filter Network improves from a weighted purity of $8.4\% \pm 0.6\%$ to $11.7\% \pm 1.0\%$. Given this weighting, the model learns to prioritize higher p_{T} and longer tracks, while disregarding less informative tracks. Using this bias, we can achieve the same TrackML score with a constructed graph size reduced by approximately 25%. Using this technique to improve the TrackML score is an ongoing work.

2.2.4 Graph Construction

Having chosen a feature set, to train the learned embedding space we use a training paradigm commonly referred to as a Siamese Network [51], where a particular spacepoint - called the *source* - is run through an MLP, here 6 layers each with 512 hidden channels, hyperbolic tan activations, and layer normalization. The final layer of the MLP takes the features to an 8-dimensional latent space. A different, comparison spacepoint - called the *target* - is also run through this same Embedding Network, and the L2 norm distance d in the latent space between the source and target enters a comparative hinge loss

$$\mathcal{L}_{\text{hinge}} = \begin{cases} d^p, \text{ if } y_{ij} = 1 \\ \max(0, 1 - d^p), \text{ if } y_{ij} = 0 \end{cases} \tag{2.3}$$

where p is a hyperparameter that we choose to be 2.

If the source i and target j spacepoints share an edge in the event's truth graph [2], we

[2] one can also designate $y_{ij} = 1$ for source and target in the same track, rather than *immediate neighbors* in the track. This does lead to similar performance in later stages of the pipeline, but the more lax concept of truth leads to graphs around three times more dense than the strict track neighbor definition.

designate them as neighbours with $y_{ij} = 1$, otherwise they are designated $y_{ij} = 0$. In this way, the hinge loss draws together truth graph neighbors and repels non-neighbors.

Training performance of the Embedding Network is highly dependent on choice of source-target example pairs. In early epochs, it is enough to choose random pairs. However, at some point, many random pairs will contribute no gradient to the loss, as they will be separated by a distance greater than the margin. At that point, it is useful to implement hard negative mining. We run a GPU-optimised k-nearest-neighbor (KNN) algorithm [3] to mine examples around each source vector, within the hinge margin $d = 1$. The computational overhead of the KNN step is significantly offset by the examples mined which all contribute to the loss.

A similar technique is used in the Filter Network, where the vast majority of the edges produced from the graph construction in the embedded space are easy to classify as fake. This is already a highly imbalanced dataset, with around 98.5% of edges fake. Again, within several epochs, the Filter Network is able to classify many of these as fake, so we balance each batch with all true edges, the same number of hard negatives (i.e. negatives the filter is unsure of) and the same number of easy negatives (to maintain performance on these edges). The Filter Network is a MLP that takes the 24-feature concatenated edge features and feeds forward through 3 layers of 1024 hidden channels, to a binary cross-entropy loss function.

2.2.5 GNN Edge Classification

In choosing the best GNN architecture, memory usage remains a significant constraint. The Interaction Network (IN) [50] presented in these results does appear to marginally attain the best performance against Attention Graph Neural Networks (AGNN) [54, 30] – the other class of GNN considered for the pipeline. However, both of these networks require gradients to be retained in memory for every graph edge. Indeed, this anisotropic treatment of edges (i.e. a node is able to receive the messages of each of its neighbors in a non-uniform way) is what allows these two architectures to be so expressive. Depending on hardware availability,

[3] We use two high-performance libraries, FAISS [52] and Pytorch3D [53], depending on number of nearest neighbors k. Fastest performance is obtained with FAISS for $k > 35$, Pytorch3D for $k <= 35$.

we have found two solutions to the memory constraint. Access to next-generation Nvidia A100 GPUs allowed an IN to be trained with 8 steps of message passing, aggregating edge features at each node, and each node and concatenated edge features passing through two-layer MLPs of [128, 64] hidden features and ReLU activations. Choice of aggregation function should be permutation invariant. In this work, we take it to be a summation.

For lower-memory GPUs, such as the Nvidia V100, we attained similar performance training the AGNN architecture, with [64, 64, 64]-channel MLPs applied to each edge and node. Adding residuals [55] across the 8 message passing steps greatly improved performance in this case. To fit full-event training on a single V100, it was necessary to employ various techniques, such as mixed precision training and gradient checkpointing. The latter stores only the input of each layer, not the gradients. On the backward pass, gradients are re-calculated on the fly, allowing for a 4x reduction in memory usage for an 8-iteration GNN. Another technique explored is to split the events piecemeal and train on each piece as a standalone batch. There is a noticeable impact on performance due to messages being interrupted at the graph edges. In future work, we will present ongoing efforts to parallelise these graph pieces across multiple GPUs, retaining the high performance that full-event training allows.

2.3 Results

2.3.1 Tracking Performance of the TrackML pipeline

Tracking Efficiency and Purity

The performance of a tracking pipeline is mainly characterized by tracking efficiency and purity. For efficiency calculations, only charged particles that satisfy

$|\eta| < 4.0$ and $p_{\mathrm{T}} > 100$ MeV are considered. These *selected* particles, $N_{particles}$(selected), are hereafter referred to as *particles*.

The overall tracking efficiency, known as *physics efficiency* ϵ_{phys} (Eq. 2.4), is defined as the fraction of particles that are *matched* to at least one reconstructed track. A particle is

considered to be matched to a reconstructed track when 1) the majority of spacepoints in the reconstructed track belong to the same true track, and 2) the majority of spacepoints in the matched true particle track are found in the reconstructed track[4].

To measure the efficiency of the tracking pipeline itself, we also define the *technical efficiency* ϵ_{tech} (Eq. 2.5) as the fraction of *reconstructable* particles matching at least one reconstructed track. Reconstructable particles have a trajectory that leaves at least five spacepoints in the detector. Tracking purity (Eq. 2.6) is defined as the fraction of reconstructed tracks that match a selected particle[5].

$$\epsilon_{\text{phys}} = \frac{N_{particles}(\text{selected, matched})}{N_{particles}(\text{selected})} \tag{2.4}$$

$$\epsilon_{\text{tech}} = \frac{N_{particles}(\text{selected, reconstructable, matched})}{N_{particles}(\text{selected, reconstructable})} \tag{2.5}$$

$$\text{Purity} = \frac{N_{tracks}(\text{selected,matched})}{N_{tracks}(\text{selected})} \tag{2.6}$$

Averaged over 50 testing events from the TrackML dataset, the physics efficiency for particles with $p_{\text{T}} > 500$ MeV is $88.7 \pm 0.3\%$ and the technical efficiency is $97.6 \pm 0.3\%$. Without any fiducial p_{T} cut, the physics efficiency becomes $67.2 \pm 0.1\%$ and the technical efficiency $91.3 \pm 0.2\%$. The tracking purity is $58.3 \pm 0.6\%$. Using the TrackML challenge scoring system and all tracks in the event, we obtained a score of 0.877 ± 0.005 [6]. The errors quoted are statistical only.

Figure 2.5 shows the p_{T} distribution of particles as well as the tracking efficiency as a function of particle p_{T}. The physics efficiency for particles with p_{T} of [100, 300] MeV

[4]This nomenclature and the associated definitions broadly follow [56, 7].

[5]HEP tracking literature often quotes fake rate = 1 − purity

[6]We obtained a score of 0.914 ± 0.006 by training the pipeline with a dataset that includes noise hits, that we otherwise removed from our training dataset to facilitate the noise impact studies of section 4.1.2

is 43%, therefore, is not displayed in the plot. The physics efficiency for particles with $p_{\mathrm{T}} > 700$ MeV is above 88%. The technical efficiency is 82% for particles with p_{T} of [100, 300] MeV, and increases to above 97% for particles with $p_{\mathrm{T}} > 700$ MeV. Figure 2.5 also shows the η distribution of particles with $p_{\mathrm{T}} > 500$ MeV as well as the tracking efficiency as a function of the particle η. The physics efficiency is higher in the *barrel* region of the detector (volumes 8,13,17 in Figure 2.1), while the technical efficiency is almost flat across the η range. In Figure 2.5 the p_{T} and η of the matched truth particle were used, rather than the p_{T} and η of the reconstructed track. We leave a study of track quality and detector resolution effects for future work.

Systematic Studies

Before using a tracking algorithm in production, it is necessary to measure its sensitivity to systematic effects, including pile-up, noise and digitization errors, and uncertainties in the measurement of detector properties (alignment, rotation, magnetic field map, etc.).

Measuring precisely the impact of pile-up collisions on tracking performance is beyond the scope of this work, but we can estimate pile-up's impact on tracking performance by plotting efficiency and purity as a function of the number of spacepoints in the detector. Figure 2.6 shows that the effect of the increased detector occupancy is a smooth performance degradation O(%). In future work, we will study the origin of this degradation to achieve the stable performance of traditional algorithms [57].

The impact of noise spacepoints can be estimated using the TrackML dataset by studying the inference performance of the tracking pipeline, trained without any noise spacepoints, as a function of the fraction of noise spacepoints (up to a maximum of 20% of the total). Table 2.1 shows the technical tracking efficiency and purity for different noise levels. The efficiency decreases by $\simeq 1.6\%$ and the purity by $\simeq 5.4\%$ when 20% of noise spacepoints are presented. The loss of efficiency happens primarily for particles with $p_{\mathrm{T}} < 500$ MeV (Figure 2.7).

Detector misalignment effects are approximated by shifting by up to 1 mm the x-axis of

Table 2.1: Technical efficiency and purity for different noise fractions $(N_{\text{spp}}^{\text{noise}}/N_{\text{spp}}) * 100\%$

Noise	ϵ_{tech}	Purity
0	91.5	59.3
4%	91.5	59.3
8%	91.1	58.0
12%	90.9	56.8
16%	92.2	54.8
20%	89.9	53.9

all spacepoints in the inner-most TrackML barrel detector layer or the four innermost layers (volume 8 in Figure 2.1). In both cases, the impact on the tracking efficiency is less than 0.1%. However, studying in depth misalignments, and other detector effects, requires access to experiment detailed detector simulation data. We leave these studies as future work to be performed in collaboration with each experiment.

2.3.2 Inference performance on CPU and GPU

It is crucial to characterize the computational cost of the end-to-end learned tracking algorithm. We rely on the PyTorch and TensorFlow libraries to optimize our inference pipeline on CPU and GPU. The execution time for the inference pipeline has been measured on two hardware platforms: Nvidia V100 GPUs with 16 GB on-board memory, and Intel Xeon 6148s (Skylake) CPUs with 40 cores and 192 GB memory per node. The inputs to the filtering step do not fit into the GPU memory. Therefore, edge filtering for one event is executed in mini-batches with a fixed batch size of 800k edges. Typically, the inputs to the filtering from one event are split into seven batches, leading to additional computational cost for moving data from host to GPU. The peak GPU memory consumption is about 15.7 GB as obtained from the Nvidia profiling tool.

Averaging over 500 events, it takes 2.2 ± 0.3 wall-clock seconds per event (as measured by measured by the python module *time*) to run the inference pipeline on the GPU and 202 ± 35 seconds to run it on a single CPU core. This total execution time includes every step of the calculation, and in particular the time needed to move data from host to GPU. Table 2.2 breaks down the wall-clock time for the most significant steps of the pipeline. The results show how the graph creation and filtering steps are the biggest targets for further optimization in order to surpass traditional algorithms in terms of inference time [58].

Table 2.2: Average inference time for synchronous execution of the TrackML pipeline benchmarked on CPUs and GPUs. For these step-by-step measurements, we force the pipeline to execute serially by calling `torch.cuda.synchronize` after each step. The total inference time comprises all the steps including ones not listed in the table.

	Wall time [s] on Xeon 6148s single core	Wall time [s] on Nvidia V100 synchronous
Data Loading	0.0049 ± 0.0153	0.0023 ± 0.0003
Embedding	3.02 ± 0.39	0.024 ± 0.003
Build Edge	66 ± 13	0.76 ± 0.10
Filtering	99 ± 19	1.57 ± 0.34
GNN	27 ± 2	0.45 ± 0.06
Labeling	3.23 ± 0.34	0.08 ± 0.01
Total (sync)	202 ± 35	3.3 ± 0.5

In addition, Figure 2.8 shows how the total inference time depends almost linearly on the number of spacepoints in the event for both CPUs and GPUs. The step-like dispersion in the GPU case is due to the splitting of the inputs to the filtering step into mini-baches. A step-like jump indicates one more mini-batch is added.

Many optimizations were introduced to the pipeline in order to achieve these GPU timings, which before optimization took over 20 seconds per event. These improvements include porting all data processing to the GPU-accelerated CuPy library [59], writing custom sparse operations for graph processing (e.g. doublet-to-triplet conversion [60], graph intersection methods), using FAISS [52] for large-k NN graph construction, and performing track labelling with CuGraph's connected component algorithm on GPU [61][7]. These improvements are specific to the inference stage. No CPU-specific optimization was performed in this work.

2.4 Conclusion

This works shows how a tracking pipeline based on geometric deep learning can achieve state-of-the-art computing performance that scales linearly with the number of spacepoints, showing great promise for the next generation of HEP experiments. The inference pipeline has been optimized on GPU systems, on the assumption that the next generation of HEP experiments will have widespread access to accelerators either locally in heterogeneous systems [43, 63] or remotely [64, 65].

Within the simplifying assumptions of the TrackML dataset, we have shown how the Exa.TrkX pipeline could meet the tracking performance requirements of current collider experiments. Preliminary studies suggest that this performance should be robust against systematic effects like detector noise, misalignment, and pile-up.

Much remains to be done to validate these promising results. To this end, the Exa.TrkX project is collaborating with physicists from ATLAS [66], CMS [67], DUNE [68], ICARUS [69], and MuonE [70].

The goal is to adapt the Exa.TrkX pipeline to each experiment's needs and simulated datasets, measure its performance and robustness against systematic effects according to the experiment metrics. For example, it is crucial for HL-LHC experiments to study the

[7] on CPU, track labeling uses the DBSCAN algorithm [62].

performance of tracking algorithms in dense environments, like high-p_{T} jets. Given the interest in long-lived particle observation at the HL-LHC, it will also be important to study the performance of the Exa.TrkX pipeline for tracks coming from a displaced vertex [8].

On the computational side, there are several optimization opportunities to explore systematically, including mixed precision training, multi-GPU training and inference with graph data parallelisation (that is, one event spread across multiple GPUs) [71]; locality sensitive hashing to speed-up KNN/graph construction stage [72], model quantization, operator fusion and other improvements with TensorRT [73], clustering of final node embeddings rather than hard connected components method with GravNet-style architectures [74].

The distributed training results presented in this work are promising but still preliminary. To fully exploit the capabilities of upcoming HPC systems and to further reduce training time while potentially pushing further on model size, it will be beneficial to perform further studies on large scale training of GNNs for track reconstruction. Given the size of the input graphs, this problem may be amenable to training techniques which parallelise the processing of input graphs across multiple GPUs in training.

Finally, it will be interesting to measure the computing performance of (parts of) the Exa.TrkX pipeline on domain-specific accelerators like Google TPU [75] and GraphCore IPU [76], comparing power consumption, latency and throughput with "traditional" GPUs.

[8] it may be worth noticing that in LArTPC applications [44] all tracks come from a displaced vertex.

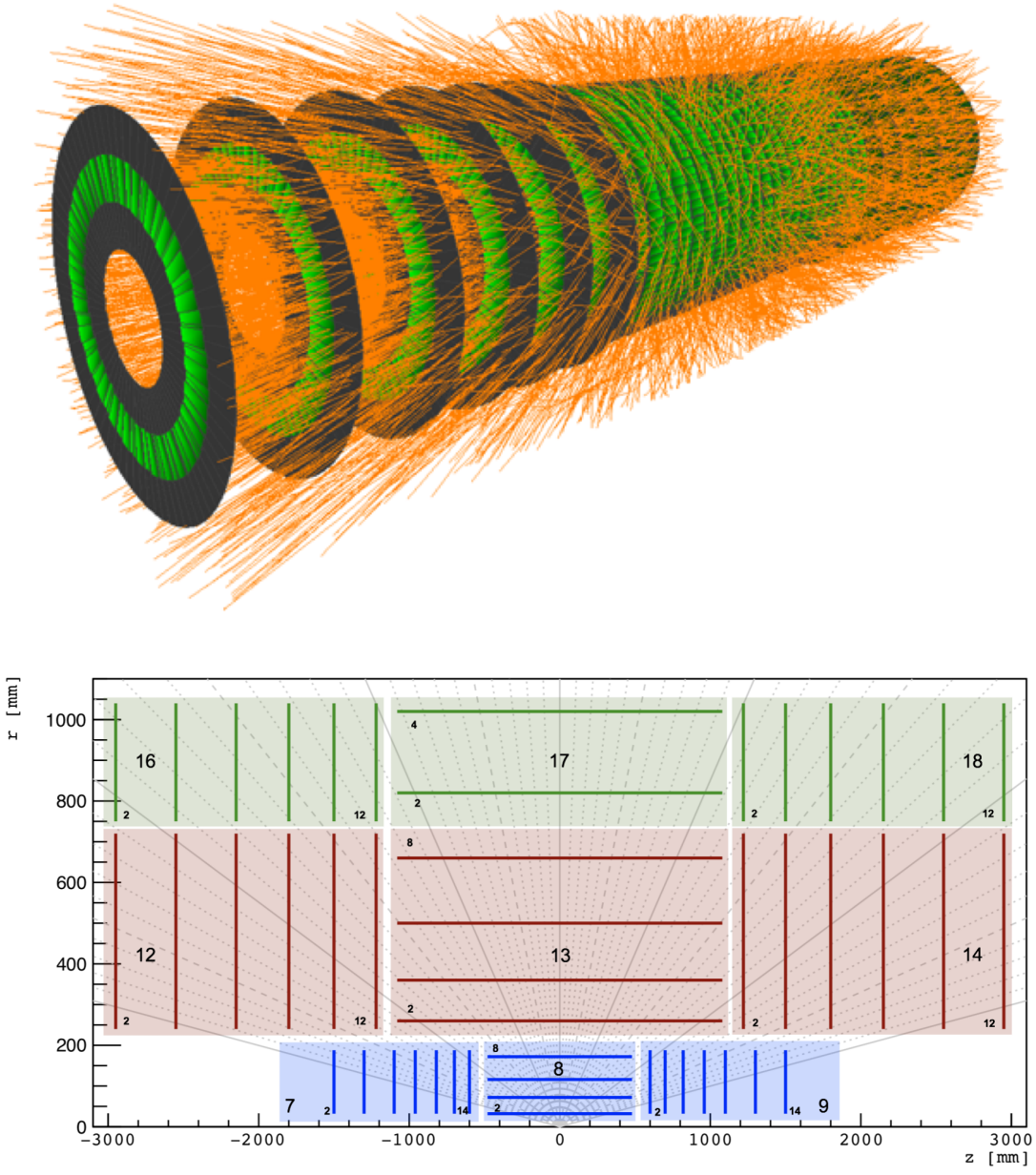

Figure 2.1: A simulated HL-LHC collision event (top) as seen by the TrackML tracking detector [7]. The detector schematic (bottom) shows the top half of the detector projected on the r-z plane. The z-axis is along the beam direction.

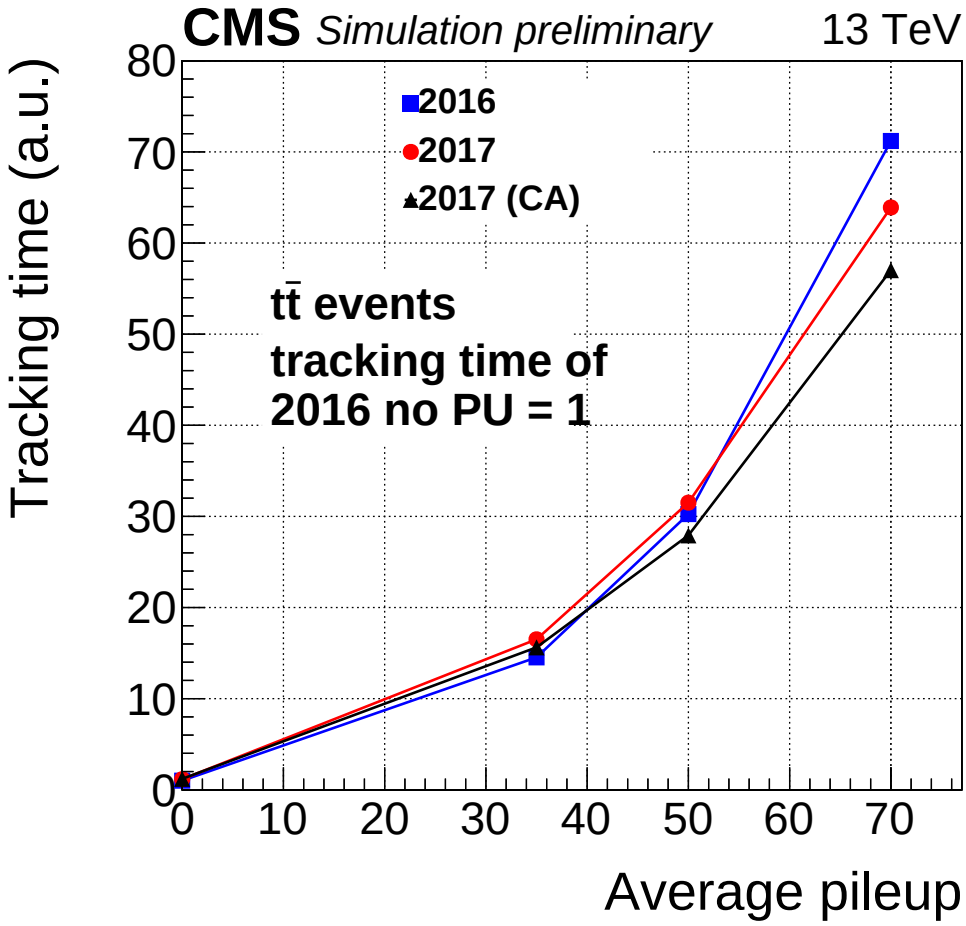

Figure 2.2: Reconstruction wall time per event as a function of the average number of interactions per bunch crossing $\langle \mu \rangle$. CMS time spent in tracking sequence for 2016 tracking, 2017 tracking with conventional seeding, and 2017 tracking with Cellular Automaton (CA) seeding [8].

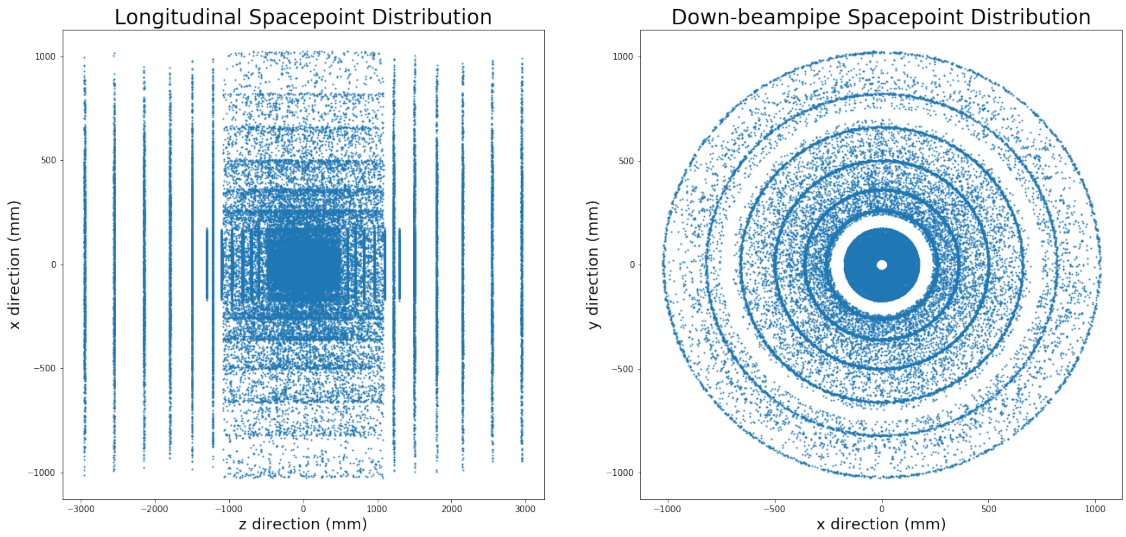

Figure 2.3: A typical event distribution of spacepoints projected on the x-z plane, parallel to the beam direction (left), and the x-y plane, orthogonal to the beam direction (right).

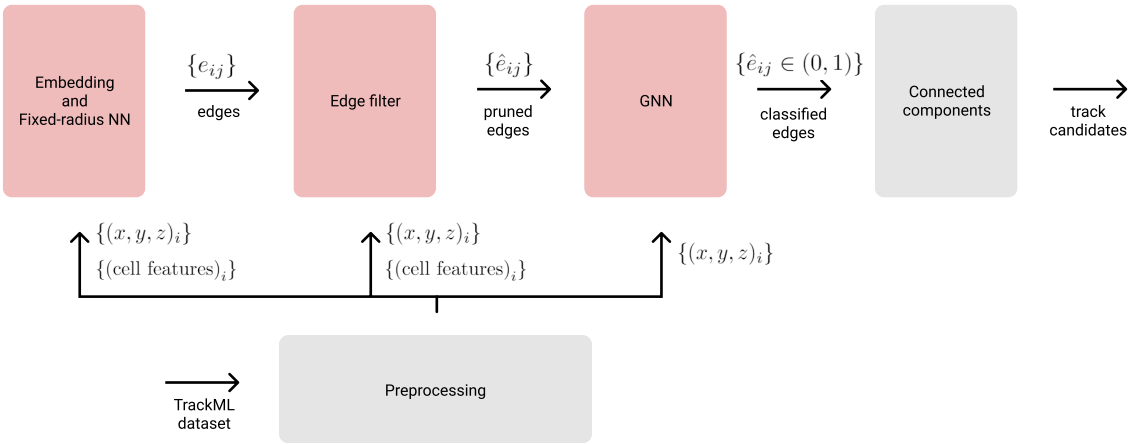

Figure 2.4: Stages of the TrackML track formation inference pipeline. Light red boxes are trainable stages.

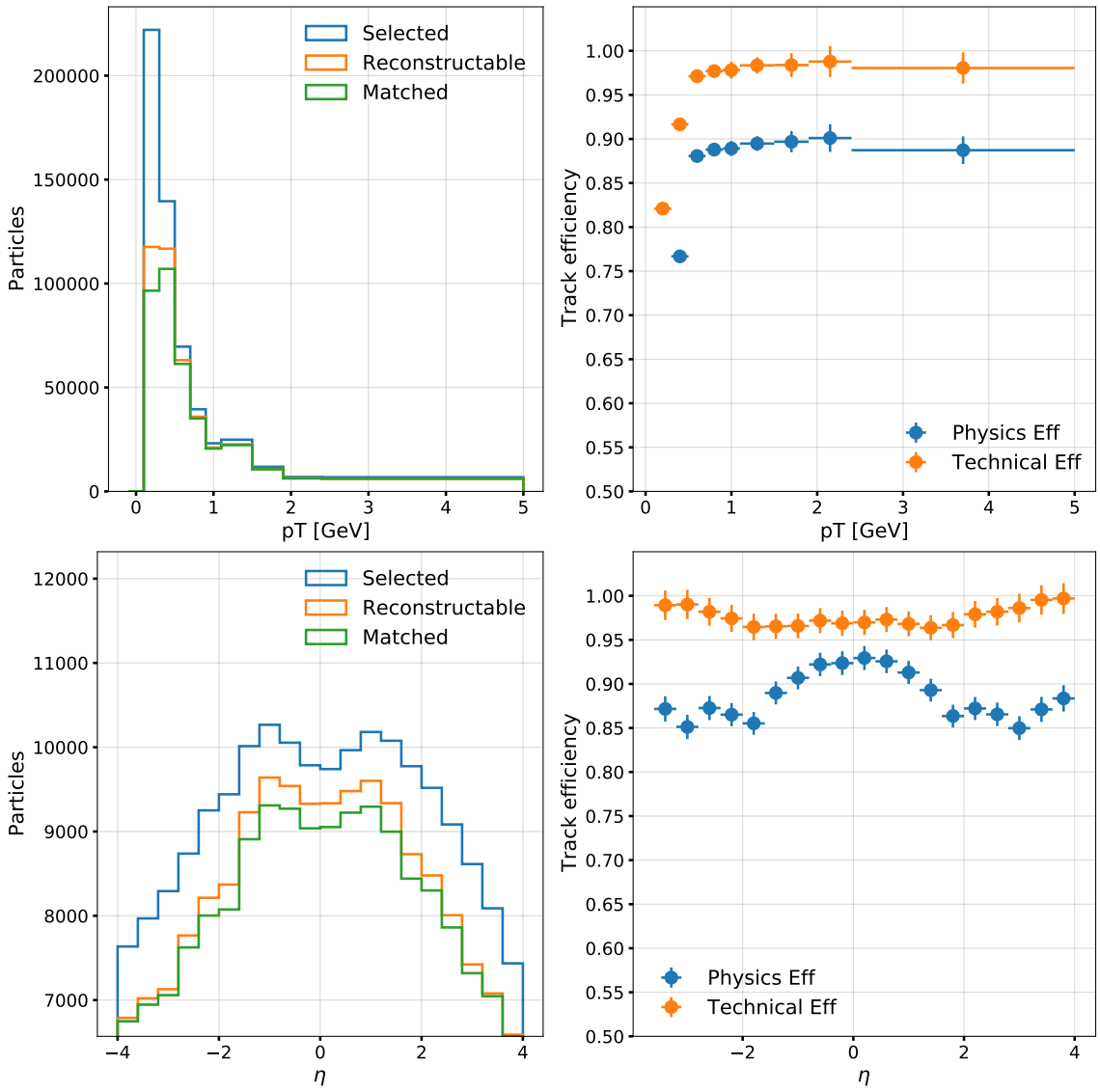

Figure 2.5: Top row: selected, reconstructable, and matched particles (left) and tracking efficiency (right) as a function of p_{T} for particles with $|\eta| < 4$. Bottom row: selected, reconstructable, and matched particles (left) and tracking efficiency (right) as a function of η for $p_{\mathrm{T}} > 0.5$ GeV. The definition of "selected", "reconstructable", and "matched" can be found in § 2.3.1

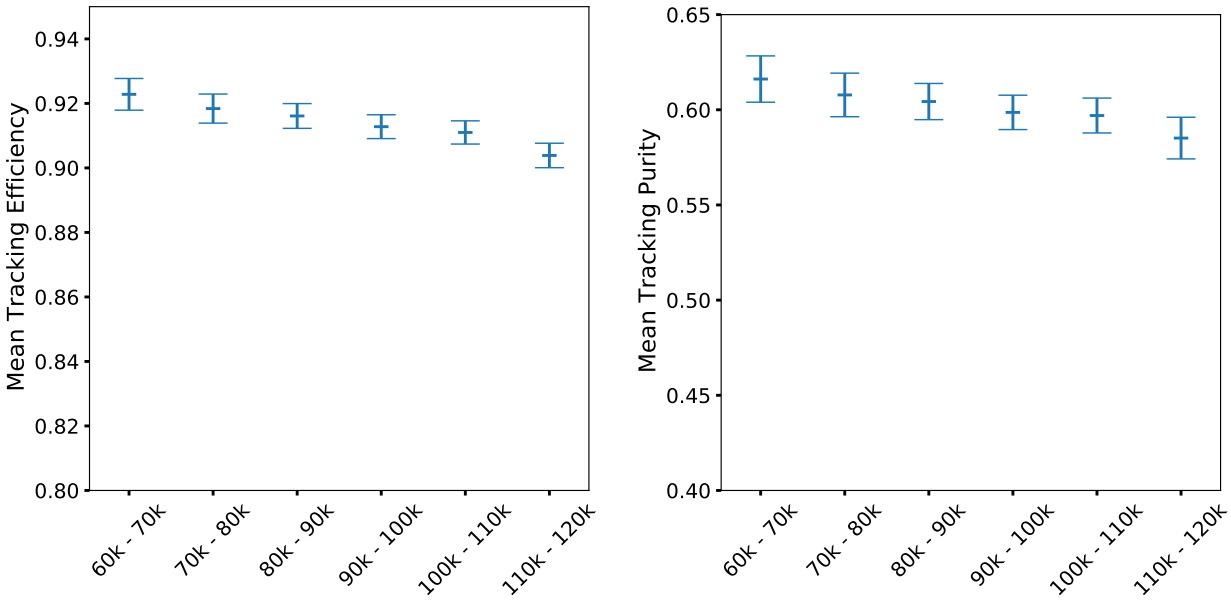

Figure 2.6: Mean and standard deviation of the technical efficiency (left) and purity (right) as a function of the total number of spacepoints in an event.

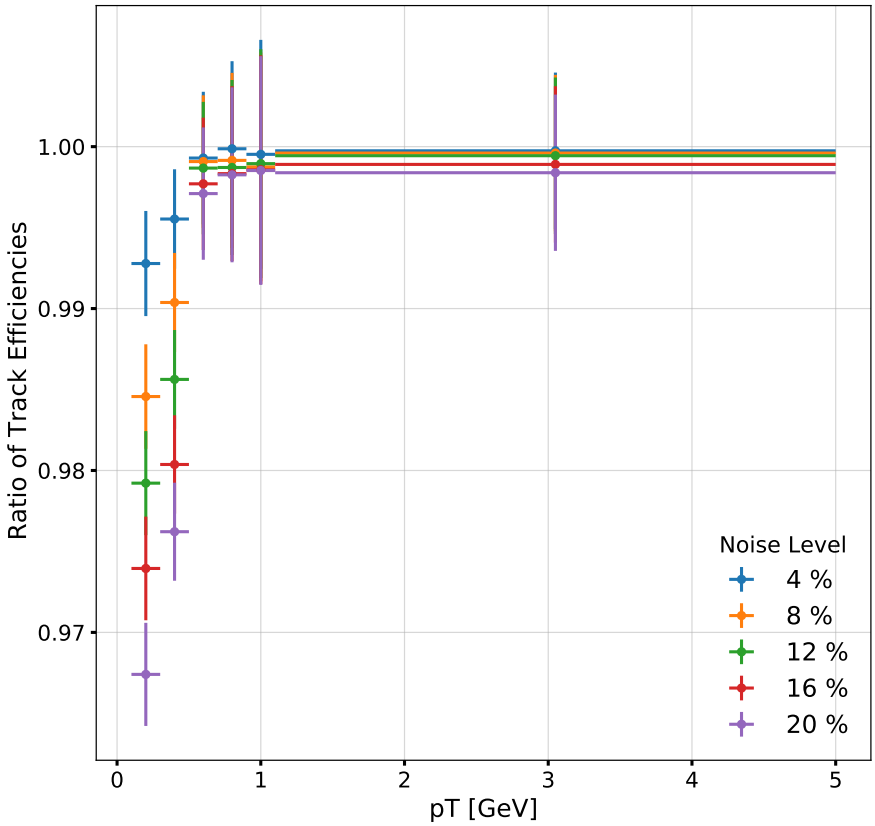

Figure 2.7: *Relative* technical efficiency as a function of p_{T}. Each curve shows the ratio of eff(noise $= N\%$)/eff(noise $= 0$).

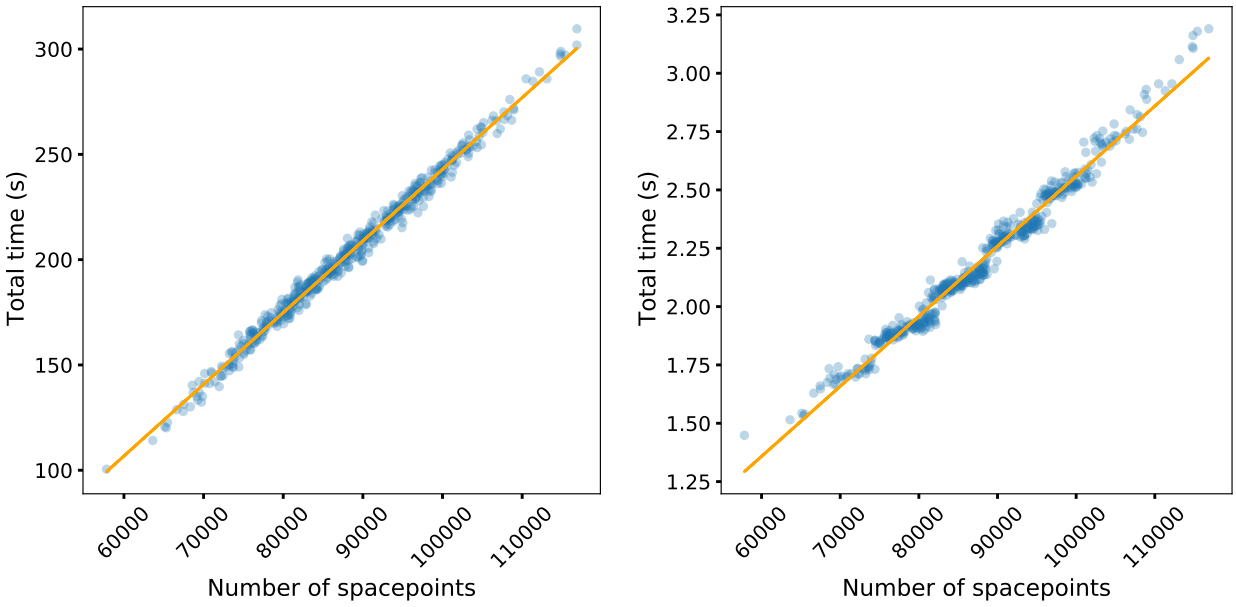

Figure 2.8: Total inference time as a function of number of spacepoints in each event for CPUs (left) and GPUs (right).

Chapter 3

ACCELERATING CNNS ON FPGAS FOR PARTICLE ENERGY RECONSTRUCTION

Deep learning (DL) techniques have emerged as powerful tools for addressing the intricate challenges in particle accelerator data reconstruction. While the next chapter will focus on the use of sparse CNNs for reconstructing hadronic particle showers in calorimeters, this chapter will discuss the higher-level task of particle energy regression, specifically for photons and electrons.

The LHC, with its complex operations and data-intensive processes, constantly seeks methods to refine and enhance its particle energy reconstruction. This becomes even more crucial when considering the demands of the High Luminosity LHC (HL-LHC). The introduction of the DeepCalo model to the LHC's toolkit promises advancements in this area. However, ensuring efficient deployment of these models is challenging, especially given the constraints posed by high data rates, dynamic experimental conditions, and the intensive computational resources they demand.

To address these challenges, this chapter introduces a design methodology that leverages hls4ml to deploy DeepCalo models on Field Programmable Gate Arrays (FPGAs). Through optimizing dataflow and processing mechanisms and integrating quantization-aware training for model compression, we aim to demonstrate a complete on-chip realization of these CNN architectures. Furthermore, this chapter provides a thorough analysis of design considerations, offering a structured framework for future DL implementations in similar scenarios.

In our testing, using the Xilinx Alveo U50 FPGA under LHC-specific conditions, we observed an inference latency of 1.34 ms for a batch of five images. This performance metric is noteworthy, showing a 5.6× improvement over current GPU-based methodologies. When

processing single images, the latencies for the standalone image model and the comprehensive model stood at 0.443 ms and 1.34 ms, respectively. Moreover, when benchmarked against the Ryzen-5600H CPU and Tesla V100 GPU, the methodology presented in this chapter exhibits a clear advantage, with speed gains of 14.1× and 7.9×, respectively.

This work was submitted to ACM TRETS on August 13 and is being reviewed. The author's personal contributions were mainly focused on training, validation and quantization, as well as resolution of disparities between FPGA and GPU performance.

3.1 Introduction

Among the many deep learning (DL) models recently designed for the HL-LHC[77], DeepCalo [78], a Keras-backed [79] DL architecture, has been tailored for simulation data from ATLAS, an all-purpose detector at the LHC. This design facilitates users in developing, training, refining, and evaluating convolutional neural networks (CNNs) to determine particle energies like those of electrons and protons [80] [81]. Differing from the traditional boosted decision tree (BDT) method, which only relies on derived scalar quantities [82], DeepCalo uses images from the electromagnetic calorimeter (ECAL). Its prowess surpasses BDTs by enhancing energy reconstruction precision by 11.9%-20.9% for electrons and 17.7%-27.5% for photons, varying by energy and detector zones [9]. DeepCalo presently offers a multi-modal 3.6M parameter model combining images, scalar metrics, and track vectors, and a leaner 1.8M parameter version focused solely on images.

Although DeepCalo performs well in energy reconstruction, its resource-intensive computation makes on-line inference challenging due to the strict latency requirements of the LHC. The LHC event-selection system consists of three tiers: level-1 trigger (L1T), high-level trigger (HLT), and offline reconstruction [83] [84]. L1T and HLT are online computing systems that operate at 40 MHz (100 kHz) with a latency of $\sim$1 μs ($\sim$10 ms) [19] [85]. The L1T is implemented using specialized electronics, while the HLT utilizes software running on a compute farm. According to [86], even for HLT, it still requires six GPU servers with a bandwidth of 24 Gb/s to run DeepCalo. Therefore it is more practical to handle this regres-

sion task offline. Furthermore, the future High-Luminosity LHC project plans to increase the beam intensity by 5× to 7× by 2027 [87] [88], which would make it extremely difficult, if not impossible, for CPU/GPU solutions to meet the stringent processing requirement.

Field Programmable Gate Arrays (FPGAs) enable customized data processing logic and have been broadly adopted to attain highly parallel dataflow processing with short latencies. The LHC has deployed FPGAs for online inference data analysis [89] [90] [91] [92] [93] [94], and facilitated the design with hls4ml [95]. hls4ml is a high-level synthesis (HLS) tool that converts high-level descriptions of ML algorithms into efficient FPGA implementations. However, the current hls4ml framework has limited support for converting large-scale models like DeepCalo. This is because hls4ml implements a fully on-chip dataflow architecture in order to avoid long-latency DRAM accesses. This approach needs to implement all the ML layers on an FPGA and therefore poses stringent constraints on the size and complexity of the model. Moreover, hls4ml's stream-based dataflow is constrained by the channel size, since resource consumption increases drastically when the channel size increases, and increased channel size is common in most CNNs. Also, quantization has been an effective technique to reduce design complexity. However, how to strike the balance between quantization errors and precision bit-widths has become a serious design concern when dealing with large models.

In this chapter, we present the first fully-automated design and optimization workflow based on hls4ml to implement DeepCalo models on FPGAs. We not only perform a comprehensive exploration of various key design factors but also propose a design that attains shorter latency (< 1 ms) than solutions on CPUs and GPUs. We extend the DeepCalo framework and integrate QKeras layers to perform quantization-aware training (QAT), which is crucial for minimizing resource consumption and maximizing model performance [96][97]. With a highly efficient streaming dataflow and optimized architectures of most neural network layers in hls4ml, we are able to support automatic conversion of large-scale CNNs to HLS and then implement on a Xilinx Alveo U50 FPGA board [98]. We further explore the rounding strategies used in hardware to reduce the quantization error when transferring the model to FPGAs, in order to obtain a good balance between resource utilization and accuracy.

3.1.1 Previous CNNs Implementations in hls4ml

FPGAs enable customized data processing logic and have been broadly adopted to attain highly parallel dataflow processing with short latency for the inference of ML models [99] [100] [101]. Previous studies in hls4ml have primarily focused on achieving millisecond-latency inference for CNNs on FPGAs. The initial work by Smith et al. [92] introduced support for streaming-based CNNs in hls4ml. This process removed allocating resources to monitor the location of elements in the sliding window or image corners management by computing and encoding positions as binary masks in advance. This strategy allows for the efficient retrieval of correct data in sliding windows through cooperation with buffered streams. Building upon this, Ref. [102] enhanced the stream-based approach with an optimized linebuffer architecture for real-time semantic segmentation tasks. The accelerator was compressed with automatic heterogeneous QAT and a filter ablation procedure, achieving a latency of 4.9 ms per image. These two convolution implementations, known as "encoded" and"linebuffer", are currently available options in hls4ml. The authors of [102] proposed using linebuffers which utilize shift registers to record previously seen pixels, thus reducing the memory needed to store duplicated pixels. For an image of size $H \times W$, with a convolution kernel of size $K \times L$, the line buffer allocates $K-1$ buffers (chain of shift registers) of depth W for the rows of the image, while the "encoded" implementation allocates K^2 buffers of depth $K \times (W-K+1)$ for the elements in the sliding input window.

In our work, we build upon the "linebuffer" scheme and focus on optimizing the dataflow of the streams along the channels. By leveraging the advantages highlighted in [102], we aim to enhance the efficiency and scalability of larger CNNs in hls4ml.

3.1.2 Data

The ATLAS detector at the Large Hadron Collider uses a cylindrical coordinate system to describe the trajectories of particles. Two main angular coordinates are used: η (pseudo-rapidity) and ϕ (azimuthal angle). The pseudorapidity, η, is defined as $\eta = -\ln[\tan(\theta/2)]$,

where θ is the polar angle with respect to the beam axis. In terms of regions determined by the magnitude of η, the detector is segmented into three distinct regions: the central region where $|\eta| < 2.5$, the crack region spanning $2.5 < |\eta| < 3.1$, and the forward (or 'endcap') region with $|\eta| > 3.1$. This is primarily due to differences in particle density, resolution, and detector response amongst these regions. For this work, we train DeepCalo using a dataset from the forward region.

The reconstruction of electrons and photons in the ATLAS detector primarily relies on two key sub-detectors: the Inner Detector and the Electromagnetic Calorimeter (ECAL). Working in tandem, the Inner Detector captures the momentum of particles, while the ECAL measures their energy. For particles of lower energy, momentum measurements from the Inner Detector often provide greater accuracy. However, as the energy of the incoming particle increases, the precision of the energy measurement from the ECAL becomes more critical.

The Inner Detector is designed to capture the trajectories of charged particles, including electrons. It is composed of silicon pixel detectors, silicon microstrip detectors, and a transition radiation tracker. Each particle's path within the detector results in a "track." This track provides detailed information about the particle, such as its momentum, charge, and vertex. The precision of the Inner Detector enables differentiation between closely spaced particle trajectories, a feature that becomes particularly important in high-particle-density environments found in high-energy collisions.

The ECAL, on the other hand, focuses on recording the energy deposited by electromagnetically interacting particles, notably electrons and photons. Designed with high granularity, the ECAL captures particle shower patterns in detail. When a primary electron or photon enters the ECAL, it can initiate an electromagnetic shower, leading to a cascade of secondary particles. The energy deposition from this cascade allows for determination of the initial particle's energy. To effectively measure this energy, the ECAL employs a clustering algorithm. The fundamental unit of this algorithm is the "TopoCluster." In this approach, high-energy cells act as seeds, with adjacent cells having significant energy being grouped

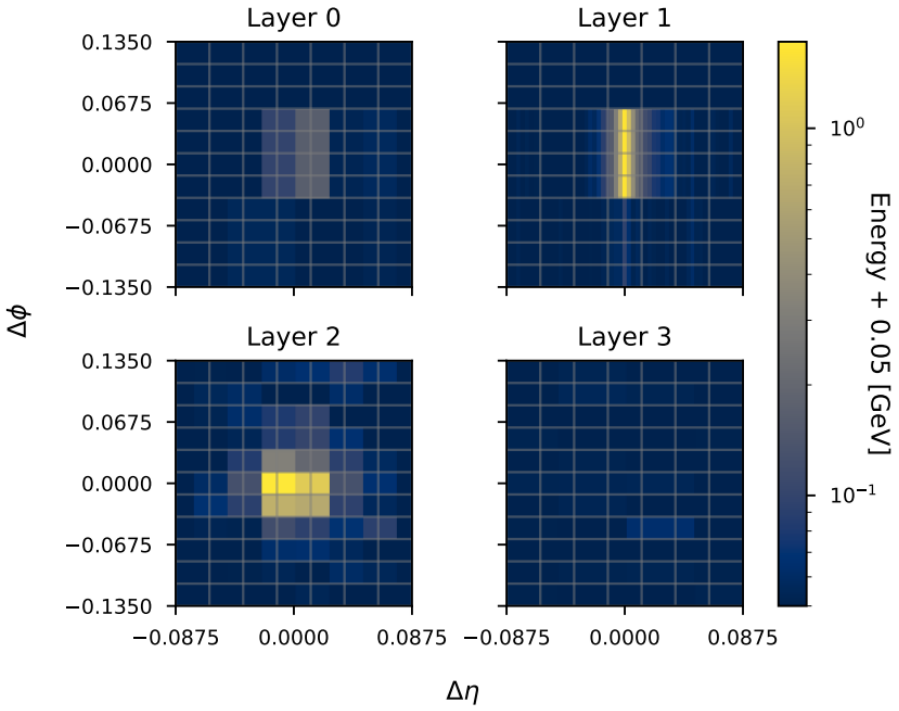

Figure 3.1: Example ECAL images for a single datapoint. [9]

around them. When a TopoCluster incorporates neighboring "satellite clusters," a "Super-Cluster" is formed. These satellite clusters are often indicative of secondary electromagnetic showers. The methodologies for developing SuperClusters vary based on the particle type and are informed by the tracking data provided by the Inner Detector.

In this work, the DeepCalo model is utilized on an FPGA to reconstruct the energy of incident electrons using multiple data inputs from the ATLAS detector. These data inputs comprise:

- **ECAL images** – These are cell readings harmonized across the different layers of the Electromagnetic Calorimeter (ECAL). They capture the energy distribution as particles traverse the ECAL.

- **ECAL time images** – Time information from the ECAL cells, recorded relative to the

ATLAS clock. This helps in distinguishing between signal and potential background noise, including pileup.

- **ECAL noise images** – Representations of noise from the ECAL cells, aiding in noise mitigation during the reconstruction process.

- **ECAL gain images** – Images showing the gain applied to each ECAL cell to enhance signal clarity.

- **Scalar variables** – These are variables that capture pertinent energy-related details. They have been used traditionally for training Boosted Decision Trees (BDTs) in energy reconstruction. Their selection is influenced by both experimental data and intuitive reasoning. Some of these variables can potentially be reconstructed from the ECAL images and were therefore omitted.

- **Tracking data** – A list of tracks derived from the Inner Detector (ID) of ATLAS. For each track that is within a cone of $\Delta R < 0.4$ relative to the prompt particle, information includes variables such as:

  - d_0 - The impact parameter in the transverse plane, which signifies the closest approach of the track to the z axis.
  - z_0 - The z-coordinate of the point where the track makes its closest approach to the z axis.
  - θ_0 - The polar angle of the momentum vector.
  - ΔR - The distance in the $\eta - \phi$ space between the track and the prompt particle.

These diverse data types are useful for achieving a comprehensive understanding of the incident electron's energy and enable the model to perform accurate reconstructions, and are the motivation behind the structure of the DeepCalo model, described in the next section.

It is pertinent to note that the data utilized in this study originates from Monte Carlo (MC) simulations rather than direct measurements from the ATLAS detector. MC simulations play a pivotal role in particle physics, providing a way to model complex systems by generating a vast number of random samples based on known physical processes. These simulations imitate the behavior of particles as they traverse and interact with the detector, yielding synthetic data that replicates expected experimental outcomes. Utilizing MC simulations allows for a controlled environment where uncertainties and systematic effects can be thoroughly understood and accounted for, serving as a crucial tool in validating and fine-tuning experimental analyses before the actual detector data is considered.

Finally, the process of event selection should be mentioned. Event selection is crucial in refining the data for analysis. It primarily involves the filtration of events, where an event refers to a specific collision and its byproducts, to retain only those relevant for the intended analyses or those that meet quality standards. To generate the final dataset, files containing Monte Carlo (MC) events from various physics channels, such as $Z \to ee$ and $Z \to ee\gamma$, are accumulated. The inclusion criteria for these channels are often driven by ensuring appropriate statistics within a target E_T range.

After an initial general preselection, which includes some elementary cuts and data cleansing, photons are categorized using truth-matching. Electrons, however, undergo a more intricate selection based on the 'tag and probe' (T&P) method. Designed to identify electrons in an unbiased manner, T&P entails pairing electrons (the tag and the probe) wherein if an electron candidate combines with the tag to represent a particular mass (e.g., that of the Z boson), it is considered a genuine electron. Notably, the T&P process, while formulating this dataset, was specifically crafted to function with both MC simulations and real data.

In addition to these primary selection processes, further refinements were implemented to optimize the neural network training. For instance, the photon datasets were restricted to include only photons with $E_T \leq 100$ GeV. Finally, to align with the common practices of ATLAS analyses, a loose likelihood particle identification (PID) criterion was applied. However, it's noteworthy that such a criterion can exclude a significant portion of data

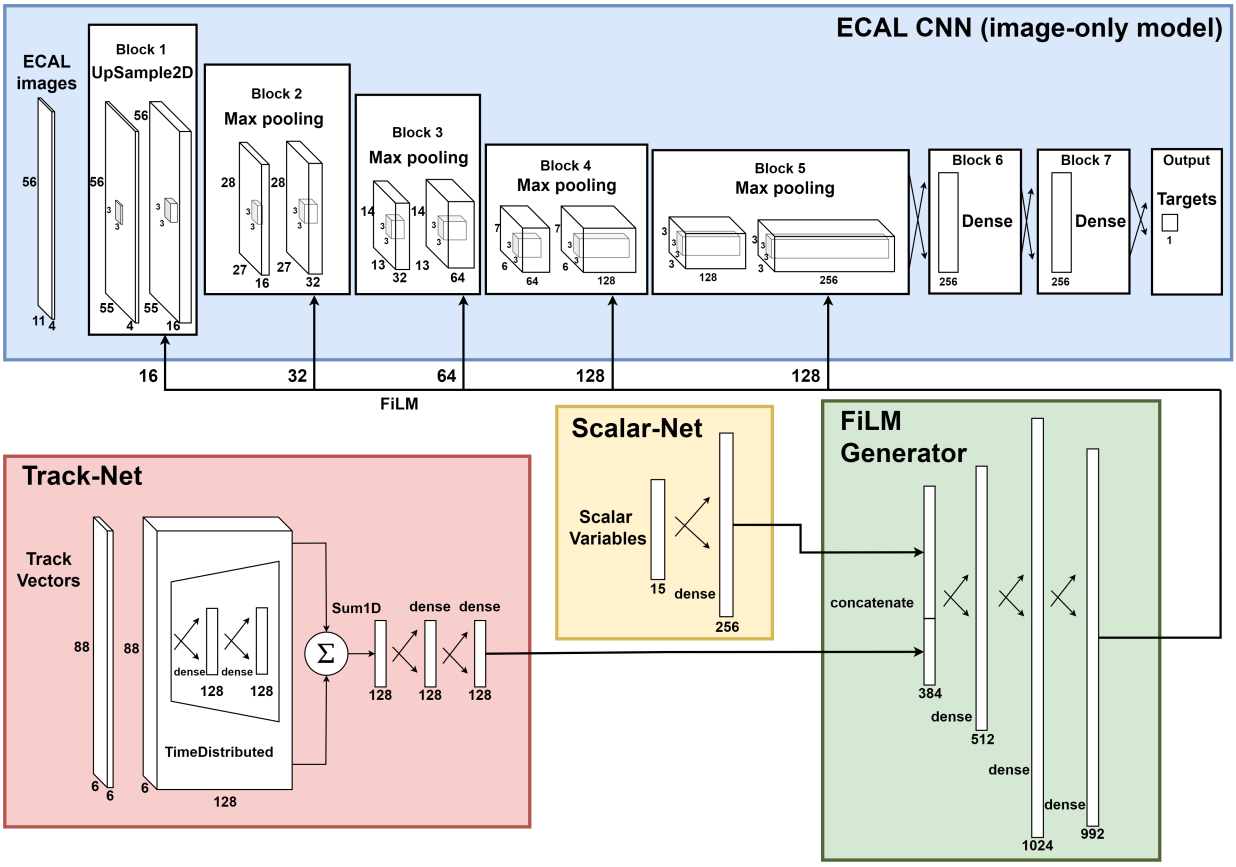

Figure 3.2: Blueprint of the Deepcalo *full model*. Beyond the principal CNN (*image-only model*), the *full model* encapsulates three additional modules: Track Net, Scalar Net, and the FiLM Generator.

points, emphasizing the balance that needs to be maintained between data quality and volume.

3.1.3 DeepCalo: Deep Learning Framework

A graphic representation of the model structure is provided in Figure 3.2. Within the whole system, the ECAL CNN, or the so-called image-only model, is the primary discriminative component. This model adopts a VGG-inspired [103] structure, incorporating five 2D convo-

lution blocks paired with three densely connected blocks. Each convolution and dense layer is sequentially trailed by a batch normalization process and a rectified linear unit (ReLU) activation function. The opening block takes on the task of reshaping the input pixel array into a more squarish format, which comprises an Upsampling2D layer paired with a 5×5 convolutional layer. The later blocks share a uniform design: initiating with a 2×2 max-pooling layer, then a tandem of a singular convolutional layer possessing a 3×3 kernel dimension. The convolution layers in the l^{th} segment harbor $16l$ filtering units. The initial pair of dense segments house layers with 256 processing units each, whereas the terminal dense block introduces a single-neuron layer, culminating in the definitive regression result, with ReLU steering the final activation.

Separate submodels are allocated for the scalar measurements and tracking vectors. The Scalar Net features a densely connected layer loaded with 256 units, in contrast, the Track Net is outfitted with a TimeDistributed layer that sequentially applies a dense network to interpret every track vector tied to a particular event. The culmination of outputs from each vector is aggregated and then directed into an ancillary dense network for advanced processing. To facilitate the integration of supplementary input metrics, the model introduces feature-wise linear modulation (FiLM) layers [104]. These layers are strategically placed post the premier convolutional layer within each 2D convolution block. This configuration allows the CNN's performance to be adaptively and dynamically tailored based on the trio of input metrics. The combined outputs from Scalar and Track Nets are channeled into the FiLM generator, a rudimentary fully-linked network. This generator's mandate is to produce the scaling and offset values for the FiLM layers, which then adjust the CNN's feature map compositions.

3.2 Methodology

3.2.1 Quantization

One of the primary considerations when migrating a machine-learning model from GPUs to FPGAs is quantization. The inherent differences between the arithmetic representations used in these platforms necessitate a transition from floating-point to fixed-point computation. This transition aims to optimize resource utilization on platforms like FPGAs, which have constraints on computational power and memory, making the traditional floating-point calculations suboptimal.

On GPUs, the preferred mode of computation is often based on floating-point representation. In this system, a number is characterized by a significand and a base. For instance, the number 12.375 in binary is represented as:

$$(1.100011)_2 \times 2^3 = 12.375 \tag{3.1}$$

Where $(1.100011)_2$ is the significand and 2^3 is the exponent.

However, FPGAs, due to their limited resources and architecture, find such floating-point calculations to be costly in terms of both computational cycles and resource utilization. As a remedy, FPGAs employ fixed-point arithmetic, where numbers have a fixed number of bits allocated for the integer and fractional parts. For instance, if we represent 12.375 in a fixed-point format with 4 integer bits and 4 fractional bits, it can be illustrated as:

$$1100.0110 \tag{3.2}$$

The primary advantage of fixed-point arithmetic over floating-point is that it simplifies hardware design for arithmetic operations, making it faster and less resource-intensive.

3.2.2 Post-Training Quantization (PTQ)

Post-training quantization (PTQ) is a method that involves the quantization of model parameters, i.e., weights and biases, after the training process is completed. The essence of

PTQ is to reduce the precision of the floating-point numbers in the model to fixed-point representation, thus making it suitable for deployment on hardware platforms such as FPGAs.

The typical steps involved in PTQ are:

1. Train the model using standard floating-point precision, typically 32-bit (FP32).

2. Convert the trained model's weights and biases to a lower bit-width fixed-point representation. A single precision can be chosen for the entire model, or differing precisions can be chosen on a layer-by-layer basis.

3. Optionally, calibrate the quantized model on a subset of the training data or validation set to fine-tune the quantization parameters, ensuring minimal loss of accuracy.

The primary advantage of PTQ is its simplicity, as it does not require any modifications to the training procedure. However, depending on the model and the reduction in bit-width, there may be a significant drop in the model's accuracy.

3.2.3 Quantization-Aware Training (QAT)

Quantization-aware training (QAT) involves the introduction of quantization during the training process itself, ensuring that the model is already optimized for reduced precision before deployment. In this approach, forward and backward passes during training are modified to simulate the effects of quantization.

The general procedure for QAT includes:

1. Introduce quantization operations in the forward pass, simulating the inference in the target hardware.

2. During the backward pass, compute gradients assuming the quantization as the identity function, to ensure the gradients differentiable.

3. Fine-tune the model with these quantization-aware modifications until convergence.

QAT often results in models that are more robust to the effects of quantization compared to PTQ. This is because the model learns to compensate for the quantization errors during the training process itself. As a result, QAT typically provides a more graceful trade-off between model size, computational complexity, and accuracy.

Tuning Precision

Addressing the complexity of balancing model performance and resource utilization when optimizing the QAT configuration, we followed a two-step approach. Initially, we conducted an extensive search for the bit widths of both models using PTQ, which serves as a reference baseline. For the image-only model, the bit widths ranged from 32 total bits down to 2 total bits, with the number of integer bits varying. However, due to the increased complexity of the full model, we only explored bit widths from 16 total bits to 2 total bits. During the PTQ evaluation, we noticed that providing similar bits for both the integer and fraction parts resulted in the lowest MAE for both models. This finding guided our subsequent QAT experiments, where we set the integer bits to half of the total bits.

3.2.4 HLS Optimizations

In the context of High-Level Synthesis (HLS), a variety of optimizations are essential for efficient data processing and transfer, particularly when implementing architectures like DeepCalo. In this work, we investigate five major optimizations: stream-based architecture, analysis of different stream types, architecture optimizations for individual layers, determination of bit-width in accumulators, and profiling the HLS model. This section provides a brief overview of these optimizations. For a detailed exploration of HLS optimizations pertaining to DeepCalo, including figures, algorithms, and discussions, the reader is referred to the appendix. For the core methodology of DeepCalo itself, please see the Introduction section.

One primary aspect of HLS optimization is data transfer between layers. While the array-based method can handle significant data volumes, it may present challenges in storage

requirements and potential timing overheads. Conversely, the stream-based approach, using FIFOs (First In First Out buffers), tends to offer better resource utilization and is more compatible with dataflow architectures. The stream-based approach was chosen for DeepCalo due to these advantages.

The choice of stream type is also crucial. Different streaming methods, such as stream-of-struct, single-stream, and array-of-streams, have varying impacts on resource usage, latency, and the architecture of processing elements. The optimal choice depends on the specifics of the application. When considering channel sizes, the selection between a single-stream and an array-of-streams approach becomes relevant. A switch function in the hls4ml tool has been developed to allow automatic data type conversion based on channel size.

In the realm of HLS optimization, architectural considerations play a pivotal role. One primary focus was on refining data transmission patterns to enhance the efficiency of processing units. Specifically, modifications were made to the transmission patterns in max-pooling and dense layers. By analyzing the interplay between layers and their corresponding data flow, we identified bottlenecks and adjusted the architecture accordingly. These adjustments aimed to reduce synthesis time, streamline performance, and ensure a consistent data flow, thereby enhancing the efficiency of the entire system.

Additionally, this study delved into the significance of bit-width in accumulators. Given the potential variations in values due to different rounding strategies and the challenges posed by non-quantized input data, it became imperative to understand the nuances of bit allocation. We assessed and implemented measures to mitigate these issues, experimenting with various bit-width configurations and evaluating their impact on system precision and computational overhead.

3.3 Results

This section presents comprehensive experiments with the DeepCalo design and optimizations for FPGAs. The latencies of the models on FPGAs are compared with other computing platforms in Section 3.3.1. In Section 3.3.2, we evaluated the performance of two DeepCalo

models under different quantization schemes and fixed-point precisions.

All quantization processes include weights and activations with homogeneous bit width. The fixed-point format is based on the Vivado HLS *ap_fixed* type [105]. Furthermore, the batch normalization folding technique [106] [107] is adopted during the HLS conversion to further lower resource utilization and latency. Specifically, the batch normalization layers that follow the convolutional and dense layers are fused together.

The design is converted to HLS C++ using hls4ml 0.5.1 and then synthesized with Vivado HLS 2019.2, targeting a Xilinx Alveo U50 FPGA with a clock frequency of 200MHz.

3.3.1 Coprocessor Performance Comparison

We first present a comparative analysis of the image-only model and the full model on different coprocessors: CPUs, GPUs, and FPGAs. The evaluation metrics include latency, speedup, power consumption, and energy consumption, with speedup normalized against the Ryzen 5 5600H CPU. We conducted experiments with batch sizes of 1, 5, and 100, reflecting typical electron collision scenarios in LHC settings.

To ensure accuracy, 10^5 repetitions were performed for each measurement of latency, power, and energy on CPUs and GPUs, calculating average values. GPU power was measured at 10-millisecond intervals, while CPU power was sampled every 20 milliseconds. For FPGAs, latency assessment involved 10^4 runs, measuring the time for data loading, processing, and writing back to DRAM in two dataflow schemes: mixed-type and all-single-stream. FPGA power consumption was averaged over 200 runs using the Xilinx Runtime Library [108].

In the analysis of the image-only model, both CPUs and GPUs showed a decrease in latency per batch with increasing batch size, although power consumption increased. GPUs, in particular, demonstrated significant performance improvement at a batch size of 100 due to their enhanced parallelism. The mixed-type dataflow scheme in FPGAs exhibited shorter latency across all batch sizes compared to the all-single-stream scheme, achieving speedups of up to 1.889× for batch size 100. This efficiency also led to slightly higher power consumption with larger batch sizes.

Table 3.1: Performance comparisons of the (a) image-only model and (b)full model on processing platforms: CPUs, GPUs, and FPGAs. Both models have floating-point precision for the CPUs and GPUs, whereas the precision for the FPGA is ap_fixed<8,2>.

(a) CNN (Image-only model)

Coprocessor	CPU			GPU			FPGA	
Type	Ryzen 7 3700X	Ryzen 5 5600H	Intel i5-12400F	RTX 2070 Super	Tesla V100	RTX 2080 Ti	single-stream	mixed-type
Batch=1								
Latency	6ms	6.227ms	4.439ms	5.98ms	3.5ms	5.8ms	0.697ms	0.443ms
Speedup	1.038×	**1**×	1.403×	1.041×	1.779×	1.074×	8.934×	**14.056**×
Power	53.82W	27.38W	40.38W	35.68W	61.08W	67.83W	18.33W	20W
Energy	322.92mJ	170.495mJ	179.247mJ	213.366mJ	213.78mJ	393.414mJ	12.778mJ	8.86mJ
Batch=5								
Latency	10ms	10.165ms	8.03ms	8ms	3.6ms	6ms	2.395ms	1.34ms
Speedup	1.017×	**1**×	1.266×	1.271×	2.824×	1.694×	4.244×	7.586×
Power	62.03W	35.22W	47.29W	40.30W	62.31W	66.23W	19W	23W
Energy	620.3mJ	358.01mJ	379.74mJ	322.4mJ	224.315mJ	397.38mJ	45.505mJ	30.82mJ
Batch=100								
Latency	50ms	72.8ms	49.3ms	8.4ms	4.8ms	6.7ms	44.4ms	23.5ms
Speedup	1.456×	**1**×	1.477×	8.667×	15.167×	10.866×	1.64×	3.098×
Power	81.02W	40.54W	62.46W	93.25W	92.22W	87.21W	19W	24W
Energy	4.051J	2.951J	3.079J	0.783J	0.443J	0.584J	0.844J	0.564J

(b) Full model

Coprocessor	CPU			GPU			FPGA	
Type	Ryzen 7 3700X	Ryzen 5 5600H	AMD EPYC 7262	RTX 2070 Super	Tesla V100	RTX 2080 Ti	single-stream	mixed-type
Batch=1								
Latency	7.52ms	8.75ms	5.865ms	8.47ms	4.8ms	8.2ms	1.106ms	0.898ms
Speedup	1.164×	**1**×	1.492×	1.033×	1.823×	1.067×	7.911×	**9.744**×
Power	53.73W	29.13W	42.65W	49.77W	60.11W	64.54W	19.76W	20.75W
Energy	404.05mJ	254.888mJ	250.142mJ	421.552mJ	288.528mJ	529.228mJ	21.855mJ	18.634mJ
Batch=5								
Latency	11.5ms	13.45ms	10.545ms	9.75ms	5.1ms	7ms	2.695ms	1.485ms
Speedup	1.17×	**1**×	1.275×	1.379×	2.637×	1.921×	4.991×	9.057×
Power	62.44W	37.67W	48.94W	51.83W	61.73W	84.18W	21W	23.775W
Energy	718.06mJ	506.66mJ	516.07mJ	505.345mJ	314.825mJ	589.26mJ	56.595mJ	35.305mJ
Batch=100								
Latency	86ms	119ms	91.4ms	15ms	7.1ms	9.5ms	53.9ms	29.7ms
Speedup	1.384×	**1**×	1.302×	7.933×	16.761×	12.526×	2.208×	4.007×
Power	86.51W	47.24W	75.41W	93.59W	116.4W	112.74W	21W	23.833W
Energy	7.44J	5.622J	6.893J	1.404J	0.826J	1.071J	1.132J	0.708J

FPGA designs surpassed CPUs and GPUs in speedup and energy efficiency for smaller batch sizes but were slightly outperformed by GPUs at batch size 100, mainly due to the GPUs' parallel processing capabilities. However, FPGAs still showed better speedup compared to CPUs and were only marginally less energy-efficient than the Tesla V100 GPU, a notable observation considering the substantial price difference between the Alveo U50 FPGA and the Tesla V100 GPU [109][110].

The full model's performance revealed similar trends. Traffic jams in FIFOs due to additional submodels led to reduced dataflow efficiency in the full model, especially at larger batch sizes. Despite this, FPGAs continued to demonstrate lower latency and energy consumption compared to CPUs and GPUs for smaller batch sizes. For batch size 100, while GPUs outperformed FPGAs in terms of speedup, FPGAs maintained lower energy consumption.

3.3.2 Quantization Technique and Model Architecture Performance Comparison

Next, we evaluate the effectiveness of DeepCalo models trained under QAT and PTQ approaches for FPGAs. Two evaluation metrics are used: Mean Absolute Error (MAE) and Interquartile Range (IQR). MAE quantifies the average prediction error, and is defined as the average absolute difference between predicted ($\hat{y}$) and true energy (y) over n data points, as shown in Eq. 3.3. IQR assesses the spread of relative errors (RE) and is defined for specific percentiles as shown in Eq. 3.4.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i| \tag{3.3}$$

$$IQR_m(RE) = P_m(RE) - P_{100-m}(RE), \quad RE = \frac{\hat{y} - y}{y} \tag{3.4}$$

Note that for QAT, we used the same hyperparameters as for the floating-point model training. The dataset, containing around 1.2 million electrons from the ECAL endcap region, was divided into 70% training, 15% validation, and 15% test sets. We employed the *Nadam*

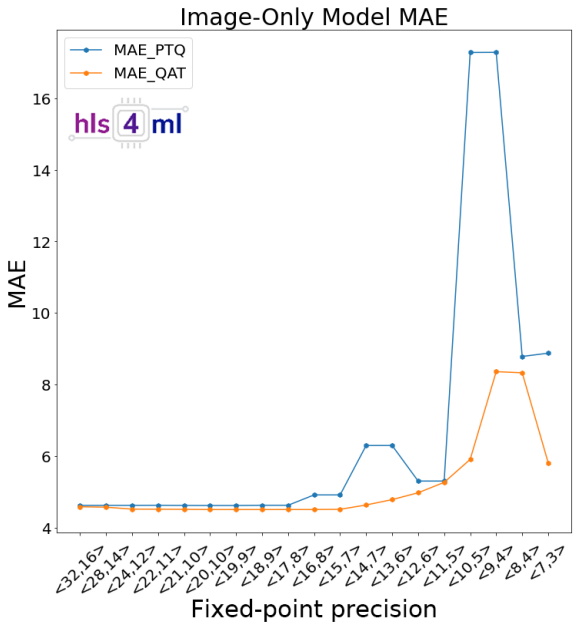

Figure 3.3: Performance of the image-only model under PTQ and QAT. Evaluation metrics are MAE and IQR.

optimizer and a *cyclical learning rate* (CLR) over 300 epochs. The CLR had a 3-epoch stepsize, with a baseline learning rate of $5e^{-4}$ and a maximum of $7e^{-2}$. The initial learning rate was determined by scanning between $1e^{-5}$ and $1e^{-2}$ before training. Models were trained with a batch size of 1,024 using the logcosh loss function and an early stopping criterion, monitored on the validation loss with a Δ_{min} of 0.001 and a patience of 150 epochs.

First, we analyze the performance impact of QAT compared to PTQ on the DeepCalo models. Figures 3.3 and 3.4 illustrate the performance of the image-only and full models, respectively. Across a wide range of bit widths, QAT consistently outperformed PTQ in terms of both MAE and IQR. Notably, in the full model, QAT led to a less pronounced degradation in MAE with decreasing bit width.

Next, we compare the image-only model and the full model using QAT. Figure 3.5 presents the MAE and IQR for both models as a function of precision. The full model consistently outperformed the image-only model, especially at smaller bit widths. This suggests that

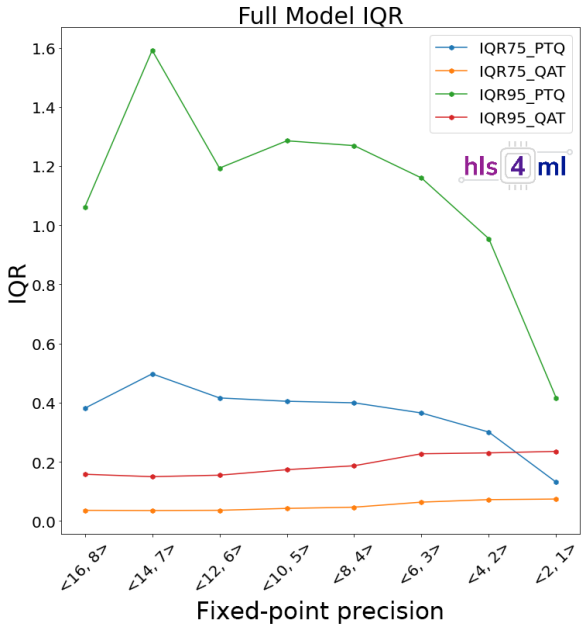

Figure 3.4: Performance of the full model under PTQ and QAT. Evaluation metrics are MAE and IQR.

the full model, with its additional inputs of high-level features and tracks, is more adept at compensating for reduced bit width, although the specifics of this mechanism have not been fully explored.

3.4 Conclusion

In this work, we present an automated design and optimization workflow based on hls4ml to deploy DeepCalo models on a Xilinx Alveo U50 FPGA, with potential applicability to other large CNNs. We highlight the importance of choosing the appropriate stream-based dataflow to balance resource utilization and achieve the desired latency. To ensure the accuracy and reliability of the converted HLS models, we illustrate our approach to eliminating potential quantization errors at an early stage, prior to conversion to HLS. To validate our methodology, we compare the FPGA implementation of both models with other co-processors using key performance indicators such as latency, speedup, power consumption, and energy effi-

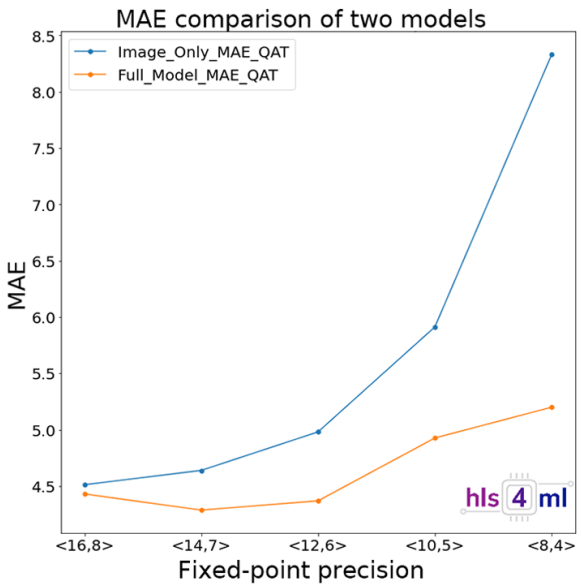

Figure 3.5: Comparative performance of the image-only model and the full model using QAT. Evaluation metrics are MAE and IQR.

ciency. Examining actual LHC conditions, the image-only model yielded an inference latency of 1.34 ms for every 5 images, which is 5.6 times faster than the existing GPU-based system. At a batch size of one, the image-only model demonstrates a latency of 0.443 ms, while the full model exhibits a latency of 1.34 ms, achieving the Level-1 Trigger requirement. Compared to the Ryzen-5600H CPU and the Tesla V100 GPU, the image-only (full) model achieves speedups of up to 14.1× (9.7×) and 7.9× (5.3×), respectively. Finally, we show that quantization-aware training significantly reduces resource usage and preserves performance compared to post-training quantization.

Chapter 4

ACCELERATING HADRONIC CALORIMETRY WITH SPARSE POINT-VOXEL CONVOLUTIONAL NEURAL NETWORKS

Due to the stochastic nature of hadronic interactions, particle showers from hadrons can vary greatly in their size and shape. Recovering all energy deposits from a hadronic shower within a calorimeter into a single cluster can be challenging and requires an algorithm that accommodates the large variation present in such showers. In this study, we demonstrate the potential of a deep learning based algorithm based on sparse point-voxel convolutional neural networks (SPVCNN) to perform hadronic calorimetry using Compact Muon Solenoid detector on the Large Hadron Collider with the hadron calorimeter and high granularity calorimeter. By employing a modified object condensation loss, we train the network to group cell deposits into clusters while filtering out noise. We show that SPVCNN performs comparably to generic topological cluster-based methods in both pileup and no pileup scenarios, with the added advantage of acceleration using GPUs, and further algorithmic development with better datasets. This type of acceleration, as part of heterogeneous computing frameworks, will be crucial for the High-Luminosity Large Hadron Collider (HL-LHC). Our findings indicate that SPVCNN can provide efficient and accurate calorimetry solutions, particularly for high level trigger (HLT) applications with latency on the order of milliseconds.

This work was presented at the Fast Machine Learning for Science Workshop 2022 [111] and a paper is in progress. The author's personal contributions included most aspects of the work, including project coordination, design, training, and evaluation.

4.1 Introduction

At the Large Hadron Collider (LHC) at the European Center for Nuclear Research (CERN), high-energy proton collisions are reconstructed using a tiered set of detectors. One of these detectors is commonly referred to as the Hadron Calorimeter. Its aim is to reconstruct showers originating from the interaction of hadronic particles with a density material. As with its namesake, the challenge of the Hcal is to recover the total energy of a single particle from the shower of secondary particles that are produced when a hadron collides with a dense material. Furthermore, due to the large variation present in Hadronic showers, there is a need for an algorithm that can adapt to a wide variety of shower topologies. Finally, a major challenge of calorimeter clustering is the ability to identify all of these clusters simultaneously. Typical collisions at the LHC can lead to several thousands of clusters. The challenge of reconstructing these all simultaneously in a short amount of time can be challenging. Here, deep learning can provide a solution through the native parallelization of the algorithm, particularly when deployed on a GPU. In this paper, we present a deep-learning algorithm capable of clustering hadronic calorimeter energy deposits into particles.

In the Compact Muon Solenoid (CMS) experiment at the LHC, hadronic calorimeters are employed to measure the energy of hadrons, such as protons and neutrons, which are particles composed of quarks and gluons. These calorimeters detect the energy deposited by hadrons as they interact with the detector material, producing a cascade of secondary particles. However, accurate measurement of hadron energy and position in a calorimeter is challenging due to factors such as fluctuations in the shower development, noise, and pileup effects. Existing methods, such as particle flow (PF) HCAL clustering [112], have limitations in incorporating the full shower information, in particular the hadronic shower depth information, and are not optimal for the increasing computational demands of the HL-LHC due to their lack of parallelism.

In this paper, we propose the use of sparse point-voxel convolutional neural networks (SPVCNN)[113] for hadronic calorimetry tasks. Sparse neural networks have shown promis-

ing results in various applications, such as 3D object recognition and semantic segmentation. Their ability to efficiently process sparse data makes them suitable for calorimetry tasks, where the data is often sparse and requires complex processing within latency constraints.

To achieve the same performance as rule based clustering algorithms, we apply the SPVCNN algorithm to the problem utilizing "panoptic segmentation" as a loss strategy for minimization. In panoptic segmentation, we both simultaneously label hits as belonging to a type of object (i.e. Pileup/electron/photon/....) and we cluster the hits to a specific set of objects. The labelling of objects is often referred to as "instance segmentation", and the clustering of objects to distinct objects is referred to as "semantic segmentation."

The process of hadronic calorimetry – specifically, the determination of hadron energy—presents a parallel to the challenge of panoptic segmentation. The task involves separating the energy deposits, attributed to both primary and secondary interactions from a hadron-induced particle shower, from extraneous detector noise and unrelated events—much like delineating objects of interest from irrelevant components within an image (instance segmentation). The complexity further escalates when we aim to separate particle showers that overlap, a scenario mirroring the segmentation of an image with objects of varying properties (semantic segmenation). Therefore, techniques from panoptic segmentation, which simultaneously tackles semantic segmentation and instance segmentation to delineate every pixel in an image, can be applied in this context.

To solve this panoptic segmentation problem, we implement an adapted structure of SPVCNN along with a modified version of the object condensation loss [114]. Object condensation is a physics motivated loss strategy that aims to produce the cluster assignments, and cluster locations from the neural network output. We apply this approach to HCAL and HGCAL datasets, comparing the performance of SPVCNN to a graph neural network (GNN) and the existing particle flow HCAL clustering method. Our findings indicate that SPVCNN can provide efficient and accurate calorimetry solutions, particularly for high level trigger (HLT) applications with latency on the order of milliseconds.

Furthermore, our results demonstrate, for the first time, the integration of hadronic

calorimeter depth profiles as core information of the CMS Hcal calorimeter clustering. Previous rule based clustering algorithms did not use the depth profile as an additional means of cluster segmentation, the addition of depth profile provides an additional handle to further enhance the overall clustering performance.

In the following sections, we present a short background on particle physics, the LHC, the CMS experiment, and hadronic calorimetry, followed by an overview of the datasets and methodology used in this study. We then discuss the results of our analysis, comparing the performance of SPVCNN with existing methods, and conclude with the implications of our findings for future research and applications in high-energy physics.

Background

The CMS Detector and Hadronic Calorimeters

The CMS experiment is one of the four major experiments at the LHC. It is a general-purpose detector designed to investigate a wide range of physics phenomena. The CMS detector [115] consists of various sub-detectors arranged as concentric cylinders. Proceeding radially outward from the interaction point, a particle would first encounter an all-silicon tracking system, a lead-tungstate crystal electromagnetic calorimeter, a brass and scintillator hadronic calorimeter, a (non-sensitive) 3.8 T solenoidal magnet, and finally muon detectors embedded in the solenoid flux-return yoke. Information from these detector components is combined to identify and measure the properties of particles produced in collisions [116, 117, 118, 112].

In particular, the Hadron Calorimeter (HCAL) consists of four components: the HCAL Barrel (HB), the HCAL Endcap (HE), the HCAL Outer (HO), and the HCAL Forward (HF) [119, 120] (see Figure 4.1). In the HCAL clustering study presented here, only the HB and HE will be considered. The HB approximately covers the region $|\eta| < 1.3$, while the HE approximately covers the region $1.3 < |\eta| < 3$. Both the HB and HE are sampling calorimeters with brass absorber and plastic scintillator tiles. As hadronic particles enter the

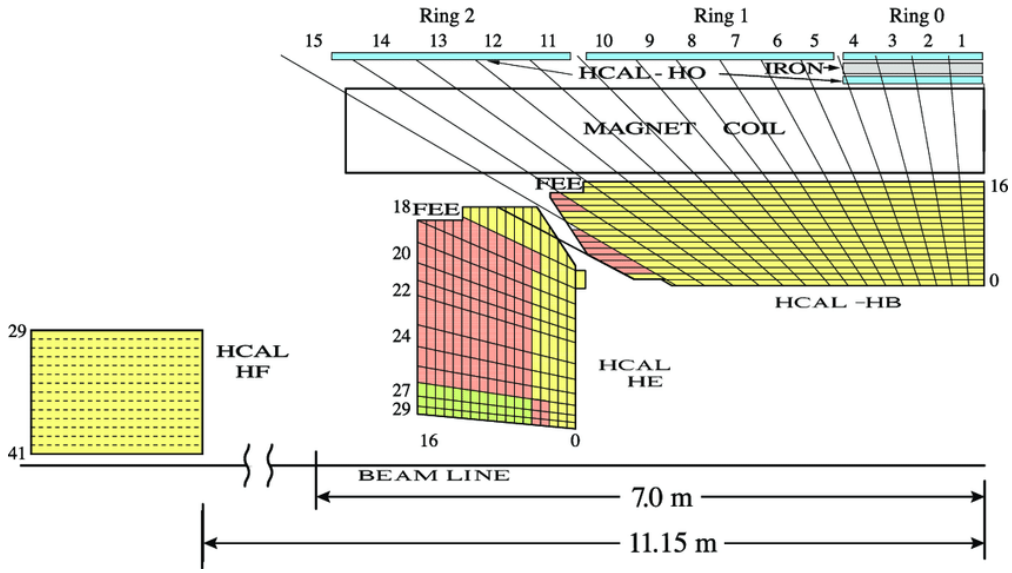

Figure 4.1: An illustration of the CMS HCAL (Hadron Calorimeter) components, including the HCAL Barrel (HB, $|\eta| < 1.3$), HCAL Endcap (HE, $1.3 < |\eta| < 3.0$), HCAL Forward (HF), and HCAL Outer (HO) [10]. In this study, only the HCAL Barrel (HB) and HCAL Endcap (HE) regions are considered.

HCAL, their nuclear interactions with the absorber material produces secondary particles and lead to the development of a shower. The ionized particles then interact in the plastic scintillator which is read out to provide energy and timing information for triggering and downstream reconstruction [119]. For the region $|\eta| < 1.74$, individual HCAL cells have size 0.087×0.087 in $\eta \times \phi$. For higher pseudorapidities, cell size increases as a function of $|\eta|$. Physics observables, such as the measurements of jets and an imbalance of energy in the transverse place (or missing transverse energy), are highly sensitive to the performance of HCAL. In particular, HCAL dominates resolution at high energy scales. During the Long Shutdown 2 (LS3) of the LHC, depth segmentation information was added to the HB and HE, with a goal of mitigating radiation damage and allowing for better calibration of depth-dependent effects [121, 122]. The HCAL megatiles, which are stacks of absorber/scintillator tiles, are integrated into depths. Currently, there are 7 depth in HE and 4 depths in HB.

The High-Granularity Calorimeter (HGCAL) will be an upgrade to the current endcap calorimeter [123], which encompasses both an electromagnetic and hadronic calorimeter. It will be installed prior to the start of HL-LHC runs and is designed to preserve energy resolution for the full planned integrated luminosity of 3000 fb^{-}1. It is a 47-layer sampling calorimeter that has a depth of about 10 hadronic interaction lengths. The first 26 layers (1.3 hadronic interaction lengths) comprise the primary electromagnetic component of the HGCAL constituting about 25 radiation lengths. The hadronic component has 12 fine sampling layer, with remaining layers having a larger fraction of absorber.

Current Calorimeter Clustering Approach

CMS employs the particle flow (PF) algorithm to reconstruct individual long-lived particles that interact with the detector [112]. The ATLAS experiment [124], also employs a similar PF algorithm [125]. A component of these PF algorithms is matching charged particle trajectories (reconstructed with tracker information) to calorimeter clusters. If a trajectory can be extrapolated into a cluster, and the measured energy of the trajectory and cluster are within certain resolution-related tolerances, then the cluster and trajectory are linked.

Calorimeter clusters that are not matched to trajectories are considered to have come from neutral particles.

In CMS, the calorimeter clusters themselves are created through a topological clustering-based scheme. This algorithm proceeds by first finding candidate "seed" calorimeter cells with energy above a set threshold that do not share a side with a higher energy seed. Topological clusters then expand through adjacent energy deposits. The next step of the algorithm splits topological clusters into multiple smaller clusters if it contains multiple seeds. Energy in a cell can be shared between multiple clusters, with sharing determined in an iterative algorithm that characterizes calorimeter showers as overlapping Gaussian distributions. The last step of the current CMS HCAL clustering algorithm combines clusters from different depths based on distance to form "multidepth" clusters. The depth information is only used in this last step, *after* clustering is performed for each depth independently.

The default PF clustering algorithm is performed in both the high-level trigger (HLT) and offline processing components of the CMS data processing workflow. At HLT-level, calorimeter cells considered must have energy of 0.8 GeV or higher to be considered in the algorithm. This threshold decreases per-event processing time, but causes worse response and resolution performance. In the offline version of the algorithm, where processing time is less important, the energy threshold for cells' consideration is depth dependent. For the four layers of HB, the thresholds are (from in-to-out) 0.1, 0.2, 0.3, and 0.3 GeVm and for the 7 layers of the HE, the threshold are 0.1, 0.2, 0.2, 0.2, 0.2, 0.2, and 0.2 GeV.

In recent years, machine learning techniques, such as graph neural networks (GNN), have been applied to calorimetry tasks to improve performance and computational efficiency [18]. While these methods have shown promising results, their potential may be limited compared to sparse neural networks, such as the SPVCNN, which can efficiently process sparse data and are more easily accelerated using GPUs [126].

Sparse CNNs and SPVCNN

Convolutional Neural Networks (CNNs), despite their revolutionary contributions to the field of computer vision, are less efficient when dealing with sparse data, a form common in various real-world applications including hadronic calorimetry. Sparse Convolutional Neural Networks (Sparse CNNs) were developed to surmount this limitation. By focusing on non-zero or "active" elements in the input data, they minimize computational complexity and memory requirements. This is achieved through specialized sparse convolution operations and data structures, such as the hash table, which enable efficient indexing and processing of the sparse data.

One specific type of sparse CNN, known as Sparse Point-Voxel Convolutional Neural Networks (SPVCNN), is tailored for handling 3D point cloud data [113]. The 3D space in an SPVCNN is partitioned into a regular grid of voxels, with each non-empty voxel linked to a set of points. This fusion of voxel-based and point-based methods facilitates efficient processing of sparse data while preserving the geometric intricacies of the point cloud.

In the architecture of an SPVCNN, there are several layers comprising sparse convolutional layers, pooling layers, and fully connected layers. The convolutional layers apply filters to the input, capturing local patterns and features. Pooling layers then reduce the spatial dimensions, paving the way for hierarchical representation learning. Finally, fully connected layers aggregate the learned features and yield the desired output.

This specialized architecture allows SPVCNN to efficiently process sparse data, making it well-suited for hadronic calorimetry tasks, where the data is often sparse and requires complex processing. Additionally, the SPVCNN can be easily accelerated using GPUs, which is crucial for meeting the computational demands of the HL-LHC and other high-energy physics experiments.

Metrics

Intersection over Union

Intersection over Union (IoU) is a common metric in the field of computer vision, especially for evaluating object detection and instance segmentation algorithms. Clustering shower deposits in our context can be seen as an instance segmentation problem, where the set of shower deposits from each primary particle must be identified and separated from others in the same volume. The IoU quantifies the overlap between two binary masks, typically the predicted mask and the ground truth mask. Mathematically, the IoU is defined as:

$$\text{IoU} = \frac{A \cap B}{A \cup B}$$

Where A and B are the areas of the predicted and ground truth masks, respectively. This is evaluated for each class and then the mean IoU (mIoU) is reported for all classes. In our case, we predict whether each measurement is noise or non-noise, so there are two classes.

Our problem of clustering shower deposits inherently aligns with the panoptic segmentation paradigm. Panoptic segmentation aims to provide a unified output for both instance-level categories (like individual objects) and semantic-level categories (like background regions or broader classes). In our context:

- **Instance Segmentation**: The identification and separation of individual shower deposits falls under instance segmentation. Each shower deposit can be seen as an individual "instance" that needs to be detected, classified, and delineated from other deposits in the same image or volume.

- **Semantic Segmentation**: Alongside the detection of individual shower deposits, our data also contains regions of noise or non-noise. Identifying and classifying these broader categories of pixels or voxels (i.e., distinguishing noise from non-noise regions) represents the semantic segmentation aspect of our problem.

Combining both of these tasks, where we need to distinguish individual shower deposits while also classifying regions as noise or non-noise, positions our problem squarely within the realm of panoptic segmentation.

Panoptic Quality

Panoptic Quality (PQ) is a metric designed to evaluate panoptic segmentation algorithms. The PQ is a combination of two terms: the segmentation quality (SQ) and the recognition quality (RQ).

- **Segmentation Quality** measures the average IoU between matched predicted segments and their corresponding ground truth segments. It captures the accuracy of the segmentation itself, disregarding label errors.

$$\mathrm{SQ} = \frac{1}{|\mathrm{TP}|} \sum_{(p,g)\in \mathrm{TP}} \mathrm{IoU}(p, g)$$

  where TP represents true positive matches between predicted segments p and ground truth segments g.

- **Recognition Quality** evaluates how well the predicted segments are classified, given they have been correctly detected.

$$\mathrm{RQ} = \frac{|\mathrm{TP}|}{|\mathrm{TP}| + \frac{1}{2}|\mathrm{FP}| + \frac{1}{2}|\mathrm{FN}|}$$

  where TP are true positive matches, FP are false positive predictions, and FN are false negatives.

Given SQ and RQ, the PQ can be formulated as:

$$\mathrm{PQ} = \mathrm{SQ} \times \mathrm{RQ}$$

For an ideal panoptic segmentation, the PQ would be 1, while a complete mismatch would result in a PQ of 0.

4.2 Datasets

In this study, we utilize three different datasets representing the calorimeter data from the CMS experiment at the LHC. These datasets include two HCAL datasets, one with pileup and one without pileup, and an HGCAL dataset.

4.2.1 HCAL Datasets

The HCAL datasets are based on simulated $t\bar{t}$ events generated using the Pythia8 event generator [127, 128] and processed through the CMS detector simulation based on the GEANT4 package [129, 130]. The pileup dataset includes additional minimum bias events overlaid on the primary $t\bar{t}$ events to simulate the effects of multiple proton-proton collisions occurring within the same bunch crossing. The number of pileup collisions overlaid on a particular $t\bar{t}$ event is chosen from a flat probability distribution between 55 and 75. Out-of-time pileup from the preceding 12 and succeeding 3 bunch crossings is based on a Poisson distribution with expected value set to the number of pileup collisions for a given event. The bunch spacing is set to 25 ns. In contrast, the no pileup dataset represents an idealized scenario without the added complexity of pileup effects.

Both HCAL datasets contain information about the simulated *measured* energy in each calorimeter cell and the corresponding cell positions. The datasets also contain a collection of each simulated particle in the hard-scatter $t\bar{t}$ process of the event, along with particle information, such as particle type, energy, and direction. Simulated particle information is not retained for pileup particles. The total simulated energy deposited by each hard-scatter particle in any calorimeter cell that it interacts with can be calculated using energy and position information for simulated particle-detector interaction points, which is retained in the datasets. Additionally, for each event, we have access to higher-level reconstructed objects, such as jets and PF particles, which are used for performance evaluation.

Using the truth information discussed above, it is possible to define a truth-level clustering definition, though it is not always clear how to account for effects such as overlapping calorimeter showers and energy overmeasurements due to pileup. To circumvent these issues, our first training target in the HCAL dataset case is to use SPVCNN to replicate the performance of the traditional, topological clustering. Traditional clustering has the benefit of guaranteed cluster continuity by construction, which is helpful in a semantic segmentation context. Further, the "seed" cells have a natural interpretation as candidate "alpha" points

in the SPVCNN context (see Section 4.3 **Training**).

The following features from each cell were used in training: the cell's x, y and z coordinates, the energy measured in the cell, the cell's eta and phi values, the time of the cell's energy measurement, and the cell's theta, R, and depth values. While the positional information is redundant, in practice it simplifies the learning process, avoiding the need to learn transformations between the differing coordinate systems. Of these features, eta, phi and depth are used as coordinates for the voxelization.

In this study, the impact of replacing traditional clustering with SPVCNN-based clustering can be examined by considering both PF particle-based observables and jet-based observables. PF particle based observables include the number of PF particles and their energy spectrum. Jet-based observables will focus on jet energy response and resolution [112]. For versions of SPVCNN trained to match traditional clustering, all of these observables should be similar when comparing full reconstruction with traditional clustering and full reconstruction with SPVCNN-based clustering.

4.2.2 HGCAL Dataset

The HGCAL dataset is based on single tau events, which are particles that decay predominantly into hadrons and neutrinos. This dataset focuses on single tau events without pileup to simplify the analysis and allow for a more straightforward comparison with the HCAL datasets. The HGCAL dataset is generated and processed using the same tools as the HCAL datasets, with the primary difference being the detector geometry and granularity. The ground truth definition for this dataset is discussed in [18]. Similar cell-level features to the HCAL dataset were used, with the omission of depth. For voxelization coordinates, the x, y and z positions were used.

4.2.3 Preprocessing

The raw datasets are preprocessed to extract relevant features for the SPVCNN, such as the cell energy deposits, noise levels, and geometrical information. For each dataset, we first

identify the active cells, which are the cells with non-zero energy deposits. These active cells are then transformed into a point cloud representation, where each point corresponds to a cell position and is associated with a feature vector containing the cell energy, local noise levels, and any additional cell-specific information. The coordinates are adjusted to have similar scale, such that unit voxels are appropriate. The features are standardized to have close to 0 mean and unit variance. Events that have a mean z coordinate less than 0 are reflected across the xy plane.

To prepare the data for the SPVCNN, we further process the point cloud representation by dividing the 3D space into a regular grid of voxels, with each non-empty voxel containing a set of points. This hybrid point-voxel representation enables efficient processing of the sparse data and preserves the geometrical details of the underlying point cloud.

After preprocessing, the resulting datasets are split into (0.8, 0.1, 0.1) training, validation, and test sets, ensuring that the events in each set are independent and representative of the overall dataset. These sets are then used to train, tune, and evaluate the SPVCNN and other algorithms for the hadronic calorimetry tasks.

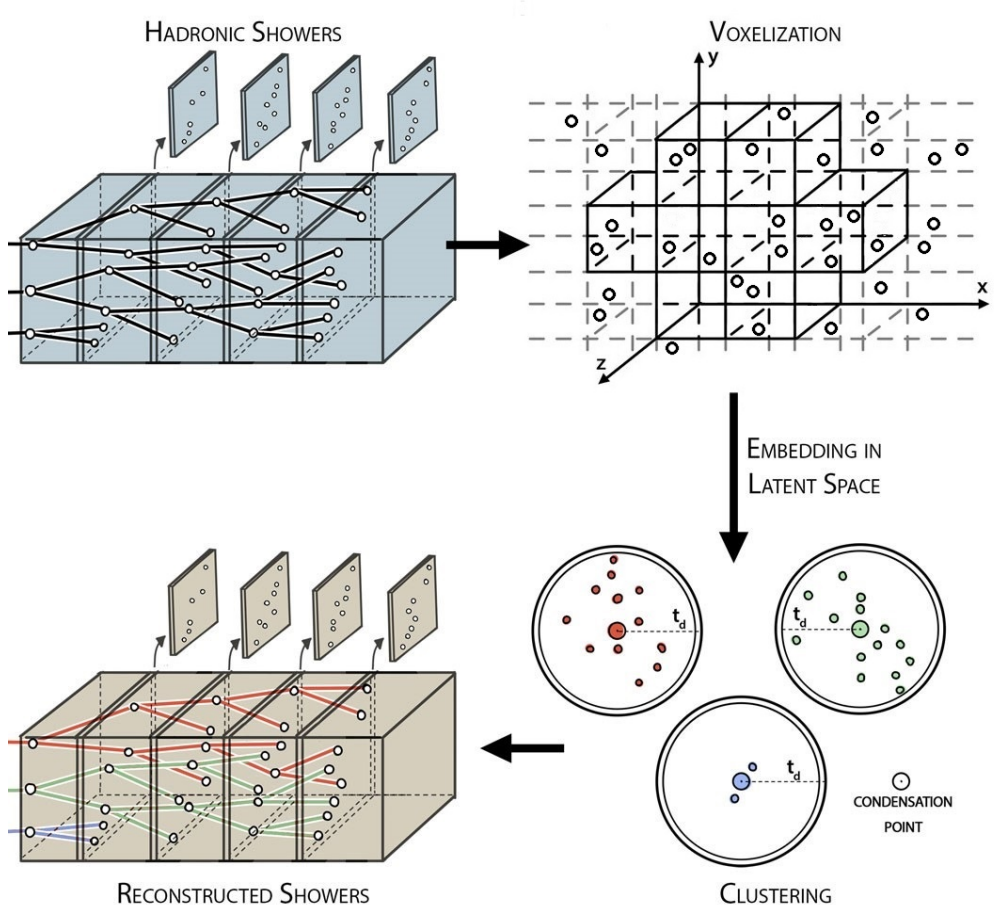

Figure 4.2: **Overview of the Proposed Approach**: This figure provides a step-by-step breakdown of the processes involved in our methodology.

Hadronic Showers: As incident particles penetrate the hadronic calorimeter, they initiate showers upon interacting with the passive material. These showered particles subsequently engage with active material regions, potentially producing several energy readings in a 3D spatial context for every primary particle.

Voxelization: Measurements are voxelized, mapping them to a regular 3D grid to enable convolutional processing.

Clustering: The voxelized data is input into SPVCNN. The output is a representation in a 5D embedded space for each point, along with a condensation score. A simple bounded nearest-neighbor clustering method (see 4.3) is then used to group the measurements.

Reconstructed Showers: Clustering assignments are used to reconstruct showers resulting from the original incident particles. This information is passed to downstream algorithms

4.3 Methodology

In this section, we detail the methodology used in this study, which involves the use of sparse point-voxel convolutional neural networks (SPVCNN) for hadronic calorimetry tasks. The specific steps we outline include the architectural design of our SPVCNN, the training process with a modified form of the object condensation loss, and the clustering methods used to make concrete cluster predictions. Each of these components plays a pivotal role in addressing the challenges inherent in hadronic calorimetry, providing a robust and efficient solution for processing calorimetry data in high-energy physics experiments.

Architectural Design

Our sparse point-voxel convolutional neural network (SPVCNN) is composed of a primary feature backbone and two specialized branches: the semantic and instance branches. The backbone, with its 20.8 million parameters, comprises a two-layer sparse convolutional stem and four stages of sparse point-voxel convolutions, modeled as per [113]. Its role is to distill 32-dimensional high-level features from the input data.

These high-level features are then concatenated with the input features, serving as the input to the semantic branch. The semantic branch (5.5K parameters) consists of four linear layers and discerns whether each cell is noise or non-noise.

The instance branch (12.4K parameters), on the other hand, carries out a dual role: it embeds each cell into a 5-dimensional space and also calculates an object condensation score, or 'beta', for each cell. This is achieved by first routing the backbone features through three linear layers, the output of which, called instance features, are then combined with the input coordinates. This merged data is then processed through two additional linear layers, culminating in a 6-dimensional vector. Here, the first dimension signifies the object condensation score, and the remaining dimensions represent the embedded space. This dual-role concept of the instance branch allows for simpler clustering during the prediction phase, as is discussed later.

Training

Our training strategy utilizes a modified version of the object condensation loss. Here, a charge 'q' is defined for each point as $q = \tanh^2(\beta) + q_{min}$, with β representing the sigmoid of the object condensation score predicted by the instance branch. An 'alpha' point is concurrently defined for each truth cluster. We investigated two strategies for defining this point: a) choosing the point with the highest predicted charge (q), and b) choosing the point with the highest energy. The latter approach aligns with traditional methods that focus on building clusters around high energy points. To further optimize the model's performance, we introduce a loss function that incorporates repulsive, attractive, and beta losses, weighted by the energy of the cells involved. This encourages the model to prioritize higher-energy cells, which have a substantial influence on the final performance.

Clustering

One advantage of the object condensation method is that it enables a straightforward way to predict clusters for use in subsequent stages of the analysis chain. Without the notion of condensation points, more complicated clustering techniques such as mean-shift clustering are often required, which can decrease throughput for dense environments. We commence by setting two thresholds: an object condensation threshold t_β, and a distance threshold t_d. Points with a predicted object condensation score greater than t_β are defined as condensation points. These points are then sorted according to their predicted object condensation scores. For each condensation point not yet allocated to a cluster, we group all unclustered points within a radius of t_d in the embedded space to form a new cluster. Points not within a radius of t_d of any condensation point are deemed as noise and subsequently discarded.

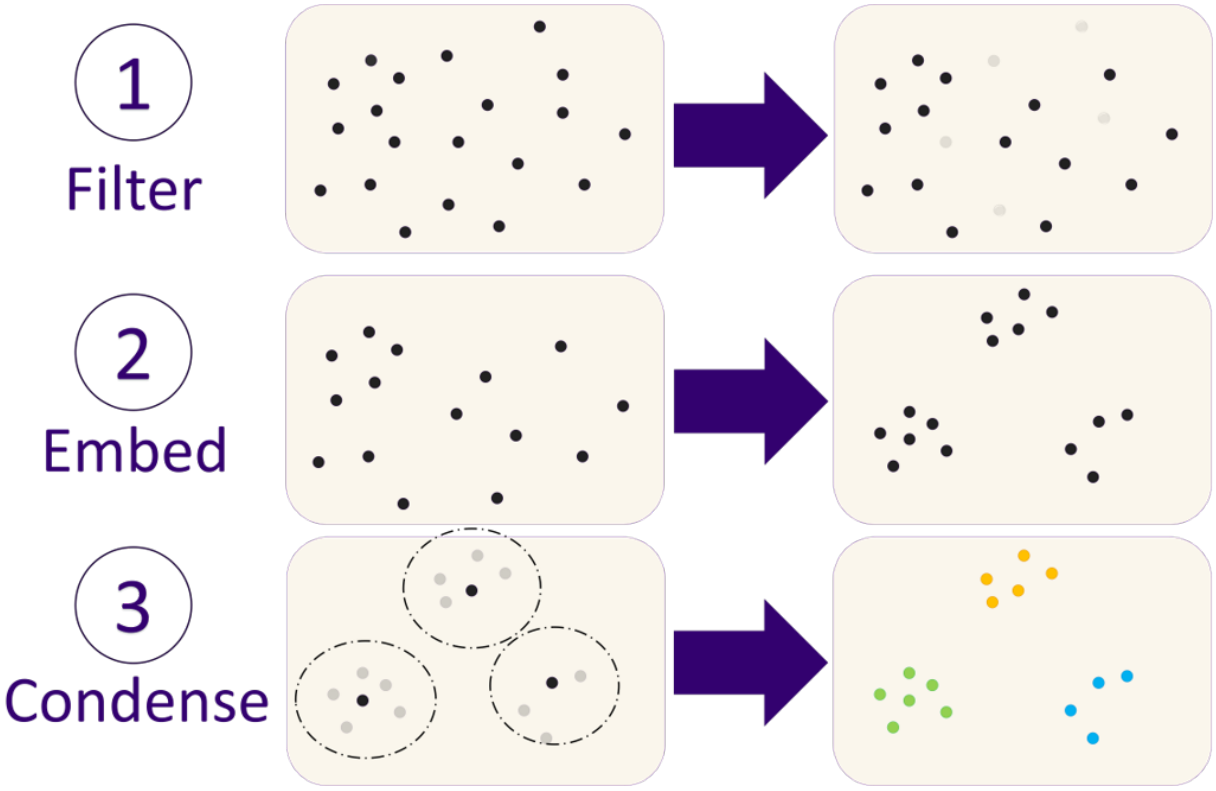

Figure 4.3: Illustration of our prediction process. Steps 1 and 2 involve the SPVCNN model, while step 3 is performed on the embeddings with a separate clustering algorithm, as described.

Integration with CMSSW

To enable HCAL clustering performance studies using downstream physics objects, such as jets, SPVCNN was integrated into the CMS software framework CMSSW [131] via the "Services for Optimized Network Inference on Coprocessors" (SONIC) approach [132]. In this approach, the SPVCNN model is hosted as a python custom backend model on an NVIDIA Triton Inference Server [133], and inferences are performed as a service. CPU-based "client" jobs request inference from the server via a gRPC call [134]. These client jobs run all components of the CMSSW workflow as normal, except that components of the code that typically execute clustering on local CPUs has been replaced with code that creates a tensor

with calorimeter cell information, sends that tensor to the SPVCNN server, retrieves the results, and finally creates cluster objects based on the labels that SPVCNN returns for each cell. The CMSSW framework uses Intel Threading Building Blocks [135] to implement task-based multithreading, allowing for asynchronous non-blocking calls to external resources, so latency increases due to client-to-server communication is not expected.

The Triton python backend operates similarly to the training forward pass, with the addition of clustering: once the calorimeter cell tensors are received in a batch, each tensor is quantized and loaded onto the GPU as a sparse tensor. These tensors are then collated into a single tensor, with the addition of a batch dimension. This tensor is then passed through the SPVCNN model. Finally, the resultant output is processed by the object condensation clustering algorithm, as described previously.

4.4 Results

HGCAL

The HGCAL detector is intended for the HL-LHC, as such it has yet to be installed in CMS and there are no conventional algorithms in place to process its output. Therefore, we restrict our performance analysis to the cluster level and compare to the GravNet GNN method, as described in [18]. An example of a typical event can be seen in Fig. 4.4.

Standard metrics for semantic and panoptic performance are Intersection over Union (IoU) and Panoptic Quality (PQ) [136]. IoU and PQ results for GravNet and SPVCNN are summarized in Table 4.1. It can be seen that SPVCNN achieves comparable or better results than GravNet. To facilitate fair comparison, we re-tuned GravNet for PQ, which is denoted by "GravNet (optimized)".

To better understand the physics performance, we compare the cluster-level energy reconstruction of SPVCNN for different simulated particles in Fig. 4.5. The results show good agreement between truth and reconstructed clusters for all particle types. Note that, since we do not compare to simulated energy, this is purely a measure of the clustering performance

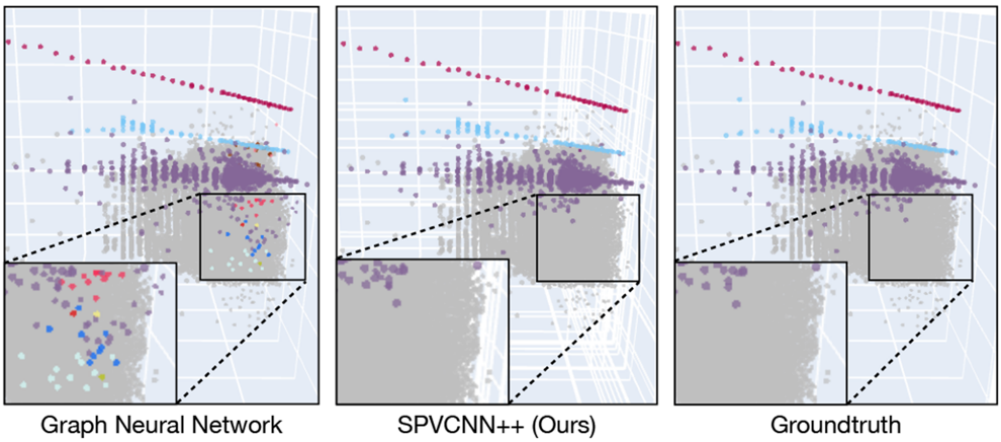

Figure 4.4: 3D visualization of HGCAL data. Each color represents a cluster. The left and middle panes show GNN and SPVCNN predictions, respectively, while the right pane shows the ground truth.

Table 4.1: IoU and PQ scores on HGCAL dataset.

Method	mIoU	SQ	RQ	PQ
GravNet	0.93	0.89	0.74	0.69
GravNet (optimized)	0.93	0.90	0.83	0.76
SPVCNN++	**0.98**	**0.92**	**0.85**	**0.80**

of the algorithm, and does not include resolution effects inherent to the detector.

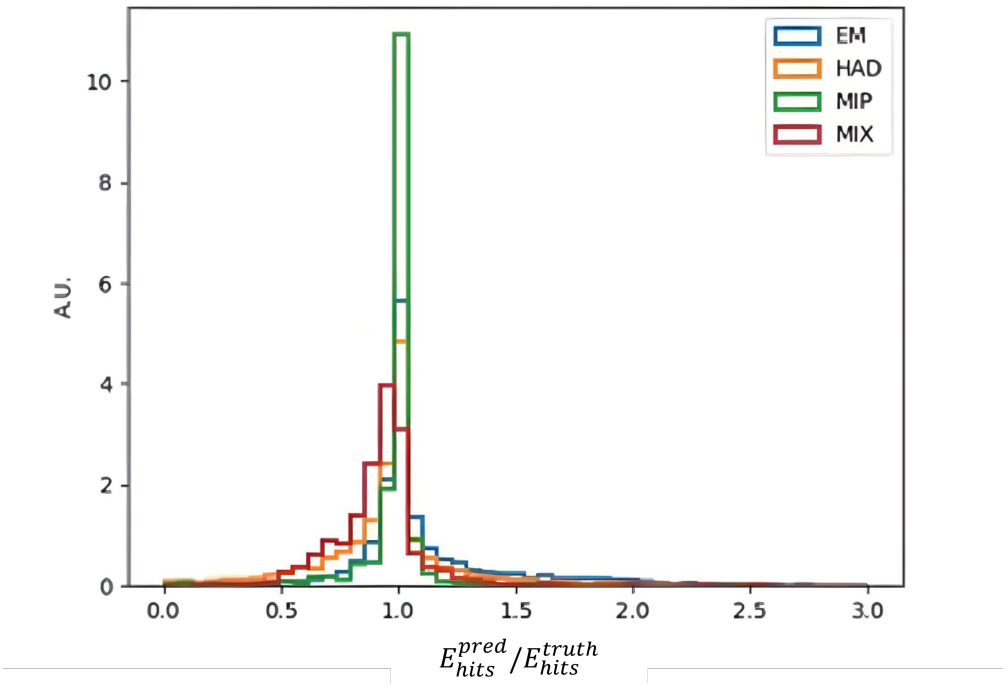

Figure 4.5: Ratio of predicted to true energy for each cluster predicted by SPVCNN on the HGCAL dataset. The energy of a predicted/true cluster is defined as the sum of the reconstructed energy of each hit within that cluster. The predicted clusters are separated into four categories, based on the particle type of the matched true cluster. EM corresponds to electromagnetic particles, HAD to hadronic particles, and MIP to muons. MIX is reserved for predicted clusters that were matched to several truth clusters of differing types.

HCAL

As HCAL is a current subdetector in CMS, it is a standard part of high-level object reconstruction in the CMS software toolchain. This enables us to perform studies of jet-level variables in addition to what we can do with HGCAL. Due to complexities in defining an effective truth definition and handling downstream corrections that are tuned for the current setup, we opted to train our system to replicate the traditional topological clustering as a first step. While the primary objective of this approach is not centered around significant accuracy improvements, it offers compelling advantages in terms of increased efficiency and ease of acceleration and integration with heterogeneous computing platforms.

To understand the impact of our clustering method on jets, we turn our attention to the AK4 and AK8 jets, commonly utilized in CMS. Define jargons AK4 and AK8 These jets are defined by the radius parameter, R, in the anti-k_{T} jet algorithm [137], where R equals 0.4 and 0.8 respectively. This parameter dictates the size of the jets, with larger values producing larger jets suited for capturing energy from widespread collisions or encompassing larger structures like fat jets from boosted objects. In the context of jet-level variables, the Jet Energy Resolution (JER) and Jet Energy Scale (JES) are central. The JER measures the precision of the energy estimation, while the JES gauges the difference between the jet's measured energy and the true energy of the corresponding particle, also referred to as the response of the jet. A response of 1 and resolution of 0 is ideal, but generally unachievable due to physical limitations.

Reconstructed jet performance is shown both for SPVCNN and PF-clustering in Fig. 4.6 for AK4 jets (top panel) and AK8 jets (bottom panel). The results are shown separately in datasets simulated with and without pileup.

An important observable that incorporates all subsystems at hadron colliders is missing transverse momentum $p_{\mathrm{T}}^{\mathrm{miss}}$ (or MET), computed as the negative vector sum of particle momenta in the transverse plane. The distribution of MET is important for inferring particles that carry away momentum from the collision but did not interact with the detector, such

as neutrinos or dark matter. The MET distribution is sensitive to mis-reconstruction of particles in the HCAL. To measure the effect of SPVCNN-based clustering on MET we consider a sample of Z bosons recoiling against a jet, where the Z boson decays to a pair of muons. The dataset is simulated using standard pileup conditions. The MET response and resolution are shown in Fig. 4.7.

It is clear from Fig. 4.6 that the performance of default clustering and SPVCNN-based clustering (with $t_\beta = 0.1$ and $t_d = 0.7$) are essentially similar at jet-level in the case of no pileup, with differences in response and resolution at the level of 1%. Performance of SPVCNN in the case *with* pileup, also shown in Fig. 4.6 is still similar to the default clustering case, though there are some differences worth noting. When taking $t_\beta = 0.1$ and $t_d = 0.7$ as the SPVCNN defaults for both AK4 and AK8 jets, reconstructed jets after SPVCNN-based clustering have response closer to 1 than the default clustering, except for lower-p_T AK4 jets, where the SPVCNN response is higher than 1. For AK4 jets, slightly worse resolution is observed for SPVCNN-based clustering at low p_T, whereas for AK8 jets, the SPVCNN-based is similar to the default, and appears better at higher p_T. The performance can be adjusted by tuning the t_β and t_d values. For example, some extra t_d points are shown, though typically, achieving better resolution results in worse response, and vice versa.

The default PF clustering algorithm executed in the CMS offline reconstruction step takes about 0.112 seconds of CPU time per event. The CPU-only component of the SONIC-based implementation of SPVCNN in this study takes about 0.007 seconds per event, which is time spent compiling calorimeter cell information into tensors for the Triton backend, and time spent creating cluster-objects in the context of CMSSW after retrieving SPVCNN results. Thus, based only on CPU-resource usage, SPVCNN is 16 times faster than generic PF clustering. The timing performance of SPVCNN on a GPU is studied using the "perf_client" provided by Triton Inference Server [138]. It was possible to saturate the resources of a single Tesla T4 GPU provided by the LHC Physics Center (LPC) at Fermilab by loading multiple model instances of SPVCNN onto a server and enabling inference request batching. Here, batching refers to the fact that the data tensors from multiple inference requests can

be combined into a single tensor. This typically increases computational efficiency on a GPU. By repeatedly sending inference requests with realistic data through the perf_client, a maximum throughput of 420 inferences per second was measured by using a batch size of 256. This corresponds to an on-GPU latency of 0.0024 seconds per event. Combined with the CPU-based component of the algorithm, this is about 0.01 seconds per event of resource utilization to run the full HCAL clustering chain with SPVCNN. The results of a timing scan against batch size are shown in Fig. 4.8. The left plot in the figure shows the average processing time per inference request, and the right figure shows the total throughput in terms of events processed per second. The scans were performed with a request concurrency of 7, which was found to maximize throughput, and used data from simulated events as input.

Fully processing a single top pair event with typical pileup can take more than 10 seconds on a single CPU thread. Given this per event latency, a single GPU hosting SPVCNN-based HCAL clustering should be able to provide inferences for over 1000 CPU cores simultaneously processing such events.

4.5 Conclusion

This study demonstrates the potential for utilizing sparse point-voxel convolutional neural networks (SPVCNN) to handle the complex task of hadronic calorimetry in the context of the Large Hadron Collider (LHC) and the upcoming High-Luminosity Large Hadron Collider (HL-LHC). Through our application of SPVCNN, we have shown it is possible to process calorimetry tasks effectively and efficiently.

Our approach has shown promising results, delivering performance that compares favorably with established methods like generic topological cluster-based techniques and graph neural network (GNN) methods in the HCAL and HGCAL environments. Furthermore, our methodology is designed to efficiently leverage GPU acceleration, which could be crucial in handling the increased data processing demands anticipated with the advent of the High-Luminosity Large Hadron Collider (HL-LHC) and shows its potential as a tool for high level

trigger (HLT) applications.

Future research could aim to refine the SPVCNN methodology and extend its application to other types of calorimetry tasks and other detectors. We anticipate that more targeted optimizations, e.g. model pruning and other model size reduction schemes, could further improve throughput and reduce latency. Additional comparative studies against existing methods would also help to further elucidate the potential benefits of this approach. Finally, we expect that the development of a truth-based training target, rather than replicating the conventional approach, could lead to improvements in reconstruction performance for HCAL. In general, in light of the ongoing advancements in machine learning technologies and the ever-increasing computational demands of high-energy physics research, continued exploration of the synergies between these two domains remains a promising avenue for future work.

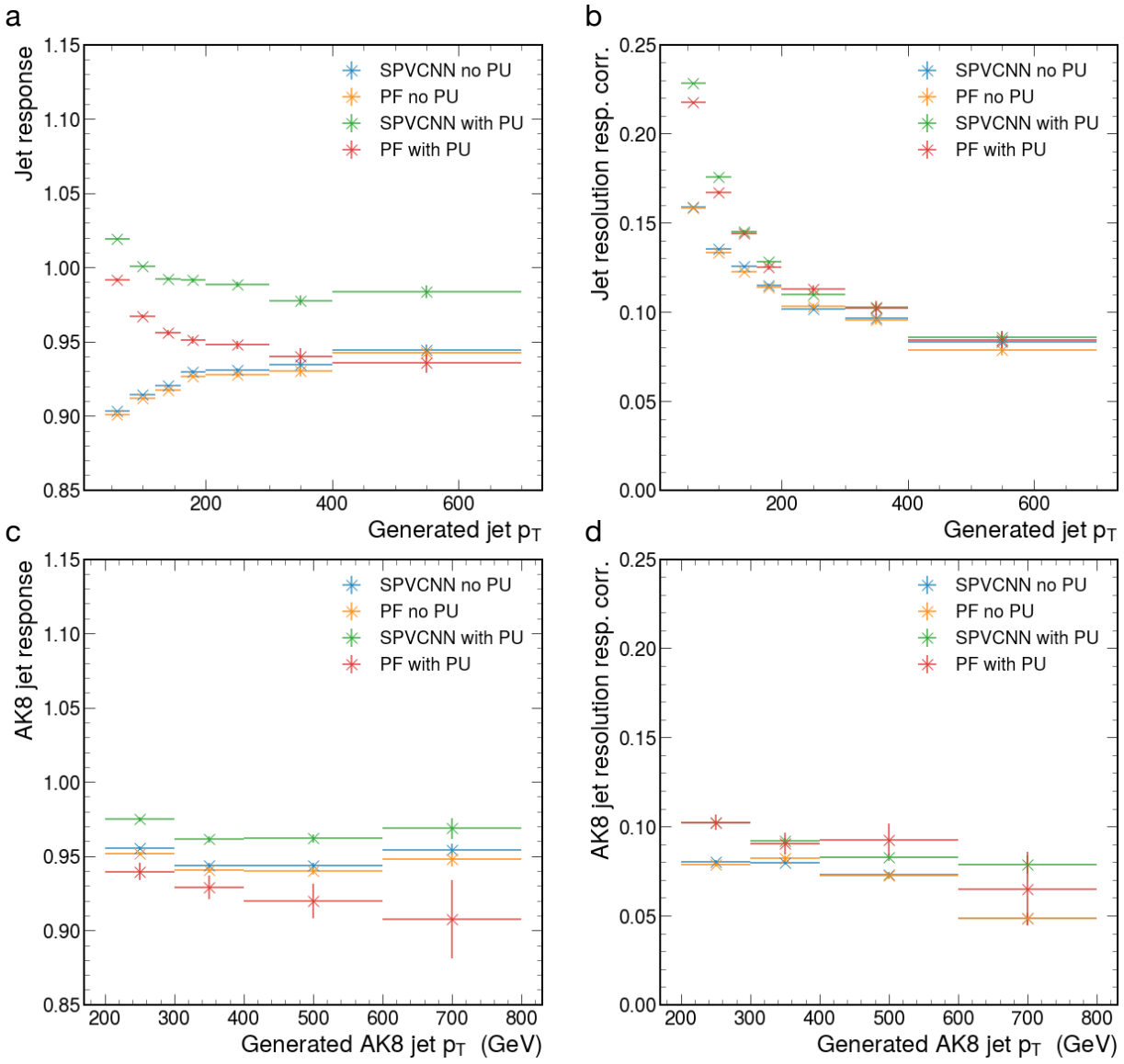

Figure 4.6: Top panel: response (a) and response-corrected resolution (b) for AK4 jets. Bottom panel: response (c) and response-corrected resolution (d) for AK8 jets.

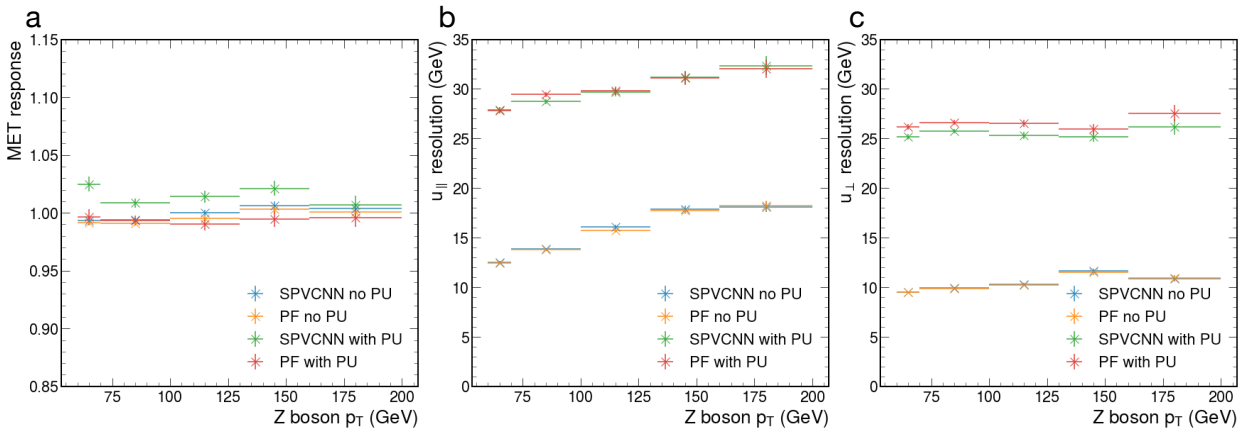

Figure 4.7: (a) Missing transverse energy response, (b) parallel recoil resolution, and (c) perpendicular recoil resolution.

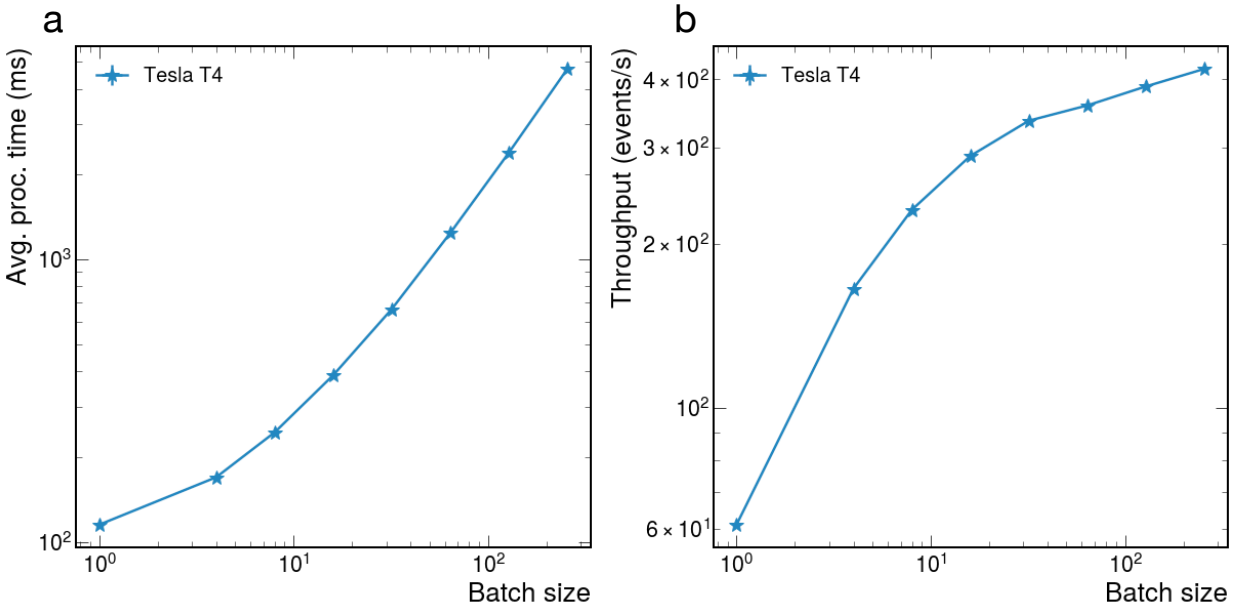

Figure 4.8: The average processing time per inference request (left) and throughput in events per second (right) as a function of batch size.

Chapter 5

CONCLUSION

This thesis has delineated the vital role of deep learning in revolutionizing data analysis within the realm of particle physics. The introduction of more powerful particle accelerators, capable of probing the intricacies of the universe at unprecedented scales, brings forth both opportunities for groundbreaking discoveries and the challenge of managing vast and complex datasets. The studies presented in this thesis highlight the potential of deep learning to address these challenges, illustrating its transformative impact in understanding phenomena like dark matter and managing computational loads in future high-luminosity experiments.

The first study presented within this thesis demonstrated the efficacy of Graph Neural Networks (GNNs) in particle tracking within the Exa.TrkX project. This approach not only paralleled the efficiency and purity of existing algorithms but also showcased the potential benefits of GPU acceleration, a pivotal aspect in managing the increased data throughput in modern experiments.

The second study focused on DeepCalo, a sophisticated multi-modal deep learning model integrating FiLM layers, dense layers, and convolutional layers. This study extended beyond a mere introduction of DeepCalo, emphasizing its novel implementation on Field-Programmable Gate Arrays (FPGAs) for low-latency applications. Such a development is critical for real-time data processing in particle physics experiments, where latency is a significant factor. The successful deployment of DeepCalo on FPGAs marks a significant stride towards more efficient and expedited data analysis, addressing a central computational bottleneck in High Energy Physics.

In the third study, the application of Sparse Point-Voxel Convolutional Neural Networks (SPVCNN) in clustering energy deposits from hadronic showers was explored. This study ac-

centuated the model's competence in effectively processing complex, high-dimensional data, demonstrating its comparative efficiency to traditional methods and its suitability for high-level trigger applications in environments akin to the High-Luminosity Large Hadron Collider.

These collective findings illuminate the transformative potential of deep learning in particle physics. The efficiency of these models in managing complex data, combined with their computational speed and adaptability, positions deep learning as an essential tool in the future of particle physics research. Particularly, these advancements are anticipated to play a crucial role in tackling the computational challenges associated with the next generation of particle accelerators.

Looking ahead, the evolution of deep learning in particle physics is poised to address some of the most compelling computational challenges posed by next-generation particle accelerators. These accelerators, like the High-Luminosity Large Hadron Collider, are expected to generate unprecedented volumes of data, presenting significant challenges in data processing and latency. The methodologies and insights developed in this thesis demonstrate how deep learning can effectively manage these challenges, enhancing the efficiency and scalability of data analysis. This advancement is not just a computational achievement; it is a crucial enabler for probing deeper into the mysteries of the universe. By alleviating the computational bottlenecks, deep learning paves the way for exploring new physics phenomena, including the elusive nature of dark matter. The potential for these technologies to facilitate groundbreaking discoveries and enrich our understanding of fundamental physics is immense. In conclusion, the integration of deep learning into particle physics is a significant technological stride, marking the beginning of a new era in which enhanced data analysis capabilities could lead to transformative discoveries in one of the most challenging domains of scientific exploration.

BIBLIOGRAPHY

[1] Cush. Standard model of elementary particles. `https://en.wikipedia.org/wiki/File:Standard_Model_of_Elementary_Particles.svg`, 2022. Own work by uploader, PBS NOVA, Fermilab, Office of Science, United States Department of Energy, Particle Data Group.

[2] Maximilien Brice. Aerial View of the CERN taken in 2008. Own work by uploader., 2008.

[3] CERN. Facts and Figures about LHC. `https://home.cern/resources/faqs/facts-and-figures-about-lhc`, 2023. Accessed on 12/14/2023.

[4] ATLAS Collaboration et al. The atlas experiment at the cern large hadron collider. *Journal of Instrumentation*, 3(08):S08003–S08003, Aug 2008.

[5] Joao Pequenao and Paul Schaffner. How ATLAS detects particles: diagram of particle paths in the detector. Own work by uploader., 2013.

[6] NickDiCicco. Message passing neural network. `https://en.wikipedia.org/wiki/File:Message_Passing_Neural_Network.png`, 2022. Own work by uploader.

[7] Sabrina Amrouche et al. The Tracking Machine Learning challenge : Accuracy phase. 4 2019.

[8] CMS Collaboration. CMS Tracking POG Performance Plots For 2017 with PhaseI pixel detector, 2017.

[9] Energy reconstruction of electrons and photons using convolutional neural networks, master's thesis (cand.scient.), 2019.

[10] S. Abdullin, V. Abramov, B. Acharya, Maurice Adams, Nural Akchurin, Ugur Akgun, E. Anderson, Gueorgui Antchev, A. Ayan, Sezgin Aydin, M. Baarmand, D. Baden, Sud Banerjee, Sun Banerjee, R. Bard, V. Barnes, Hana Bawa, G. Baiatian, G. Bencze, and The Collaboration. Design, performance, and calibration of cms hadron-barrel calorimeter wedges. *European Physical Journal C*, 55:159–171, 01 2008.

[11] F. Englert and R. Brout. *Phys. Rev. Lett.*, 13:321, 1964.

[12] P.W. Higgs. *Phys. Lett.*, 12:132, 1964.

[13] P.W. Higgs. *Phys. Rev. Lett.*, 13:508, 1964.

[14] G.S. Guralnik, C.R. Hagen, and T.W.B. Kibble. *Phys. Rev. Lett.*, 13:585, 1964.

[15] P.W. Higgs. *Phys. Rev.*, 145:1156, 1966.

[16] CMS Collaboration. Observation of a new boson at a mass of 125 gev with the cms experiment at the lhc. *Phys. Lett. B*, 716:30, 2012.

[17] ATLAS Collaboration. Observation of a new particle in the search for the standard model higgs boson with the atlas detector at the lhc. *Phys. Lett. B*, 716:1, 2012.

[18] Saptaparna Bhattacharya, Nadezda Chernyavskaya, Saranya Ghosh, Lindsey Gray, Jan Kieseler, Thomas Klijnsma, Kenneth Long, Raheel Nawaz, Kevin Pedro, Maurizio Pierini, Gauri Pradhan, Shah Rukh Qasim, Oleksander Viazlo, and Philipp Zehetner. GNN-based end-to-end reconstruction in the CMS Phase 2 High-Granularity Calorimeter. Technical Report 1, CERN, Geneva, 2023. 5 pages, 5 figures, proceedings for the 20th International Workshop on Advanced Computing and Analysis Techniques in Physics Research.

[19] The ATLAS TDAQ Collaboration. The atlas data acquisition and high level trigger system. *Journal of Instrumentation*, 11(06):P06008, jun 2016.

[20] I. Béjar Alonso, O. Brüning, P. Fessia, M. Lamont, L. Rossi, L. Tavian, and M. Zerlauth. High Luminosity Large Hadron Collider HL-LHC Technical Design Report. *CERN Yellow Report*, 10, 2020.

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

[22] ATLAS Collaboration. Performance of the ATLAS Track Reconstruction Algorithms in Dense Environments in LHC Run 2. *Eur. Phys. J. C*, 77(10):673, 2017.

[23] Serguei Chatrchyan et al. Description and performance of track and primary-vertex reconstruction with the CMS tracker. *JINST*, 9(10):P10009, 2014.

[24] Richard O. Duda and Peter E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15(1):11–15, January 1972.

[25] Joakim Gradin, M. Mårtensson, and R. Brenner. Comparison of two hardware-based hit filtering methods for trackers in high-pileup environments. *JINST*, 13(04):P04019, 2018.

[26] Daniel Funke, Thomas Hauth, V. Innocente, G. Quast, P. Sanders, and D. Schieferdecker. Parallel track reconstruction in CMS using the cellular automaton approach. *J. Phys. Conf. Ser.*, 513:052010, 2014.

[27] David Rohr, Sergey Gorbunov, Marten Ole Schmidt, and Ruben Shahoyan. GPU-based Online Track Reconstruction for the ALICE TPC in Run 3 with Continuous Read-Out. *EPJ Web Conf.*, 214:01050, 2019.

[28] ATLAS Collaboration. Fast Track Reconstruction for HL-LHC. Technical Report ATL-PHYS-PUB-2019-041, CERN, Geneva, Oct 2019.

[29] HEP.TrkX. HEP advanced tracking algorithms with cross-cutting applications, 2016.

[30] Steven Farrell et al. Novel deep learning methods for track reconstruction. In *4th International Workshop Connecting The Dots 2018 (CTD2018) Seattle, Washington, USA, March 20-22, 2018*, 2018.

[31] Exa.TrkX. HEP advanced tracking algorithms at the exascale, 2019.

[32] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning: Going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[33] Nicholas Choma et al. Track Seeding and Labelling with Embedded-space Graph Neural Networks. 6 2020.

[34] Steven Farrell et al. The HEP.TrkX Project: deep neural networks for HL-LHC online and offline tracking. In *Proceedings, Connecting The Dots / Intelligent Tracker (CTD/WIT 2017): Orsay, France, March 6-9, 2017*, volume 150, page 00003, 2017.

[35] Valerio Bertacchi. DeepCore: Convolutional Neural Network for high p_T jet tracking. 10 2019.

[36] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[37] Aristeidis Tsaris, Dustin Anderson, Josh Bendavid, Paolo Calafiura, Giuseppe Cerati, Julien Esseiva, Steven Farrell, Lindsey Gray, Keshav Kapoor, Jim Kowalkowski, Mayur Mudigonda, Prabhat, Panagiotis Spentzouris, Maria Spiropoulou, Jean-Roch Vlimant, Stephan Zheng, and Daniel Zurawski. The HEP.TrkX project: Deep learning for particle tracking. *Journal of Physics: Conference Series*, 1085:042023, sep 2018.

[38] Sabrina Amrouche et al. The Tracking Machine Learning challenge : Accuracy phase. 4 2019.

[39] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014.

[40] Catherine Biscarat, Sylvain Caillou, Charline Rougier, Jan Stark, and Jad Zahreddine. Towards a realistic track reconstruction algorithm based on graph neural networks for the hl-lhc. 2021.

[41] Xiangyang Ju et al. Graph Neural Networks for Particle Reconstruction in High Energy Physics detectors. In *33rd Annual Conference on Neural Information Processing Systems*, 3 2020.

[42] Tung D. Le, Haruki Imai, Yasushi Negishi, and Kiyokuni Kawachiya. Tflms: Large model support in tensorflow by graph rewriting. 2019.

[43] Joosep Pata, Javier Duarte, Jean-Roch Vlimant, Maurizio Pierini, and Maria Spiropulu. MLPF: Efficient machine-learned particle-flow reconstruction using graph neural networks. 1 2021.

[44] V Hewes, Adam Aurisano, Giuseppe Cerati, Jim Kowalkowski, Claire Lee, Wei keng Liao, Alexandra Day, Angrit Agrawal, Maria Spiropulu, Jean-Roch Vlimant, Lindsey Gray, Thomas Klijnsma, Paolo Calafiura, Sean Conlon, Steve Farrell, Xiangyang Ju, and Daniel Murnane. Graph neural network for object reconstruction in liquid argon time projection chambers, 2021.

[45] Aneesh Heintz et al. Accelerated Charged Particle Tracking with Graph Neural Networks on FPGAs. In *34th Conference on Neural Information Processing Systems*, 11 2020.

[46] Patrick J. Fox, Shangqing Huang, Joshua Isaacson, Xiangyang Ju, and Benjamin Nachman. Beyond 4d tracking: Using cluster shapes for track seeding. 2020.

[47] Sabrina Amrouche, Moritz Kiehn, Tobias Golling, and Andreas Salzburger. Hashing and metric learning for charged particle tracking. 2021.

[48] Wen Guan, Gabriel Perdue, Arthur Pesah, Maria Schuld, Koji Terashi, Sofia Vallecorsa, and Jean-Roch Vlimant. Quantum Machine Learning in High Energy Physics. 5 2020.

[49] Cenk Tüysüz, Kristiane Novotny, Carla Rieger, Federico Carminati, Bilge Demirköz, Daniel Dobos, Fabio Fracas, Karolos Potamianos, Sofia Vallecorsa, and Jean-Roch Vlimant. Performance of Particle Tracking Using a Quantum Graph Neural Network. 12 2020.

[50] Peter W. Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and Koray Kavukcuoglu. Interaction networks for learning about objects, relations and physics. *CoRR*, abs/1612.00222, 2016.

[51] Davide Chicco. *Siamese Neural Networks: An Overview*, pages 73–94. Springer US, New York, NY, 2021.

[52] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. 2017.

[53] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3D Deep Learning with PyTorch3D. 2020.

[54] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks, 2017.

[55] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[56] ATLAS Collaboration. Technical Design Report for the ATLAS Inner Tracker Pixel Detector. Technical Report CERN-LHCC-2017-021. ATLAS-TDR-030, CERN, Geneva, Sep 2017.

[57] Atlas Collaboration. Technical Design Report for the ATLAS Inner Tracker Pixel Detector. Technical Report ATLAS-TDR-030, CERN, Geneva, September 2017.

[58] ATLAS Collaboration. Expected Tracking Performance of the ATLAS Inner Tracker at the HL-LHC. Technical Report ATL-PHYS-PUB-2019-014, CERN, Geneva, Mar 2019.

[59] Ryosuke Okuta, Yuya Unno, Daisuke Nishino, Shohei Hido, and Crissman. Cupy : A numpy-compatible library for nvidia gpu calculations. In *31st Conference on Neural Information Processing Systems (NIPS 2017)*, 2017. `http://learningsys.org/nips17/assets/papers/paper_16.pdf`.

[60] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[61] CuGraph. `https://github.com/rapidsai/cugraph`, 2020. Accessed 2021-03-01.

[62] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, pages 226–231. AAAI Press, 1996.

[63] Farah Fahim, Benjamin Hawks, Christian Herwig, James Hirschauer, Sergo Jindariani, Nhan Tran, Luca P. Carloni, Giuseppe Di Guglielmo, Philip Harris, Jeffrey Krupa, Dylan Rankin, Manuel Blanco Valentin, Josiah Hester, Yingyi Luo, John Mamish, Seda Orgrenci-Memik, Thea Aarrestad, Hamza Javed, Vladimir Loncar, Maurizio Pierini, Adrian Alan Pol, Sioni Summers, Javier Duarte, Scott Hauck, Shih-Chieh Hsu, Jennifer Ngadiuba, Mia Liu, Duc Hoang, Edward Kreinar, and Zhenbin Wu. hls4ml: An open-source codesign workflow to empower scientific low-power machine learning devices, 2021.

[64] Jeffrey Krupa, Kelvin Lin, Maria Acosta Flechas, Jack Dinsmore, Javier Duarte, Philip Harris, Scott Hauck, Burt Holzman, Shih-Chieh Hsu, Thomas Klijnsma, and et al. Gpu coprocessors as a service for deep learning inference in high energy physics. *Machine Learning: Science and Technology*, 2(3):035005, Apr 2021.

[65] Valentin Kuznetsov, Luca Giommi, and Daniele Bonacorsi. Mlaas4hep: Machine learning as a service for hep, 2020.

[66] ATLAS Collaboration. The ATLAS simulation infrastructure. *The European Physical Journal C*, 70(3):823–874, sep 2010.

[67] S. Chatrchyan et al. The CMS experiment at the CERN LHC. *JINST*, 3:S08004, 2008.

[68] Deep underground neutrino experiment. `http://www.dunescience.org/`.

[69] L. Bagby et al. Overhaul and Installation of the ICARUS-T600 Liquid Argon TPC Electronics for the FNAL Short Baseline Neutrino Program. *JINST*, 16(01):P01037, 2021.

[70] G. Abbiendi et al. Measuring the leading hadronic contribution to the muon g-2 via μe scattering. *Eur. Phys. J. C*, 77(3):139, 2017.

[71] Simone Scardapane, Indro Spinelli, and Paolo Di Lorenzo. Distributed training of graph convolutional networks. *IEEE Transactions on Signal and Information Processing over Networks*, 7:87–100, 2021.

[72] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, page 604–613, New York, NY, USA, 1998. Association for Computing Machinery.

[73] NVIDIA TensorRT. `https://docs.nvidia.com/deeplearning/tensorrt/index.html`, 2020. Accessed 2021-03-01.

[74] Shah Rukh Qasim, Jan Kieseler, Yutaro Iiyama, and Maurizio Pierini. Learning representations of irregular particle-detector geometry with distance-weighted graph networks. *Eur. Phys. J. C*, 79(7):608, 2019.

[75] Norman P. Jouppi et al. In-datacenter performance analysis of a tensor processing unit. *SIGARCH Comput. Archit. News*, 45(2):1–12, June 2017.

[76] Zhe Jia, Blake Tillman, Marco Maggioni, and Daniele Paolo Scarpazza. Dissecting the graphcore ipu architecture via microbenchmarking. 2019.

[77] Dan Guest, Kyle Cranmer, and Daniel Whiteson. Deep learning and its application to lhc physics. *Annual Review of Nuclear and Particle Science*, 68(1):161–181, 2018.

[78] Frederik G. Faye. Deepcalo, 2019.

[79] François Chollet et al. Keras. `https://keras.io`, 2015.

[80] ATLAS collaboration. Electron and photon reconstruction and performance in ATLAS using a dynamical, topological cell clustering-based approach. Technical report, CERN, Geneva, 2017. All figures including auxiliary figures are available at https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/PUBNOTES/ATL-PHYS-PUB-2017-022.

[81] ATLAS collaboration. Electron and photon performance measurements with the ATLAS detector using the 2015–2017 LHC proton-proton collision data. *Journal of Instrumentation*, 14(12):P12006–P12006, dec 2019.

[82] Darin Acosta, Andrew Brinkerhoff, Elena Busch, Andrew Carnes, Ivan Furic, Sergei Gleyzer, Khristian Kotov, Jia Fu Low, Alexander Madorsky, Jamal Rorie, Bobby Scurlock, Wei Shi, and on behalf of the CMS Collaboration. Boosted decision trees in the level-1 muon endcap trigger at cms. *Journal of Physics: Conference Series*, 1085(4):042042, sep 2018.

[83] Pallabi Das and on behalf of the CMS Collaboration. An overview of the trigger system at the cms experiment. *Physica Scripta*, 97(5):054008, apr 2022.

[84] ATLAS collaboration. Operation of the atlas trigger system in run 2. *Journal of Instrumentation*, 2020.

[85] ATLAS collaboration. Performance of the cms level-1 trigger in proton-proton collisions at $\sqrt{s} = 13tev$. *Journal of Instrumentation*, 15(10):P10017, oct 2020.

[86] Jeffrey Krupa, Kelvin Lin, Maria Acosta Flechas, Jack Dinsmore, Javier Duarte, Philip Harris, Scott Hauck, Burt Holzman, Shih-Chieh Hsu, Thomas Klijnsma, Mia Liu, Kevin Pedro, Dylan Rankin, Natchanon Suaysom, Matt Trahms, and Nhan Tran. GPU coprocessors as a service for deep learning inference in high energy physics. *Machine Learning: Science and Technology*, 2(3):035005, apr 2021.

[87] Burkhard Schmidt. The high-luminosity upgrade of the lhc: Physics and technology challenges for the accelerator and the experiments. *Journal of Physics: Conference Series*, 706(2):022002, apr 2016.

[88] ATLAS and CMS Collaborations. Report on the physics at the hl-lhc and perspectives for the he-lhc, 2019.

[89] Javier Duarte et al. Fast inference of deep neural networks in FPGAs for particle physics. *JINST*, 13(07):P07027, 2018.

[90] Jennifer Ngadiuba et al. Compressing deep neural networks on FPGAs to binary and ternary precision with HLS4ML. *Mach. Learn. Sci. Tech.*, 2:015001, 2021.

[91] Elham E Khoda, Dylan Rankin, Rafael Teixeira de Lima, Philip Harris, Scott Hauck, Shih-Chieh Hsu, Michael Kagan, Vladimir Loncar, Chaitanya Paikara, Richa Rao, Sioni Summers, Caterina Vernieri, and Aaron Wang. Ultra-low latency recurrent neural network inference on fpgas for physics applications with hls4ml, 2022.

[92] Thea Aarrestad et al. Fast convolutional neural networks on FPGAs with hls4ml. *Mach. Learn. Sci. Tech.*, 2(4):045015, 2021.

[93] Abdelrahman Elabd, Vesal Razavimaleki, Shi-Yu Huang, Javier Duarte, Markus Atkinson, Gage DeZoort, Peter Elmer, Scott Hauck, Jin-Xuan Hu, Shih-Chieh Hsu, Bo-Cheng Lai, Mark Neubauer, Isobel Ojalvo, Savannah Thais, and Matthew Trahms. Graph neural networks for charged particle tracking on FPGAs. *Frontiers in Big Data*, 5, mar 2022.

[94] S. Summers, G. Di Guglielmo, J. Duarte, P. Harris, D. Hoang, S. Jindariani, E. Kreinar, V. Loncar, J. Ngadiuba, M. Pierini, D. Rankin, N. Tran, and Z. Wu. Fast inference of boosted decision trees in FPGAs for particle physics. *Journal of Instrumentation*, 15(05):P05026–P05026, may 2020.

[95] FastML Team. fastmachinelearning/hls4ml, 2021.

[96] Claudionor N. Coelho, Aki Kuusela, Shan Li, Hao Zhuang, Jennifer Ngadiuba, Thea Klaeboe Aarrestad, Vladimir Loncar, Maurizio Pierini, Adrian Alan Pol, and Sioni Summers. Automatic heterogeneous quantization of deep neural networks for low-latency inference on the edge for particle detectors. *Nature Machine Intelligence*, 3(8):675–686, jun 2021.

[97] Claudionor N. Coelho Jr et al. Qkeras, 2019.

[98] Xilinx. Alveo u50 data center accelerator card, 2023.

[99] Nimmy M Philip and N M Sivamangai. Review of fpga-based accelerators of deep convolutional neural networks. In *2022 6th International Conference on Devices, Circuits and Systems (ICDCS)*, pages 183–189, 2022.

[100] Yuhao Wu. Review on fpga-based accelerators in deep learning. In *2023 IEEE 6th Information Technology,Networking,Electronic and Automation Control Conference (ITNEC)*, volume 6, pages 452–456, 2023.

[101] Yunxiang Hu, Yuhao Liu, and Zhuovuan Liu. A survey on convolutional neural network accelerators: Gpu, fpga and asic. In *2022 14th International Conference on Computer Research and Development (ICCRD)*, pages 100–107, 2022.

[102] Nicolò Ghielmetti et al. Real-time semantic segmentation on FPGAs for autonomous vehicles with hls4ml. *Mach. Learn. Sci. Tech.*, 2022.

[103] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[104] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C. Courville. Film: Visual reasoning with a general conditioning layer. *CoRR*, abs/1709.07871, 2017.

[105] Xilinx. Overview of arbitrary precision fixed-point data types, 2023.

[106] Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. *CoRR*, abs/1806.08342, 2018.

[107] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew G. Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. *CoRR*, abs/1712.05877, 2017.

[108] Xilinx. Xilinx runtime (xrt) library: query command, 2021.

[109] Xilinx. Alveo u50 data center accelerator card/buy online from amd, 2023.

[110] SHI. Nvidia tesla v100 - gpu computing processor overview, 2023.

[111] Alex Schuy. Low-latency calorimetry clustering at the lhc with spvcnn, October 2022. Talk presented at Fast Machine Learning for Science Workshop 2022, Southern Methodist University.

[112] The CMS Collaboration. Particle-flow reconstruction and global event description with the CMS detector. *Journal of Instrumentation*, 12(10):P10003–P10003, oct 2017.

[113] Haotian Tang, Zhijian Liu, Shengyu Zhao, Yujun Lin, Ji Lin, Hanrui Wang, and Song Han. Searching efficient 3d architectures with sparse point-voxel convolution, 2020.

[114] Jan Kieseler. Object condensation: one-stage grid-free multi-object reconstruction in physics detectors, graph, and image data. *The European Physical Journal C*, 80(9), sep 2020.

[115] The CMS Collaboration. The cms experiment at the cern lhc. *Journal of Instrumentation*, 3(08):S08004, aug 2008.

[116] The CMS Experiment. Electron and photon reconstruction and identification with the CMS experiment at the CERN LHC. *Journal of Instrumentation*, 16(05):P05014, may 2021.

[117] The CMS Experiment. Performance of the CMS muon detector and muon reconstruction with proton-proton collisions at $\sqrt{s}$ =13 TeV. *Journal of Instrumentation*, 13(06):P06015–P06015, jun 2018.

[118] The CMS Collaboration. Description and performance of track and primary-vertex reconstruction with the CMS tracker. *Journal of Instrumentation*, 9(10):P10009–P10009, oct 2014.

[119] *The CMS hadron calorimeter project: Technical Design Report*. Technical design report. CMS. CERN, Geneva, 1997.

[120] López, Silvia Goy. Cms detector performance. *EPJ Web Conf.*, 182:02076, 2018.

[121] CMS Collaboration. Technical proposal for the upgrade of the CMS detector through 2020. Technical report, 2011.

[122] Candan Isik. Phase 1 Upgrade of the CMS Hadron Calorimeter. Technical report, CERN, Geneva, 2022.

[123] The Phase-2 Upgrade of the CMS Endcap Calorimeter. Technical report, CERN, Geneva, 2017.

[124] The ATLAS Collaboration. The atlas experiment at the cern large hadron collider. *Journal of Instrumentation*, 3(08):S08003, aug 2008.

[125] Jet reconstruction and performance using particle flow with the ATLAS detector. *The European Physical Journal C*, 77(7), jul 2017.

[126] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. Point-voxel cnn for efficient 3d deep learning, 2019.

[127] Christian Bierlich, Smita Chakraborty, Nishita Desai, Leif Gellersen, Ilkka Helenius, Philip Ilten, Leif Lönnblad, Stephen Mrenna, Stefan Prestel, Christian T. Preuss, Torbjörn Sjöstrand, Peter Skands, Marius Utheim, and Rob Verheyen. A comprehensive guide to the physics and usage of pythia 8.3, 2022.

[128] Torbjörn Sjöstrand, Stephen Mrenna, and Peter Skands. PYTHIA 6.4 physics and manual. *Journal of High Energy Physics*, 2006(05):026–026, may 2006.

[129] S. Agostinelli, J. Allison, K. Amako, J. Apostolakis, H. Araujo, P. Arce, M. Asai, D. Axen, S. Banerjee, G. Barrand, F. Behner, L. Bellagamba, J. Boudreau, L. Broglia, A. Brunengo, H. Burkhardt, S. Chauvie, J. Chuma, R. Chytracek, G. Cooperman, G. Cosmo, P. Degtyarenko, A. Dell'Acqua, G. Depaola, D. Dietrich, R. Enami, A. Feliciello, C. Ferguson, H. Fesefeldt, G. Folger, F. Foppiano, A. Forti, S. Garelli, S. Giani, R. Giannitrapani, D. Gibin, J.J. Gómez Cadenas, I. González, G. Gracia Abril, G. Greeniaus, W. Greiner, V. Grichine, A. Grossheim, S. Guatelli, P. Gumplinger,

R. Hamatsu, K. Hashimoto, H. Hasui, A. Heikkinen, A. Howard, V. Ivanchenko, A. Johnson, F.W. Jones, J. Kallenbach, N. Kanaya, M. Kawabata, Y. Kawabata, M. Kawaguti, S. Kelner, P. Kent, A. Kimura, T. Kodama, R. Kokoulin, M. Kossov, H. Kurashige, E. Lamanna, T. Lampén, V. Lara, V. Lefebure, F. Lei, M. Liendl, W. Lockman, F. Longo, S. Magni, M. Maire, E. Medernach, K. Minamimoto, P. Mora de Freitas, Y. Morita, K. Murakami, M. Nagamatu, R. Nartallo, P. Nieminen, T. Nishimura, K. Ohtsubo, M. Okamura, S. O'Neale, Y. Oohata, K. Paech, J. Perl, A. Pfeiffer, M.G. Pia, F. Ranjard, A. Rybin, S. Sadilov, E. Di Salvo, G. Santin, T. Sasaki, N. Savvas, Y. Sawada, S. Scherer, S. Sei, V. Sirotenko, D. Smith, N. Starkov, H. Stoecker, J. Sulkimo, M. Takahata, S. Tanaka, E. Tcherniaev, E. Safai Tehrani, M. Tropeano, P. Truscott, H. Uno, L. Urban, P. Urban, M. Verderi, A. Walkden, W. Wander, H. Weber, J.P. Wellisch, T. Wenaus, D.C. Williams, D. Wright, T. Yamada, H. Yoshida, and D. Zschiesche. Geant4—a simulation toolkit. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 506(3):250–303, 2003.

[130] J. Allison et al. Recent developments in Geant4. *Nucl. Instrum. Meth. A*, 835:186–225, 2016.

[131] G. L. Bayatian et al. CMS Physics: Technical Design Report Volume 1: Detector Performance and Software. 2006.

[132] Javier Duarte et al. FPGA-accelerated machine learning inference as a service for particle physics computing. *Comput. Softw. Big Sci.*, 3:13, 2019.

[133] NVIDIA. Nvidia triton inference server. `https://docs.nvidia.com/deeplearning/triton-inference-server/user-guide/index.html`. Accessed: 2022-09-07.

[134] gRPC Authors. grpc. `https://grpc.io/`. Accessed: 2022-09-06.

[135] oneTBB. oneapi threading building blocks. `https://github.com/oneapi-src/oneTBB`. Accessed: 2022-09-06.

[136] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollár. Panoptic segmentation, 2019.

[137] Matteo Cacciari, Gavin P Salam, and Gregory Soyez. The anti-ktjet clustering algorithm. *Journal of High Energy Physics*, 2008(04):063–063, April 2008.

[138] NVIDIA. perf_client. `https://docs.nvidia.com/deeplearning/triton-inference-server/archives/triton_inference_server_1140/user-guide/docs/perf_client.html`. Accessed: 2023-07-04.

[139] Particle Data Group, C Amsler, M Doser, P Bloch, A Ceccucci, G F Giudice, A Höcker, M L Mangano, A Masoni, S Spanier, N A Törnqvist, B Krusche, and T DeGrand. Review of particle physics. *Physics Letters B*, 667(1-5):1–6, September 2008. A booklet is available containing the Summary Tables and abbreviated versions of some of the other sections of this full Review. All tables, listings, and reviews (and errata) are also available on the Particle Data Group website: http://pdg.lbl.gov.

[140] Lyndon Evans and Philip Bryant. Lhc machine. *Journal of Instrumentation*, 3(08):S08001, aug 2008.

[141] Cern accelerating science. (n.d.), 2016.

[142] ATLAS Collaboration. Observation of a new particle in the search for the standard model higgs boson with the atlas detector at the lhc. *Physics Letters B*, 716(1):1–29, 2012.

[143] K Binder, D Heermann, L Roelofs, and ... Monte carlo simulation in statistical physics. *Computers in ...*, 1993.

[144] Xilinx. Amd ryzen 5 5600h, 2023.

[145] NVIDIA. Nvidia v100 tensor core, 2023.

[146] K. Pedro. Soniccms, 2020.

[147] Yu-Hsin Chen, Tushar Krishna, Joel Emer, and Vivienne Sze. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. In *IEEE Journal of Solid-State Circuits*, volume 52, pages 127–138. IEEE, 2017.

[148] Zhu Qiu, Jason Cong, and Youxiang Li. Flexcnn: A flexible systolic array-based fpga accelerator for convolutional neural networks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2021.

[149] Xilinx vitis ai, 2023.

[150] Michaela Blott, Thomas B Preußer, Nicholas J Fraser, Giulio Gambardella, Kenneth O'brien, Yaman Umuroglu, Miriam Leeser, and Kees Vissers. Finn-r: An end-to-end deep-learning framework for fast exploration of quantized neural networks. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 11(3):1–23, 2018.

[151] Yaman Umuroglu, Nicholas J. Fraser, Giulio Gambardella, Michaela Blott, Philip Leong, Magnus Jahre, and Kees Vissers. Finn: A framework for fast, scalable binarized neural network inference. pages 65–74, 2017.

[152] Timothy Dozat. Incorporating nesterov momentum into adam. In *International Conference on Learning Representations*, 2016.

[153] Claudionor N. Coelho, Aki Kuusela, Shan Li, Hao Zhuang, Jennifer Ngadiuba, Thea Klaeboe Aarrestad, Vladimir Loncar, Maurizio Pierini, Adrian Alan Pol, and Sioni Summers. Automatic heterogeneous quantization of deep neural networks for low-latency inference on the edge for particle detectors. *Nature Machine Intelligence*, 3(8):675–686, jun 2021.

[154] ATLAS collaboration. Measurements of higgs boson properties in the diphoton decay channel with 36 fb^{-1} of pp collision data at $\sqrt{s} = 13$ TeV with the atlas detector. *Phys. Rev. D*, 98:052005, Sep 2018.

[155] Xilinx. Vivado hls 2019.1: pragma hls dataflow. `https://www.xilinx.com/htmldocs/xilinx2019_1/sdaccel_doc/hls-pragmas-okr1504034364623.html#sxx1504034358866`, 2019.

[156] Xilinx. Vivado hls 2019.1: pragma hls data_pack, 2019.

[157] Xilinx. Vivado hls 2019.1: pragma hls unroll, 2019.

[158] Xilinx. Vitis high-level synthesis user guide (ug1399): Quantization modes, 2023.

[159] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[160] Xilinx. Xilinx dsp48e2 block, 2023.

[161] Xilinx. Alveo u250 data center accelerator card, 2023.

[162] Leslie N. Smith. Cyclical learning rates for training neural networks, 2017.

[163] ATLAS Collaboration. Computing and Software Public Results, 2017.

[164] Tal Ben-Nun and Torsten Hoefler. Demystifying Parallel and Distributed Deep Learning: An In-Depth Concurrency Analysis. 2018.

[165] Alexander Sergeev and Mike Del Balso. Horovod: fast and easy distributed deep learning in TensorFlow. 2018.

[166] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning. *CoRR*, abs/1712.04621, 2017.

[167] Martín Abadi et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. 2016.

[168] Ville Bergholm, Josh Izaac, Maria Schuld, Christian Gogolin, M. Sohaib Alam, Shahnawaz Ahmed, Juan Miguel Arrazola, Carsten Blank, Alain Delgado, Soran Jahangiri, Keri McKiernan, Johannes Jakob Meyer, Zeyue Niu, Antal Száva, and Nathan Killoran. Pennylane: Automatic differentiation of hybrid quantum-classical computations. 2020.