U N I V E R S I T Y O F S P L I T
FACULTY OF ELECTRICAL ENGINEERING, MECHANICAL ENGINEERING
AND NAVAL ARCHITECTURE

Arijana Burazin Mišura

ARTIFICIAL INTELLIGENCE ALGORITHMS FOR THE REAL-TIME SELECTION OF PHYSICALLY IMPORTANT EVENTS WITHIN THE CMS L1 TRIGGER FPGA DEVICES

Split, 2024.

U N I V E R S I T Y O F S P L I T
FACULTY OF ELECTRICAL ENGINEERING, MECHANICAL ENGINEERING
AND NAVAL ARCHITECTURE

Arijana Burazin Mišura

ARTIFICIAL INTELLIGENCE ALGORITHMS FOR THE REAL-TIME SELECTION OF PHYSICALLY IMPORTANT EVENTS WITHIN THE CMS L1 TRIGGER FPGA DEVICES

DOCTORAL THESIS

Split, 2024.

The research reported in this thesis was carried out at (Department name), University of Split, Faculty of Electrical Engineering, Mechanical Engineering and Naval Architecture.

Supervisor: Assoc. Prof. Josip Musić, Ph. D., FESB, University of Split, Croatia
Co-Supervisor: Prof. Linda Vicković, Ph. D., FESB, University of Split, Croatia
Disertation number: xxx-xxx

BIBLIOGRAPHIC INFORMATION

Keywords: xxxxx, yyyyy
Scientific area: xxxxx
Scientific field: xxxxx
Scientific branch: xxxxx
Institution of PhD completion: University of Split, Faculty of Electrical Engineering, Mechanical Engineering and Naval Architecture
Supervisor of the thesis: Xxxx dr.sc. Xxxx Yyyyy
Number of pages: yyy
Number of figures: yyy
Number of tables: yyy
Number of references: yyy

Committee for assessment of doctoral dissertation:

1. Xxxx. dr. sc. Xxxx Yyyyy, Institution name and City/Town
2. Xxxx. dr. sc. Xxxx Yyyyy, Institution name and City/Town
3. Xxxx. dr. sc. Xxxx Yyyyy, Institution name and City/Town
4. Xxxx. dr. sc. Xxxx Yyyyy, Institution name and City/Town
5. Xxxx. dr. sc. Xxxx Yyyyy, Institution name and City/Town

Committee for the defence of doctoral dissertation:

1. Xxxx. dr. sc. Xxxx Yyyyy, Institution name and City/Town
2. Xxxx. dr. sc. Xxxx Yyyyy, Institution name and City/Town
3. Xxxx. dr. sc. Xxxx Yyyyy, Institution name and City/Town
4. Xxxx. dr. sc. Xxxx Yyyyy, Institution name and City/Town
5. Xxxx. dr. sc. Xxxx Yyyyy, Institution name and City/Town

Dissertation defended on: xx. xx. 2024.

Artificial Intelligence Algorithms for the Real-Time Selection of Physically Important Events Within the CMS L1 Trigger FPGA Devices

Abstract

The Large Hadron Collider is the largest particle accelerator in the world with the purpose of colliding protons accelerated to nearly the speed of light, resulting in the creation of new and exotic particles. Four detectors surround the collision points and record the data resulting from the proton-proton collisions. This data is then used for a wide range of physical analyses that intend to improve our understanding of particle physics and predict new theories. To increase its discovery potential, the LHC project aims to significantly boost the luminosity in the High Luminosity-Large Hadron Collider phase, which will, in turn, increase the number of collision events. The detector technologies are improved to address the challenges posed by the increased luminosity. However, it has become imperative to optimize the algorithms in the data selection system to effectively manage the presence of multiple interactions in the same bunch crossing (pile-up). This research focuses on using neural networks to select physically important events in real-time in the Compact Muon Solenoid, one of the LHC detectors. Simple multilayer perceptron and convolutional neural networks are developed to successfully distinguish electromagnetic showers from different types of backgrounds using raw detector images. Data selection and quantization methods are proposed to reduce the size of the presented models. Finally, profiling tools enable the determination of the precision settings needed to keep model inference accuracy, which further reduces the memory and computational requirements. The suggested approach was assessed using simulated data since the detector is still in the production phase. The inference time was less than 1 microsecond and signal-background classification achieved a classification accuracy of 97.01% for 2-bit-only quantization. This accuracy is the same as for the full-precision standard network, with a slightly decreased false negative rate. Overall, the research output presented within the thesis confirms the successful implementation of developed models in the targeted device while meeting the latency requirement, which makes them candidates for one of the level 1 trigger algorithms.

Keywords:
HL-LHC, CMS, Level 1 Trigger, Machine Learning, Neural Network, Classification, Quantized Neural Network

Algoritmi umjetne inteligencije za selekciju fizikalno bitnih događaja u realnom vremenu na FPGA uređajima unutar CMS L1 okidača

Sažetak

Veliki Hadronski Sudarač (engl. *Large Hadron Collider*, LHC) je najveći akcelerator čestica na svijetu. U njemu se sudaraju protoni, ubrzani skoro do brzine svjetlosti, što rezultira stvaranjem novih i egzotičnih čestica. Četiri detektora okružuju točke sudara i bilježe dobivene podatke iz proton-proton sudara. Ti se podaci zatim koriste za širok raspon fizikalnih analiza s ciljem poboljšanja našeg razumijevanja i predviđanja različitih teorija fizike čestica. Kako bi se povećala mogućnost otkrića, LHC projekt planira značajno povećati luminozitet u HL-LHC (engl. *High Level* LHC), što će rezultirati povećanjem broja sudara. Kako bi bilo moguće pratiti izazove koje predstavlja povećanje luminoziteta trebalo je poboljšati tehnologije koje se koriste unutar detektora. Međutim, to ipak nije dovoljno, pa je imperativ postala optimizacija algoritama u sustavu odabira podataka kako bi se učinkovito upravljalo prisutnošću višestrukih interakcija u istom skupu sudara (pile-up). Ovo istraživanje usmjereno je na korištenje neuronskih mreža u odabiru fizički važnih događaja u stvarnom vremenu u Kompaktnom Muonskom Solenoidu, jednom od LHC detektora. Razvijene su jednostavne višeslojne perceptronske i konvolucijske neuronske mreže koje uspješno razlikuju elektromagnetske pljuskove od različitih vrsta pozadine koristeći neobrađene slike detektora. Predložene su metode odabira podataka i kvantizacije koje smanjuju veličinu predstavljenih modela. Konačno, alati za profiliranje omogućuju utvrđivanje postavki preciznosti potrebnih za održavanje točnosti zaključivanja modela, što dodatno smanjuje memorijske i računalne zahtjeve. Predloženi pristupi ispitani su korištenjem simuliranih podataka budući da je detektor još uvijek u fazi izrade. Vrijeme zaključivanja bilo je manje od 1 mikrosekunde te je kod klasifikacije signala i pozadine postignuta točnost od 97,01% pri kvantizaciji modela i podataka na samo 2 bita. Ova točnost je ista kao i za standardnu mrežu pune preciznosti, međutim s blagim padom stope lažno negativnih rezultata. Sveukupno, rezultati istraživanja predstavljeni u okviru teze potvrđuju uspješnu implementaciju razvijenih modela u ciljani uređaj uz ispunjavanje zahtjeva latencije, što ih čini kandidatima za jedan od algoritama okidanja razine 1.

Ključne riječi:
HL-LHC, CMS, okidač razine 1, strojno učenje, neuronska mreža, klasifikacija, kvantizirana neuronska mreža

Acknowledgments

This PhD thesis ....

Contents

List of Tables

List of Figures

List of Acronyms

2D	Two-Dimensional
3D	Three-Dimensional
ACC	Accuracy
AI	Artificial intelligence
ALICE	A Large Ion Collider Experiment
ANN	Artificial Neural Network
ASIC	Application-Specific Integrated Circuits
ATLAS	A Toroidal LHC Apparatus
AUC	Area Under the Curve
BDT	Boosted Decision Tree
BRAM	Block Random Access Memory
CDF	Collider Detector at Fermilab
CE-E	Electromagnetic Section
CE-H	Hadronic Section
CERN	European Organization for Nuclear Research
CMS	Compact Muon Solenoid
CMSSW	CMS Software Components
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CSC	Cathode Strip Chamber
DSP	Digital Signal Processor
DT	Drift Tube
DUNE	Deep Underground Neutrino Experiment
ECAL	Electromagnetic calorimeter
EM	Electromagnetic
FE	Feature Extraction
FF	Flip-Flop
FN	False Negative
FNR	False Negative Rate
FP	False Positive
FPGA	Field Programmable Gate Array
FPR	False Positive Rate
FS	Feature Selection
FZ	Float Zone

GEM	Gas Electron Multiplier
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
HCAL	Hadronic calorimeter
HDL	Hardware Description Language
HEP	High Energy Physics
HGCAL	High Granularity Calorimeter
HL-LHC	High Luminosity LHC
HLS	High-Level Synthesis
HLT	High-Level Trigger
L1T	Level 1 Trigger
LHC	Large Hadron Collider
LHCb	Large Hadron Collider beauty
LS	Long Shutdown
LSTM	Long Short-Term Memory
LUT	LookUp Table
PU	Pile Up
MC	Monte Carlo
ML	Machine Learning
MLP	Multilayer Perceptron
NN	Neural Network
PF	Particle Flow
QCD	Quantum Chromodynamics
RF	Random Forrest
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristic
ROI	Region of Interest
RPC	Resistive Plate Chambers
RTL	Register Transfer Level
SHAP	Shapley Additive Explanations
SOTA	State-Of-The-Art
SVM	Support Vector Machine
TC	Trigger Cell
TLU	Threshold Logic Unit
TN	True Negative
TNR	True Negative Rate
TP	True Positive

TPG Trigger Primitive Generators

TPR True Positive Rate

1 INTRODUCTION

The 21st century has witnessed a technological revolution, the impact of which is visible in many aspects of life. One area of study that plays a crucial role in this transformation is machine learning (ML). The digital footprints left by sensors, connected devices, and the Internet have generated a huge amount of information. ML's data-centric approach is essential for uncovering the valuable insights hidden within enormous amounts of data. This field of study, which often serves as the foundation for artificial intelligence (AI) applications, has infiltrated all aspects of our daily lives, from virtual personal assistants on our phones to self-driving cars. Although the original development was purely for scientific purposes, today, ML profoundly impacts industries, healthcare, finance, and more. The field of ML is currently being revolutionized by neural networks (NN), which are ML models inspired by the structure and function of the human brain. Their ability to learn from data, recognize patterns, and make predictions has made them useful in various fields, particularly in high-energy physics (HEP).

1.1 Motivation

The Large Hadron Collider (LHC) [1] experiment, organized by the European Organization for Nuclear Research (CERN), represents the peak of scientific research in particle physics. Its revolutionary discoveries deepened the understanding of the universe and raised technology and international cooperation to a new level. LHC is the world's largest particle accelerator, employing thousands of superconducting magnets to accelerate and steer particle beams in opposite directions inside a 27-kilometer ring. Inside the accelerator tube, protons collide, creating unstable particles that rapidly decay into a cascade of more stable particles. The collisions occur at four locations - detectors around the accelerator ring: A Toroidal LHC Apparatus (ATLAS), a Compact Muon Solenoid (CMS) [2], A Large Ion Collider Experiment (ALICE), and a Large Hadron Collider beauty (LHCb), as shown on Figure 1.1. ATLAS is a general-purpose detector that shares the scientific goals of CMS experiments but uses different technical solutions and a different magnet-system design. As the topic of this thesis concerns the processing of data obtained by collisions in CMS, this detector will be described in more detail in the Chapter 2.1. The ALICE detector aims to study the physics of strongly interacting matter at the highest energy densities. At the same time, LHCb is

specialized in investigating the slight differences between matter and antimatter by studying a type of particle called the "beauty quark", or "b quark".

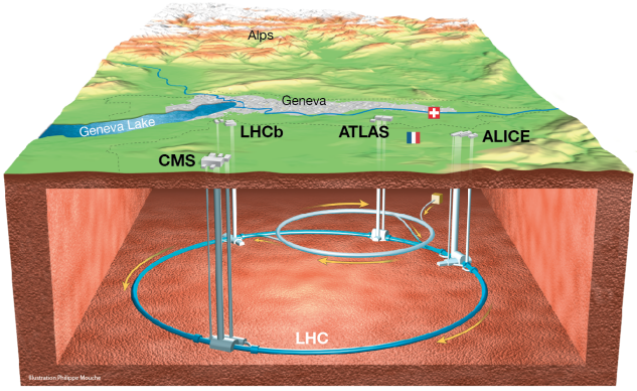


Figure 1.1. Overall view of the Large Hadron Collider, showing the ATLAS, CMS, ALICE, and LHCb detectors [3].

It is important to note that proton-proton collisions are not limited to the primary hard-scattering event. They also involve many concurrent soft collisions, named pile-up (PU), which deposit additional energy in the detector parts. These energy deposits complicate identifying the resulting particles, which can significantly impact the understanding and studying of the primary results of collisions.

The LHC operates in several phases, as shown in Figure 1.2, each with specific goals and energy levels. Periods of physics production, where protons or heavy ions collide, alternate with shutdown periods. Those are crucial periods when essential repairs and upgrades occur, ensuring optimal performance and functionality. The initial system worked effectively during LHC Run 1, and Francois Englert and Peter W. Higgs were awarded the Nobel Prize in Physics in 2013. "for the theoretical discovery of a mechanism that contributes to our understanding of the origin of mass of subatomic particles, and which recently was confirmed through the discovery of the predicted fundamental particle, by the ATLAS and CMS experiments at CERN's Large Hadron Collider" [4].

To provide a more extensive data set that would increase the potential for discoveries, the integrated luminosity (total number of collisions) steadily increased throughout LHC Run 2. Currently, the LHC is in the Run 3 data collection phase since 2022. The LHC is undergoing a critical upgrade to greatly expand its physics reach by rigorously increasing peak luminosity (rate of collisions - the number of potential collisions per unit area per second) by a factor of five beyond the original design value (to $5x10^{34}cm^{-2}s^{-1}$) [5]. Although the construction work for the HL-LHC is currently in progress, the installation and upgrade will occur during the LS3 period. The concept of luminosity is the proportionality factor between the number

of events per second dR/dt and the cross-section σ and can be calculated using

$$L = \frac{1}{\sigma}\frac{dR}{dt}. \tag{1.1}$$

Another evaluation parameter is the integrated luminosity, proportional to a number of events of interest [6], calculated as

$$L_{int} = \int_0^T L(t)dt. \tag{1.2}$$

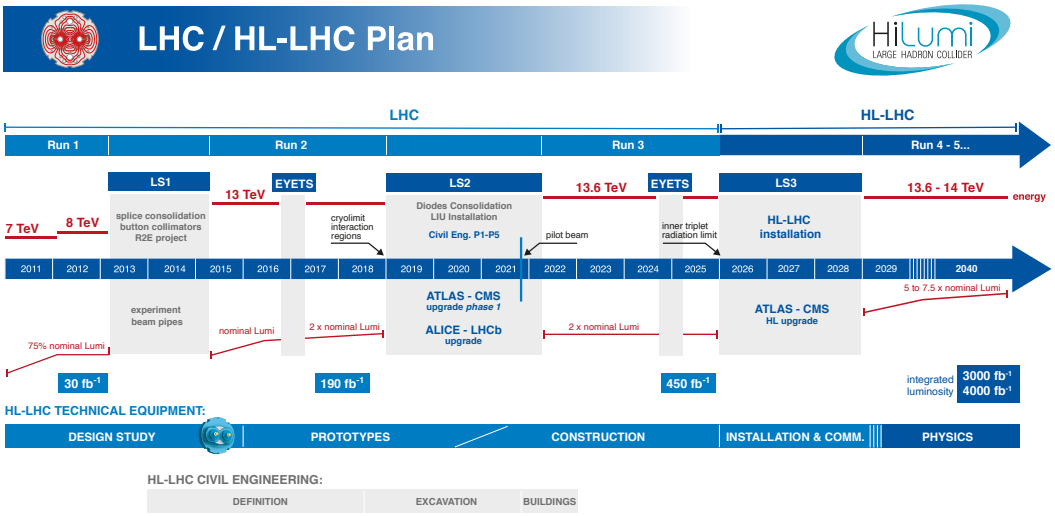


Figure 1.2. The collision energy (upper line) and the luminosity (lower line) are displayed in the LHC's baseline plan for the next decade. LS refers to "Long Shutdown", scheduled periods when the LHC is not operational. During this time, significant maintenance, upgrades, and enhancements are undertaken to improve its performance and prepare for future runs. The project equipment will be in the high-luminosity configuration after LS3 [7].

The integrated luminosity will be increased by a factor of ten (to 4000 fb^{-1}), as shown in Figure 1.3, leading to an increase in the average number of proton-proton collisions per bunch crossing (pile-up, PU) to around 200. Such an enormous number of particles produced in collisions and their induced radioactivity can significantly damage detectors and electronics.

Therefore, parts of LHC equipment, mostly detectors ATLAS and CMS, must be modified to deal with the upgraded performance. The updated system must be radiation-tolerant and capable of handling high PU, particularly at the trigger level. The trigger plays a crucial role in the data acquisition system by selecting a subset of collision events and filtering out only the most scientifically significant ones for further analysis. To address these challenges, CMS proposes an innovative solution: the replacement of the crucial part of trigger architectures, endcap calorimeters, with the High-Granularity Endcap Calorimeter (HGCAL). The

hardware improvement enables the development and use of advanced low-latency trigger algorithms that effectively and accurately select crucial events from the vast amount of generated data. With its state-of-the-art technological advancements, the new configuration, named High Luminosity LHC (HL-LHC), surpasses existing accelerator limits.

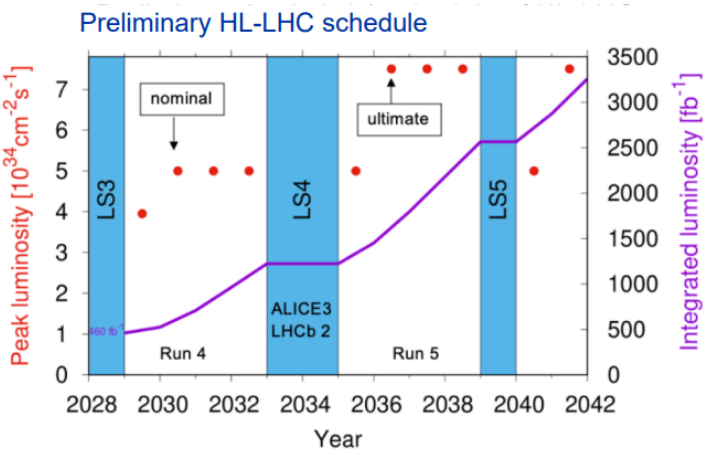


Figure 1.3. The forecast for luminosity value during the HL-LHC era, the peak luminosity is indicated by red dots, while a violet line shows the integrated luminosity [8].

1.2 Literature Review

In the field of HEP, innovative approaches to data analysis, pattern recognition, and interpretation are constantly needed to drive revolutionary discoveries. Traditional analytical approaches, which are reliant on manual data exploration and interpretation, are encountering significant challenges in managing the scale and complexity of the data. Consequently, for several years, researchers have turned to ML techniques as an efficient means to process these large volumes of data in HEP experiments.

ML was first applied to HEP analysis in the 1990s, and since the 2010s, it has been widely used in particle and event identification and reconstruction. In their review article [9], the authors discuss the implementation details, resource requirements, and development areas of ML in particle physics. In the beginning, Boosted Decision Trees (BDT) were the most extensively utilized ML algorithm in HEP [10, 11], mostly used to classify particles and events. The usage of NNs in HEP started in 1987 when Denby in [12] suggested using NN for track reconstruction and cluster finding. The initial use of NNs and other ML methods relied heavily on manual feature engineering, which demanded a more in-depth understanding of particle decay phenomenology. In 2014, Baldi [13] showed that in some classification problems in HEP, shallow NNs using low-level data achieve almost the same performance as those using high-level features, which opened up new possibilities for using NNs in HEP problems. Manual feature engineering, used for traditional approaches, can be a

time-consuming and incomplete process, so relying on NNs to extract relevant features from raw data automatically is the more efficient and effective solution. The advancements in computing technology have facilitated the greater application of NNs in many domains, such as image and speech recognition, natural language processing, and predictive analytics. As a result, NNs have become a popular choice among scientists, as they offer superior performance in many complex and challenging tasks, including those that involve high-dimensional data, complex patterns, and non-linear relationships. Therefore, in recent years, there has been a notable trend in the employment of NN.

An innovative approach to using NNs to process detector data was proposed by Cogan et al. [14], who recognized that the projection of the calorimeter structure, which is present in almost all detectors used in HEP, is similar to image pixels. This way of data representation allowed physicists to use new tools in image processing, Convolutional Neural Networks (CNNs). Authors in [15–17] employ CNN for the classification of particle collisions, in [18] CNN is applied in the filtering of track pixel seeds while authors in [19] use CNN for anomaly detection. Other advanced NN architectures are also used in HEP: in [20] distance-weighted graph NN is used for particle reconstruction, generative adversarial networks are often used for simulation of physical events [21, 22], and in [23] authors use automated tool based on autoencoder for CMS Resistive Plate Chambers (RPC) current monitoring to ensure stable operation.

HEP experiments typically employ complex trigger systems to select physically interesting events from the data stream. The first who suggested the use of NNs for triggering was Denby, 1990. In [24], he presented the idea of using NN to increase the trigger efficiency in the Collider Detector at Fermilab (CDF) experiment. The available dataset was small, with 450 samples, and the region of interest (ROI) was a cell in the calorimeter with the highest energy deposit, covering an 8x8 area. There was a massive gap between training and testing accuracy for suggested feedforward NN, and the plan was to implement NN in hardware. Although NNs have been widely used in physics analysis due to constrained conditions and low latency requirements, they are not frequently applied in low-level triggering. In high-level CMS triggers, where there are no such resource constraints, NNs are used to label jets, sprays of particles produced by the hadronization of quarks and gluons. However, due to the limited size, required latency, and radiation exposure, the traditional approach to NNs is not acceptable in CMS Level 1 Trigger (L1T). For this reason, work has begun on new approaches that enable NN usage in limited conditions.

The reduction of memory and processing requirements while preserving the accuracy of the NN was attempted by quantization of the network. The first efficient approach to aggressive NN quantization was presented in 2015 by Courbariaux et al. [25] with BinaryConnect, an NN that uses binary weight values while maintaining accuracy. In their study, Rastegari et al. [26] introduced XNOR-Net, a binary convolutional neural network that was tested on the ImageNet dataset. The researchers experimented with two different approaches. The

first approach involved binarizing only the weights of the network, which the researchers called Binary-Weight-Networks. The second approach, which they called XNOR-Networks, involved binarizing both the weights and the input to the network. Both approaches were able to save approximately 32 times (more) memory. This was made possible by using binary weights, which allowed convolutions to be performed without using the multiplication operation. As a result, the first approach achieved an inference speedup of approximately 2 times, while the second approach achieved a speedup of approximately 58 times, compared to the model that used complete precision for the network and data.

The reduction in model size and the development of machine learning-compatible Field-Programmable Gate Array (FPGA) devices have created new opportunities for implementing NNs on L1T. The use of ML on FPGAs is an area of active research, with limited resources impacting the size and complexity of potential models.

An early attempt to deploy a CNN for particle track reconstruction on an FPGA was presented at NIPS 2017 [27]. Today, the implementation of ML models on edge devices is possible using a dedicated tool such as *hls4ml* [28, 29], a library for interpreting and translating ML algorithms into hardware-readable code. The created library contains numerous techniques that enable the implementation of low energy consumption and increased usability and is fully compatible with QKeras, a quantization extension to Keras. The first who employed *hls4ml* are authors in [30], where they present a fully connected classifier (based on high-level features) for jet substructure implemented in FPGAs. The quantized model achieved an Area Under the Curve (AUC) value of 99.55% with a latency of 75 ns. Additionally, a detailed description of the translation process of an NN into an FPGA implementation is provided in the paper. The same dataset was used in the [31], where authors also introduced QKeras. After implementing quantized models with *hls4ml* in FPGA, it is demonstrated that resource utilization can be reduced by up to 98% without significant loss in model accuracy while performing inference within $O(10)$ ns. Two multi-class classification tasks are considered in [32]: handwritten digit recognition with the MNIST dataset and jet identification in simulated proton-proton collisions at LHC. The developed models undergo aggressive quantization into binary and ternary networks, which are then implemented in FPGA. Upon comparison with binary models, ternary models demonstrate significantly improved accuracy values that closely align with the original baseline models, with a smaller resource cost, and with comparable latency. In the publication [33], a continuation of the QKeras/*hls4ml* approach is presented. This work introduces an extension to the *hls4ml* library, which facilitates the implementation of convolutional architectures. The created CNN model is used as a digit classifier on the Street View House Numbers Dataset. After conversion to FPGA firmware, it can be executed with 5 µs latency while consuming less than 10% of the FPGA resources. Due to its proven efficiency, implementing NN models in FPGA using *hls4ml* has become an active area of research [34–36].

The research results show that aggressive quantization techniques can help an NN model

meet the challenging L1T requirements. The proposed input data selection and quantization methods can further reduce model size. By utilizing appropriate *hls4ml* configuration settings, additional resources can be saved, leading to reduced latency.

1.3 Hypothesis

The primary objective of this doctoral research is to develop a novel principle for the real-time selection of physically significant events, such as electromagnetic (EM) showers, utilizing the power of artificial intelligence algorithms, particularly NNs. This selection is a crucial part of the CMS real-time trigger system implemented on the FPGA devices. Although used FPGA devices represent the state of the art, they still have limited capabilities compared to the Central Processing Unit (CPU) and Graphics Processing Unit (GPU) based devices on which NNs are usually running. However, FPGAs are preferred in L1T systems due to their low latency, parallel processing capabilities, flexibility, and efficiency. These characteristics make FPGAs suitable for the real-time data processing demands of HEP experiments. This research aims to enable efficient real-time selection by extracting hidden insights from the raw detector data using quantized NNs combined with data selection and quantization techniques.

As a result of this research, the following hypotheses have been proposed:

• **H1:** *Standard neural network architectures can be adapted to classify physically important events in real-time and under conditions of limited computing resources while maintaining the classification accuracy of CMS level 1 triggers.*

• **H2:** *It is possible to implement developed architectures of neural networks and other machine learning algorithms in a form suitable for operation on FPGA devices. Additional improvements in FPGA performance can be achieved by appropriately selecting and aggressively quantizing the input data of the neural network.*

• **H3:** *It is possible to implement a new model of generating input data (images) obtained based on the CMS simulator for application in neural networks and other machine learning algorithms.*

The first hypothesis (H1) is based on the model's quantization principle, which would significantly reduce the model's size and, therefore, its need for memory and computing resources. By developing and testing neural network models, quantized Multilayer Perception (MLP), and Convolutional Neural Network (CNN), it is possible to classify EM signals from different backgrounds (other signals or noise that can be detected along with the EM showers) with high accuracy. The performance and efficiency of proposed NN models are examined using standard metrics, such as accuracy, AUC, true positive rate (TPR), and true negative rate (TNR). Based on the obtained results, it is possible to impact the TPR and TNR values using different threshold values, depending on preferences.

The second hypothesis (H2) assumes that it is possible to implement developed models

on FPGA devices used for level 1 triggering. Analyzing and selecting the optimal precision needed for individual model layers is possible using appropriate profiling tools. It leads to additional resource savings, looking specifically at the number of used Block Random Access Memories (BRAMs) and LookUp Tables (LUTs) while simultaneously lowering the latency time. As FPGA devices are simultaneously divided between a large number of applications and triggers, each saving significantly contributes to the functionality of the entire system. Furthermore, the empirical evidence suggests that the implemented models not only reduce resource consumption but also meet the required latency, as described in Chapter 3.4. This highlights the efficiency and effectiveness of the models, making them a valuable asset to any project seeking to optimize its resources.

The third hypothesis (H3), described in Chapter 2.5 is based on the assumption that HGCAL images based on energy deposits on calorimeter layers could be improved if, instead of the cluster barycenter, the emphasis would be on observing the longest sequence of deposited energies. This assumption particularly applies to PU images, where the decay may occur in a direction other than the beamline. As the quality of images describing the decay of particles is directly responsible for the correct selection of physically interesting events, this new approach could improve the accuracy of the trigger algorithm i.e. proposed NN models.

1.4 Dissertation Outline

The research is divided into theoretical and empirical parts in accordance with the objectives stated in the previous section. The initial part of the theoretical research given in Chapter Materials and Methods 2 describes the CMS detector, including its subdetectors and their functions. Additionally, the research clarifies the challenges associated with detector upgrades required for efficient operation in the project's HL-LHC era. Subsequently, the section provides a detailed analysis of the CMS Trigger functionality. One of the main challenges with the CMS upgrade is to develop an algorithm that can efficiently select events of interest to the CMS upgrade.

The following section 2.2 provides a comprehensive summary of the ML algorithms used in the thesis. The proposed algorithms include fully connected NN and convolutional NN. The performance of these algorithms will be compared with other well-known ML methods, such as support vector machines and random forests. In addition to this, the thesis explores existing solutions for using ML to solve various HEP problems in the literature. It will provide a complete analysis of the current state of described ML algorithms in HEP, with a focus on NN. It will highlight the most promising approaches and outline areas that require further development and research.

The hypotheses that have been presented will be put to the test in the empirical section of the thesis 3. The development of an application that generates HGCAL images based on data collected from the CMSSW simulator is a crucial aspect of empirical research. The

hypotheses presented earlier will be evaluated using generated images based on the findings from the theoretical section. The developed NN models are put to the test by evaluating their performance using standard metrics such as accuracy, precision, and recall. The final step is to implement the most successful models in *hls4ml*. The results obtained from the empirical analysis are examined to determine whether the proposed approach is suitable and can be implemented in practice in the incoming LHC phase.

For this research study, the computations were carried out in different environments. Data simulation was done using CMSSW 12.1. Image generation and ML model development were done on a laptop computer, MacBook Pro, with a 2.6 GHz Dual-Core Intel Core i5 processor and 8 GB 1600 MHz DDR3 RAM. The operating system was IOS macOS Catalina, version 10.15.4, which provided a stable and reliable environment. The programming language used was Python 3.8.5, with Keras 2.15.0, TensorFlow 2.4.1, and QKeras 0.8.0. Finally, FPGA implementation was done on Xilinx Virtex UltraScale, part number xcvu13p-fhgb2104-2-e, by using Vivado HLS 2020.1 and *hls4ml* 0.6.0.

2 MATERIALS AND METHODS

The chapter begins with a description of the CMS detector, focusing on the parts relevant to this thesis. After that, an overview of the ML techniques used in HEP is provided. The subsequent sections cover the creation of the dataset, the development of the NN models, and proposed reduction methods.

2.1 The CMS Detector

CMS is one of four LHC detectors located on the accelerator tube at one of the points where protons collide in the LHC's ring. As a result of the collisions, new unstable particles are generated, which immediately break apart. Decay products disperse in all directions and interact with detector layers made of different materials, as shown in Figure 2.1. The results of the interactions provide a 3D image of the observed collisions, whose analysis can lead to discoveries of new physical phenomena.

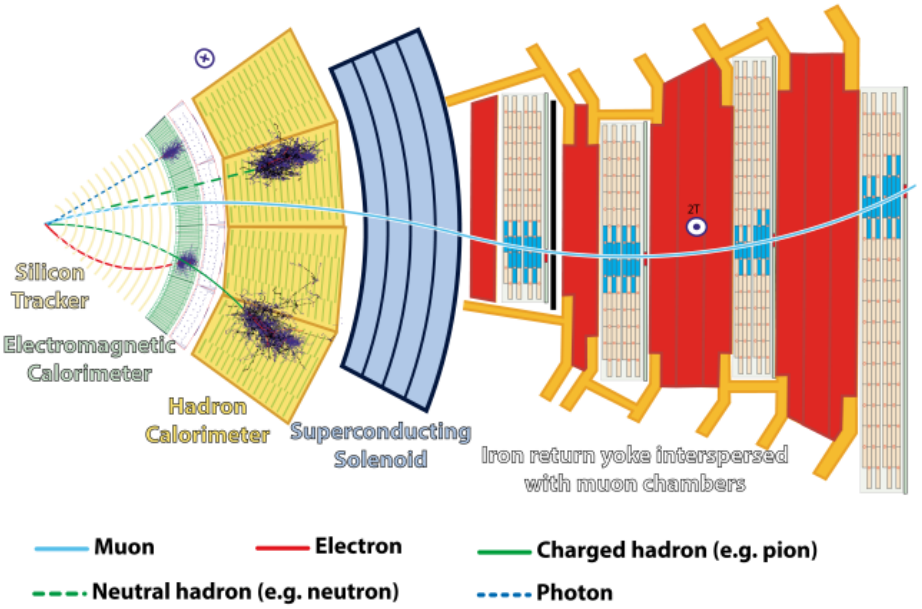


Figure 2.1. The illustration of a CMS transverse slice shows the different sub-detectors and how they interact with individual particles. The solenoid produces a 4 Tesla axial field and requires a return yoke to control the field outside of the solenoid, with a field strength of around 2 Tesla [37].

One of the most significant scientific breakthroughs in particle physics, the discovery of the Higgs boson that confirmed the existence of the Brout–Englert–Higgs mechanism, was made possible with the help of the CMS detector [38]. The CMS, one of two general-purpose detectors at the LHC, has made crucial discoveries in particle physics in tandem with ATLAS. Although they share scientific goals, each detector uses different technical solutions and event reconstruction software.

The Compact Muon Solenoid (Figure 2.2) is named after its three main characteristics. First, its compactness: even though ATLAS is almost twice as large as CMS, CMS is double-weighted due to the detector material it accommodates, making it more massive at 14 000 tons. Secondly, detecting muons is at the forefront of CMS' design, making it a powerful tool for uncovering thrilling discoveries in physics. Unlike electrons and mesons, muons can only come from the decay of heavy particles, which makes them easily identifiable and possibly interesting. Lastly, at the center of CMS, is a 13 m long superconducting solenoid with a 6 m inner diameter, providing a magnetic field of 3.8 T, which was the most powerful one ever built at the time of its creation. To achieve accurate measurement of the momentum of charged particles, a magnetic field of significant strength is an absolute necessity.

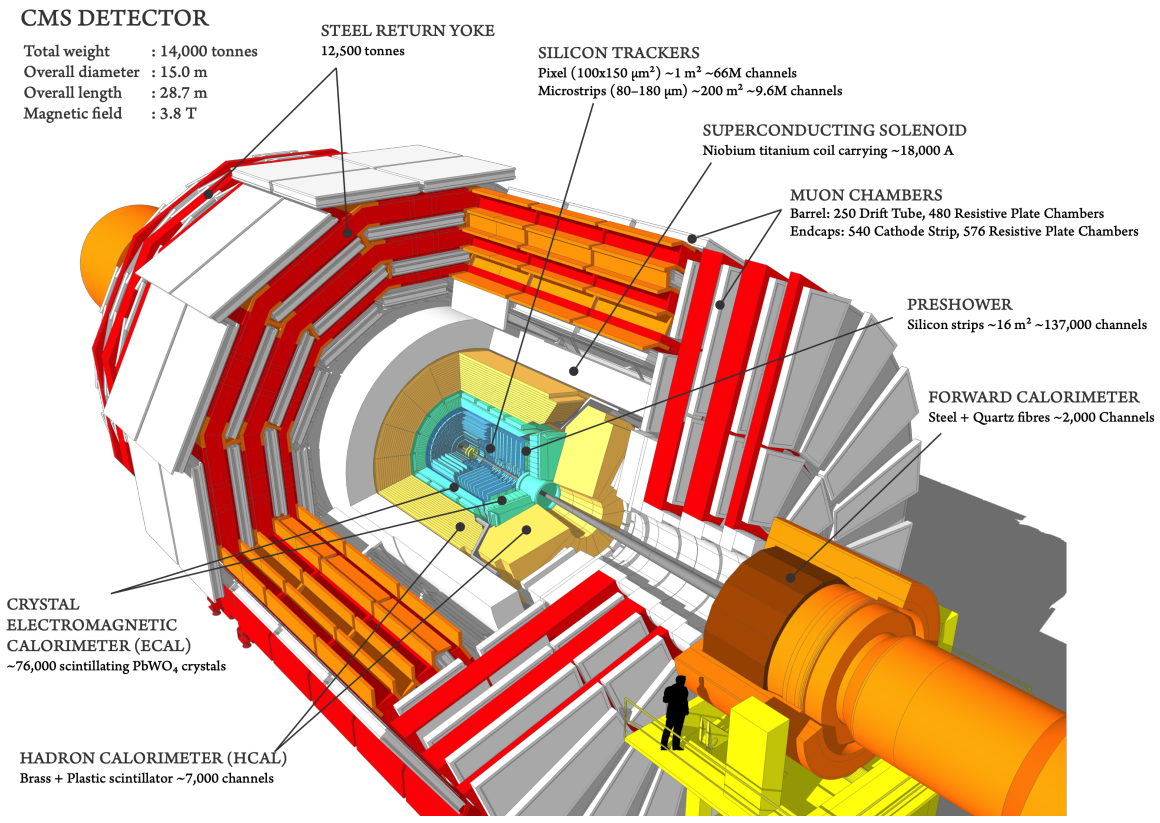


Figure 2.2. Illustration of the CMS detector and its main components [39].

2.1.1 Coordinate System and Conventions

The CMS uses a right-handed coordinate system, with the origin located at the nominal collision point (where particles are expected to collide). The z-axis points along the beam axis (toward the Jura mountains), the y-axis points vertically upwards, and the x-axis points inwards to the center of the collider ring, as shown in Figure 2.3.

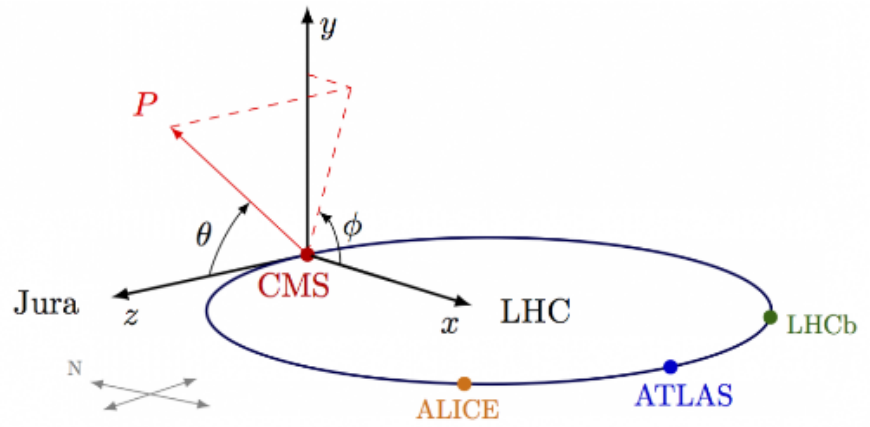


Figure 2.3. Visual representation of the cartesian and polar coordinate system used in CMS project [40].

Since the detector has a cylindrical structure, a polar coordinate system is also used. The azimuthal angle ϕ is measured from the x-axis in the x-y plane while the polar angle θ is measured from the z-axis. The polar angle is usually transformed into pseudorapidity, defined as

$$\eta = -ln\, tan\frac{\theta}{2} \tag{2.1}$$

which can have values from 0 at $\theta = \frac{\pi}{2}$ to $\pm\infty$ at $\theta = 0(\pi)$, as visible in Figure 2.4.

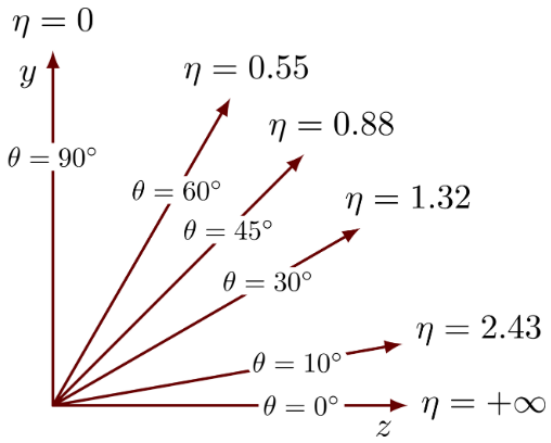


Figure 2.4. Relation between the polar angle θ and pseudorapidity η [40].

Another kinematic variable that is suitable for describing collisions in a CMS detector is the transverse momentum p_t, the component of momentum perpendicular (i.e. transverse) to the beam line, calculated with

$$p_t = \sqrt{p_x^2 + p_y^2} \tag{2.2}$$

where $p_t x$ and $p_t y$ are the components of the momentum of a particle along the x-axis and y-axis.

The angular distance R is the measure of the apparent separation between two particles expressed as

$$\Delta R = \sqrt{(\Delta\eta)^2 + (\Delta\phi^2)}. \tag{2.3}$$

Defined parameters provide crucial information about particle interactions and simplify the analysis of complex events.

2.1.2 Subdetectors

As it is visible from Figure 2.1 and Figure 2.2, CMS consists of layers of subdetectors that utilize different materials to capture and measure particle energy and momentum. Its main sub-detectors (going from the inside out) are an inner tracker, an Electromagnetic calorimeter (ECAL), a Hadronic calorimeter (HCAL), and muon chambers. The tracker, ECAL, and HCAL are all housed within the superconducting solenoid magnet, while the muon chambers are outside the magnet but within its magnetic field.

2.1.2.1 The Inner Tracker

The inner tracking system lies at the core of the detector setup surrounding the collision interaction point. The tracker comprises concentric layers of silicon sensors, each of which is completed by endcaps that extend its acceptance up to a pseudorapidity of $|\eta| < 2.5$. It comprises pixel and strip detector modules, capturing the spatial position of the particle passage. The pixel detector is made up of 66 million pixels, covering approximately 1 m^2 in total area. It is surrounded by the strip tracker, which has 10 million read-out channels and covers approximately 200 m^2 of total area [41]. Using the obtained data, it is possible to precisely and efficiently measure the trajectories of charged particles produced in LHC collisions and accurately reconstruct secondary vertices. The drawback of tracker technologies is the impossibility of detecting neutral particles, which is not the case with subsequent subdetector layers, the Electromagnetic calorimeter, followed by a Hadronic calorimeter.

2.1.2.2 Calorimeters

When particles collide, they create unstable particles that decay almost immediately after their creation before reaching the calorimeters. The rest of the decay products interact with

the calorimeter layers, producing cascades of low-energy secondary particles, known as particle showers. Over time, these low-energy particles are absorbed. They deposit energy into the calorimeters, which is then measured and analyzed.

The electromagnetic calorimeter is a hermetic, homogeneous calorimeter composed of fast and compact lead tungstate (PbWO4) crystals, which are mounted in the central barrel and two endcaps [42]. Each crystal is equipped with two silicon avalanche photodiodes on the barrel part and vacuum phototriodes on endcaps, used as photodetectors determining the energy and the position of the particles [42, 43]. To enhance the detection capabilities, a preshower detector is positioned in front of the endcap crystals. The main goal of the ECAL is to measure the energy of particles that interact electromagnetically. ECAL records two prominent cases: the first case is particles that get absorbed and release their full energy in ECAL, such as electrons, positrons, and photons. The second case involves charged and neutral hadrons, which can initiate hadronic showers in the ECAL and are then completely absorbed in the hadron calorimeter.

The Hadronic calorimeter is located behind the tracker and the ECAL. It measures the timing, energy, position, and angle of particles interacting hadronically, like protons, neutrons, pions, and kaons. The main parts of HCAL, the barrel and endcap, are located inside the magnetic field; both consist of alternating layers of brass and plastic scintillator plates as active material [44]. The outer and a forward hadronic calorimeter are extended beyond the magnet. The outer calorimeter utilizes the CMS magnet coil/cryostat, and the steel of the magnet return yoke as its absorber. The forward calorimeter is based on different technologies and is not discussed in this thesis.

2.1.2.3 The Muon Chambers

The detection of muons is of central importance to CMS because they are expected to be produced in the decay of a number of potential new particles. Muons are low-interacting particles that pass the calorimeters' layers without significant interaction, and having a lifetime long enough to leave CMS, they are detected in the outermost shell of CMS detector, muon chambers. They cover very large areas ($O(100\ m^2)$), characterized by low occupancy and low radiation levels. The CMS Muon System is composed of different types of gas detectors, namely, Drift Tubes (DTs), Cathode Strip Chambers (CSCs), RPCs, and Gas Electron Multipliers (GEMs). DTs can identify muon tracks in the barrel part of the detector, while CSCs are primarily used in the endcap disks where the magnetic field is uneven, and particle rates are higher than in the barrel. RPCs, on the other hand, are fast gaseous detectors that provide a muon trigger system parallel to those of the DTs and CSCs. Finally, GEMs complete the system, enabling optimum trigger performance with maximum selection efficiency of muons even in a high flux environment [42, 45].

2.1.3 Trigger System

Retaining all the events generated at CMS for analysis is not feasible due to limited storage and processing capabilities. Besides, most of the generated events are low-energy glancing collisions that do not lead to revealing new phenomena. Therefore, a data reduction system is necessary to drastically reduce the data rate by selecting only potentially interesting events, which can be saved for subsequent offline analysis.

Experiments described in [42, 46, 47] adopt a two-level approach in selecting events of potential physics interest. In contrast, the ATLAS trigger system employs a more complex three-level approach. The first step in CMS data selection is the Level-1 Trigger, implemented in custom-designed electronics, followed by the High-Level Trigger (HLT), a simplified version of the CMS offline reconstruction software that operates on a computer cluster. L1T reduces the incoming data stream from 40 million events/s (bunch crossing rate) to 100 thousand events/s by discarding over 98% of the generated events while the HLT additionally reduces the data flow to 1000 (0.025‰) events per second, Figure 2.5.

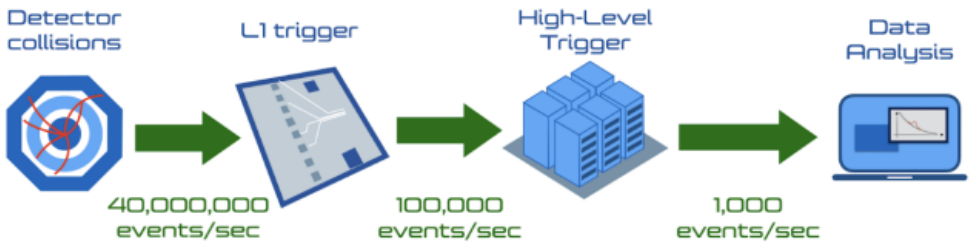


Figure 2.5. The data processing flow in CMS experiment: Forty million collisions are generated per second. The L1T selects 100 000 events, which are then passed to HLT. Only 1000 events per second are kept for further studies [48].

2.1.3.1 The First Level Trigger

The L1T is a real-time system, selecting events containing detector signals consistent with an electron, photon, muon, τ lepton, jet, or missing transverse energy. It is implemented in Application-Specific Integrated Circuits (ASIC) chips and FPGA devices, with a typical latency of $O(1)$ µs. There are several reasons why FPGAs are chosen for devices that run ML inference. First, the possibility of parallelism (many resources work on different parts of the problem simultaneously) makes them extremely efficient. Second, they exhibit predictable real-time latencies, which is of ultimate importance for L1T applications. Finally, they tend to consume less power than GPUs or CPUs while solving similar problems. Taking into account all the above and considering the ratio between flexibility and efficiency for each of the available architectures (Figure 2.6), it has been decided that FPGAs are the best choice for the L1T architecture.

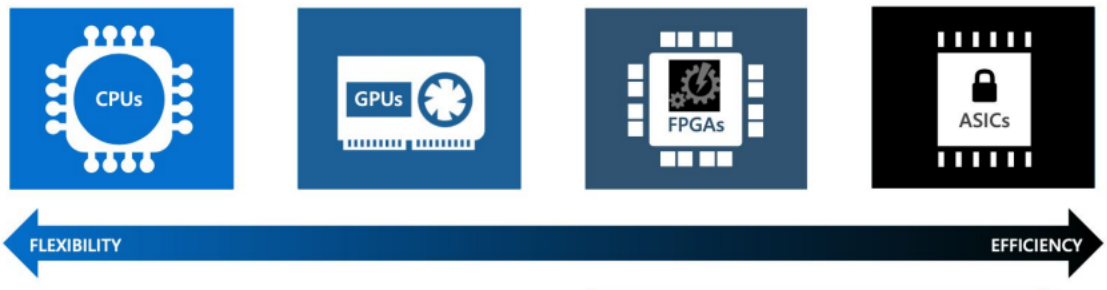


Figure 2.6. Comparison of different computing architectures: flexibility vs. efficiency [28].

The L1T relies on coarsely segmented data collected from the calorimeters and the muon system to make rapid decisions. The high-resolution data is stored in pipelined memories in the front-end electronics for later analysis. This approach allows the L1T to quickly and efficiently filter out uninteresting events while maintaining high precision through the use of fine-grained data in later stages of analysis.

The L1T system is divided into local, regional, and global components, after which the Global Trigger makes the final decision.

2.1.3.2 High-Level Trigger

Events that pass through L1T are processed by a High-Level Trigger (HLT) consisting of computer farms. The HLT system can access the entire readout, enabling it to perform intricate computations on precise and complete data sets. This allows for the execution of complex analysis similar to that carried out in offline software. Since HLT algorithms are not included in this thesis topic, the HLT system is not described here. More information may be found in [49, 50].

2.1.4 The CMS Upgrades for HL-LHC

To exploit the complete physics potential of the HL-LHC and simultaneously deal with the huge amount of generated data and unprecedented amount of radiation, LHC detectors, including CMS, are going through extensive upgrades during the shutdown periods. The high radiation doses in the HL-LHC environment require radiation-hard technologies to prevent equipment damage. Also, there is a need to develop strategies for dealing with the challenges posed by a high PU environment, including high occupancies, high data rates, and complex event reconstruction. The requested specifications have highlighted the necessity to enhance the energy resolution of jets in calorimeters. The CMS system utilizes the idea of "particle flow" (PF) to reconstruct events: individual particles, such as electrons, muons, photons, and hadrons, are identified and reconstructed by combining information from all detector parts [51, 52]. The CMS experiment relies heavily on the PF flow event-reconstruction algorithm

in most of its analyses. The PF algorithm requires comprehensive and accurate measurement of the energy and momentum of each particle necessary in the PF steps. The first step (for charged particles) is track reconstruction. Then follows the clustering of energy deposits in the calorimeters and linking them to particular reconstructed tracks. Characteristics of the determined clusters can be essential for some particle identification, while, for example, muon identification requires a combination of information coming from tracker and muon chambers. It is evident that two crucial elements for particle flow are accurate tracking of charged particles in the inner detector and high-resolution calorimetry to differentiate contributions from nearby particles. The L1T system used in previous phases of the project does not meet the required properties.

Some of the most important CMS detector upgrades implemented to ensure valid performance in the HL-LHC phase include:

- **Tracker**

  If the integrated luminosity exceeds $500 fb^{-1}$, the current tracker settings' performances will be seriously degraded. Therefore, a new tracker is needed that has increased granularity and extended coverage up to $|\eta| \sim 4$ and is capable of efficient tracking in L1T.

- **Calorimeters**

  The current structure of the calorimeter endcap is insufficient to handle the anticipated high levels of radiation and event accumulation. The existing forward calorimeters, the electromagnetic calorimeter, and the hadron calorimeter, were designed for an integrated luminosity six times lower than the one planned in the HL-LHC phase [53]. Under new conditions, these settings expect performance degradation, leading to an unacceptable loss of discoveries in physics. Therefore, each endcap will be replaced with a new HGCAL [54]. This update will enable 3D shower imaging and more precise timing, making it the most significant improvement. Unlike homogeneous calorimeters, where the same medium serves as both absorber and detector, sampling calorimeters consist of layers of passive absorber that alternate with the active detector layers. The HGCAL is an advanced sampling calorimeter that uses silicon and plastic scintillators as its active components and Lead, Copper-Tungsten, Copper, and Stainless Steel as absorbers. A schematic view of the HGCAL is shown in Figure 2.7.

  The material, layout, and thickness of the sensors in the HGCAL layers vary depending on their position. Active silicon sensors will be used in high-radiation areas with high track density. Hence, the entire electromagnetic segment (CE-E) and the inner hadronic compartment ($|\eta| > 2.4$) will be constructed using silicon sensors. The sensors are hexagonal on 8-inch wafers, maximizing sensor area and allowing for full coverage tiling. In the CE-E area, copper, copper-tungsten, and lead are used as ab-

sorbers. Plastic scintillator tiles will be used for the lower radiation region of CE-H with steel as an absorber and readout by on-tile silicon photomultipliers (SiPMs).

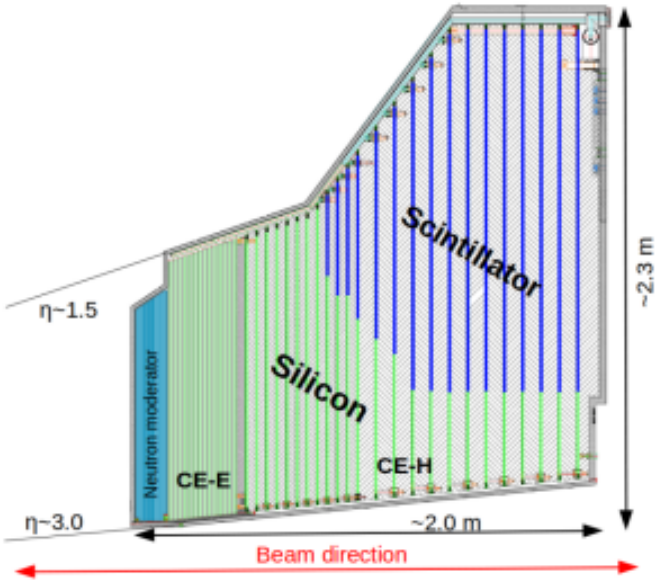


Figure 2.7. A schematic view of the HGCAL: it covers a pseudorapidity range from 1.5 to 3. CE-E area is covered with silicon sensors, the same as the high-radiation part of CE-H. Scintillator tiles are used for the low radiation part of CE-H [55].

The CE-E consists of 28 layers altogether per endcap. In comparison, the hadronic part consists of 22 layers per endcap, 8 of which are Si layers and 14 are Si + scintillator mixed layers. Thin sensors with 120 µm active thickness will be used within a 70 cm radius from the beam axis on silicon disks. There are over 400 individual cells with a size of 0.5 cm^2 in the high-density layout of those sensors. At a distance greater than 70 cm from the beam, the sensor's active thickness increases to 200-300 µm, and there are approximately 200 individual cells. Physically thinned Float Zone (FZ) silicon is used for 200 and 300 µm sensors, while 120 µm sensors are epitaxial with a thick handle wafer. The HGCAL endcaps will consist of 620 m^2 of silicon and 400 m^2 of scintillator material, with over 6 million readout channels. In the recent HGCAL design revision, the total number of layers was reduced to 47 [56] but could also change till actual production. The described calorimeter design ensures a high longitudinal and transverse granularity required for the particle flow algorithms. To mitigate the effects of increased sensor leakage current caused by irradiation, the HGCAL will be kept at a low temperature of -30° C. An overview of the main parameters is given in Table 2.1.

Table 2.1. HGCAL active material parameters.

Both endcaps	Silicon	Scintillators
Area	$\sim 620\ m^2$	$\sim 400\ m^2$
# Modules	~ 27000	~ 4000
Channel size	$0.5-1\ cm^2$	$4-30\ cm^2$
# Channels	$\sim 6M$	$\sim 240k$
Op. temp.	$-30^\circ\ C$	$-30^\circ\ C$

2.1.4.1 The HGCAL Trigger Architecture and Data processing

Although the exact design of the system is still unknown, it is planned that the initial data processing will be carried out within the on-detector ASICs to minimize the amount of data that needs to be transmitted off the detector. The second stage of data processing, which is associated with reconstruction algorithms such as energy clustering and pile-up estimation, will be executed through off-detector FPGAs.

To efficiently transfer trigger data out of the detector, it is necessary to reduce the data size by a factor of approximately 20. This reduction is achieved through various techniques and methods, which ensure that the data is compressed without losing any critical information. The implementation of such techniques is vital to facilitate the smooth and reliable transfer of trigger data in a limited bandwidth scenario. The reduction techniques are [57]:

- The timing information is discarded, and both the dynamic range and resolution of measured energies are reduced.
- An additional reduction is provided by sending only alternate layers in the CE-E.
- The sensor cells are grouped into larger trigger cells (TCs).
- The system selects and sends the most energetic trigger cells off the detector using a mixture of optical and electrical links, ensuring the accuracy and efficiency of the process.

The visualization of the last two steps is shown in Figure 2.8

After completing the final step mentioned earlier, the 3D clustering algorithm is performed on a full depth view of the detector [47]. On that way, both longitudinal and transverse shower profiles specific to each physics object are exploited in cluster generation, resulting in a detailed shower "image". With the created 3D clusters, it is possible to classify them as physically interesting (EM or hadronic) or irrelevant (PU).

The conditions under which the experiment takes place, the exposure to radiation, the speed of data flow, and the limited computer resources available for the primary data selection make their processing even more difficult. The distance of the control room from the

location of the experiment also affects the possibilities of measurement, recording, and data transfer.

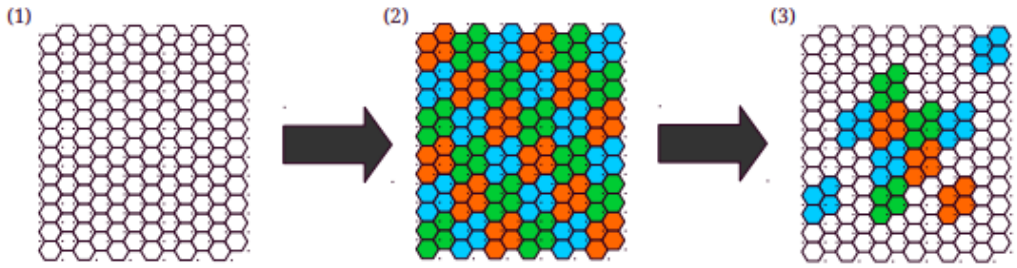


Figure 2.8. Illustration depicting the process of trigger data reduction that occurs in the front-end ASICs. The sensor cells, shown in (1), are grouped into trigger cells represented by (2). During the data reduction process, only the most energetic cells, indicated by (3), are selected and transferred for further processing [57].

The CMS L1 new trigger system is designed to take advantage of the new features provided by the upgraded sub-detectors in order to maintain a high efficiency of physics event selection in the incoming HL-LHC phase. The illustrative layout of the architecture and data flow of the Phase-2 trigger system is shown in Figure 2.9. With a latency of 12.5 µs, not only information from the calorimeters and muon detectors is utilized, but also the information from the new tracker and HGCAL can be incorporated. As visible on Figure 2.9, upgraded L1T receives inputs from the calorimeters, the muon spectrometers, and the track finder. The calorimeter trigger inputs include data from the barrel calorimeter (BC), the HGCAL, and the hadron forward calorimeter (HF). It is composed of a barrel calorimeter trigger (BCT) and a global calorimeter trigger (GCT). The muon trigger receives data from various detectors, including DT, RPC, CSC, and GEM. It comprises a barrel layer-1 processor and muon track finders that process data from three separate pseudorapidity regions. These are referred to as BMTF, OMTF, and EMTF for the barrel, overlap, and endcap, respectively. The muon track finders send their muon candidates to the global muon trigger (GMT) for possible combination with tracking information. The track finder (TF) distributes tracks to various parts of the design, including the global track trigger (GTT). The correlator trigger (CT) in the center (yellow area) consists of two layers dedicated to particle-flow reconstruction. All objects are directed to the global trigger (GT) to make the final L1 trigger decision. External triggers, such as inputs from the MIP Timing Detector (MTD), flowing into the GT are also shown. The components being developed within the Phase-2 L1T project are all located in the same area, which is highlighted in blue. The different stages of processing are shown on the right: trigger primitives, local and global trigger reconstruction, particle-flow trigger reconstruction (PF), and global decision.

To ensure the updated CMS meets the conditions and features needed for efficient functioning in the HL-LHC era, the CMS collaboration has been actively testing prototype de-

tector modules, including HGCAL. These tests have helped to refine the design and functionality of HGCAL, making it one of the most advanced detector modules available today. To confirm the effectiveness of the planned changes, the detector components and their associated electronics are extensively tested in test bench-based experiments and beam test experiments with single particles [58–61]. In particle physics, a test beam is utilized to calibrate and study particle detectors prior to their deployment in large-scale experiments. It provides a controlled environment to study the detector's response to known particles of specific energies.

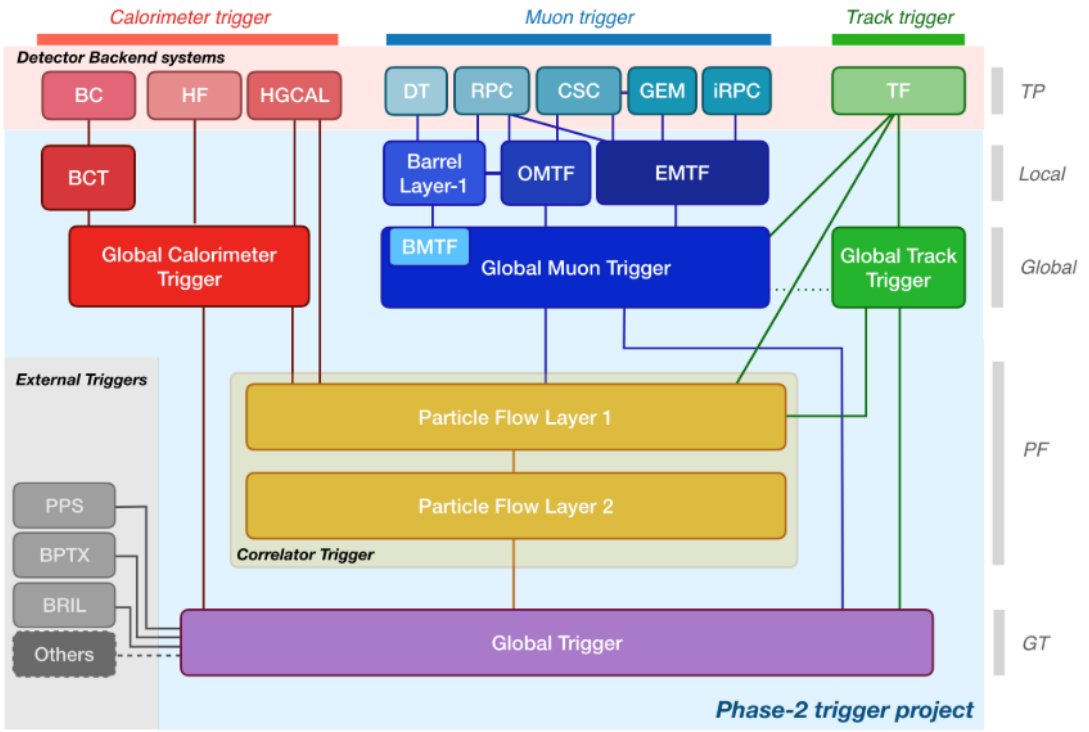


Figure 2.9. Schematic overview of the CMS L1 Phase-2 enhanced trigger architecture [47].

2.2 Machine Learning in High Energy Physics

The large-scale experiments, such as those conducted at the LHC, are characterized by vast amounts of data and intricate event structures. Traditional analytical approaches, which rely on manual exploration and interpretation of data, are facing challenges in handling the size and complexities of these datasets. In recent years, ML techniques have revolutionized the way of data analysis, providing unprecedented abilities to detect complex patterns and extract valuable insights [62–65]. Advancements in specialized hardware development have also played a crucial role in improving the efficiency and accuracy of data selection [66, 67]. This hardware can be effectively integrated with advanced ML algorithms, bringing numerous benefits, including faster and more accurate data analysis, improved decision-making, and

enhanced efficiency in data management. In summary, the combination of ML techniques and specialized hardware has transformed the field of HEP research, enabling researchers to uncover new insights and make revolutionary discoveries that would have been impossible using traditional methods [29, 30, 68–70].

2.2.1 Fundamental ML Principles

Artificial intelligence (AI) is an area of study that focuses on developing computers with human-like cognitive abilities like analyzing data, recognizing images, understanding human language, making predictions, and much more. ML is a subset of AI based on extracting knowledge from data and enabling computers to learn from it autonomously or semi-autonomously. Arthur Samuel wrote about "the field of study that gives computers the ability to learn without being explicitly programmed" already in 1959 [71].

In contrast to traditional algorithms that are developed based on domain knowledge and then implemented in software to analyze data, ML algorithms learn directly from data and often produce the desired results. Utilizing the power of ML, researchers can quickly identify crucial features and correlations in large datasets, leading to more accurate predictions and valuable insights. ML techniques involve training algorithms to recognize patterns and make predictions based on data, making them highly effective for analyzing high volumes of data. After the completion of model training, it can be used to predict new data.

Depending on the structure of the training dataset, ML techniques can be classified into two main categories, as shown in Figure 2.10:

1. **Supervised Learning**: in this technique, labeled data is used. The data set is usually partitioned into two subsets, namely the training set and the testing set. The model is trained on the training set, where a unique data chunk is used for each training step to estimate a model's predictive power. When the training is finished, the model can be used to predict the output of new data, as shown in Figure 2.11. This process uses a testing set and is less computationally intensive. It is called inference, and it provides insight into how well a model can generalize to new data. Supervised learning can be separated into two types of problems: classification and regression. Classification is used to assign labels to data based on patterns learned from training data. In binary classification, data is categorized into two distinct groups, while in multi-class classification, data is sorted into one of several possible classes. On the other hand, regression is used to predict a continuous output variable based on input features by establishing a mathematical relationship between dependent and independent variables.

2. **Unsupervised Learning**: In this technique, there is no label connected with input data. The model learns to identify patterns and relationships in the data and, based on learned information, groups similar data points together, as illustrated in Figure 2.11. This technique can be helpful in discovering hidden patterns and structures in data.

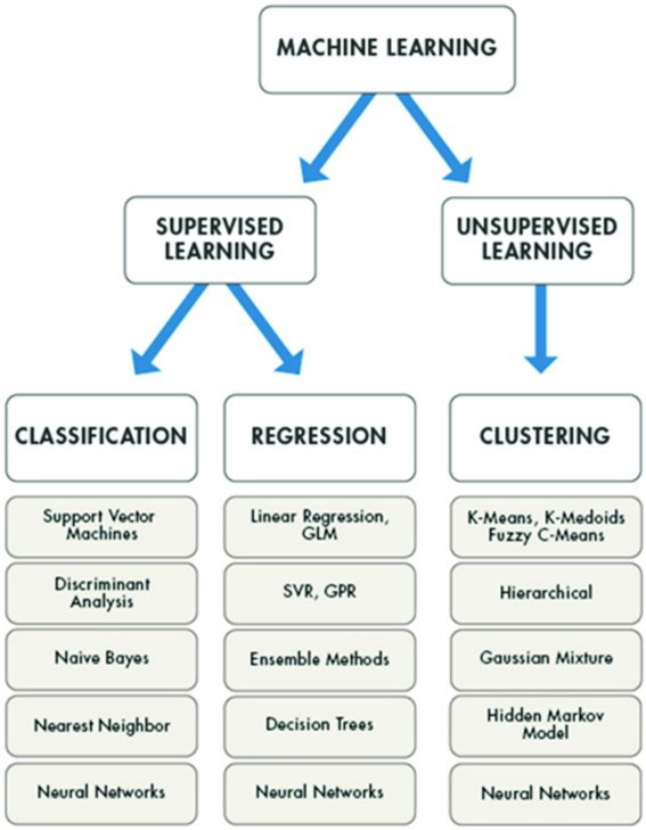


Figure 2.10. Classification of ML algorithms [72].

Furthermore, **Semi-Supervised Learning** combines the techniques described earlier. This training process relies on a small amount of labeled data and a larger amount of unlabeled data. When there is a shortage of labeled data, employing this approach can enhance the performance of machine learning models.

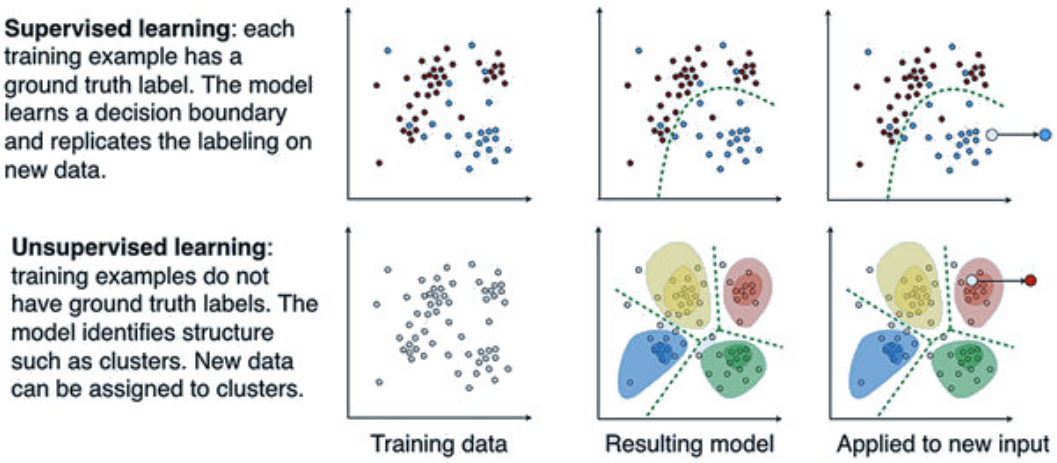


Figure 2.11. Comparison of supervised and unsupervised learning [73].

Overall, ML techniques have found the usage in various industries such as healthcare [74, 75], finance [76, 77], transportation [78], and many others. In this thesis, a supervised learning approach is implemented, encompassing both binary and multiclass classification.

2.2.2 Challenges

A potential issue with ML models is overfitting (Figure 2.12), where the model is excessively complex and accurately fits the training set but struggles to generalize to new data [79]. There are several reasons why overfitting may occur, like the presence of noise, the limited size of the training set, and the complexity of classifiers.

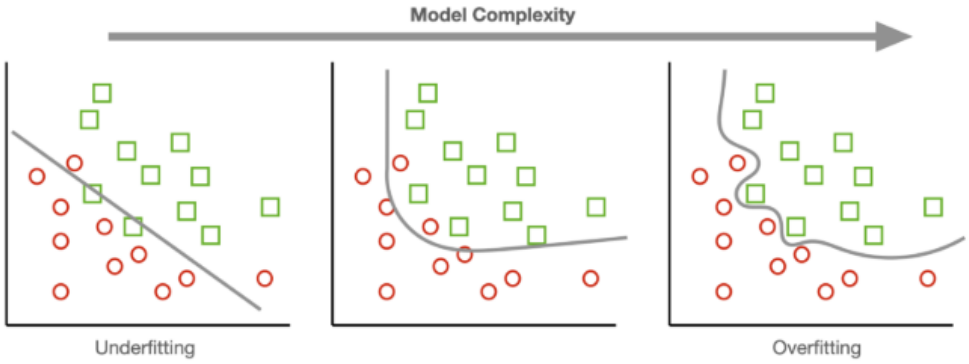


Figure 2.12. The relationship between model fitting and model complexity: the fitted model is depicted as a grey line separating two classes in the data: red circles and green squares. Overfitted models can fit to noise in the training data, while underfitted models may miss important details [80].

A range of strategies can be employed to prevent overfitting, including cross-validation, early stopping, data expansion, and regularization [81]. There are different types of cross-validation [82–84], and the main concept involves partitioning the dataset into subsets, training the model on some subsets, and evaluating its performance on the remaining subsets. The early stopping strategy is introduced to prevent overfitting by stopping training before performance optimization stops [85]. The data expansion approach aims to increase the size of the dataset. By expanding the dataset, the model is trained on more diverse data, which can result in better generalization and robustness. By using this strategy, NN hyperparameters can be fine-tuned more effectively, resulting in an improved model performance. The regularization strategy aims to limit the effect of useless features on the model. As more features are added to the model, its complexity increases as well. However, an overfitted model tends to incorporate all the features, regardless of their effect on the final output. To prevent this, it is crucial to reduce model complexity by identifying and excluding low-impact features, which will improve its ability to generalize. This allows the model to concentrate on the most impactful features, resulting in accurate predictions and improved performance. To prevent overfitting and improve generalization, a penalty term (regularizer) is added to the

cost function to penalize large coefficients. Some commonly used regularization methods are:

- **L1 Regularization (Lasso)**

  L1 regularization adds the L1 norm values of the coefficients as a penalty term to the loss function, which encourages sparsity in the model. This sparsity causes some coefficients to become exactly zero, effectively selecting a subset of features.

- **L2 Regularization (Ridge)**

  L2 regularization adds the squared values of the coefficients as a penalty term to the loss function. It penalizes large coefficients but does not lead to sparsity, allowing all features to be retained. By implementing the technique, it is possible to prevent multicollinearity and improve model stability.

ML algorithms need a lot of data to train and learn patterns well. Having more data helps to create a better model that can handle complex problems and perform well on new data. However, obtaining access to an extensive dataset is not always feasible. To address this, various methods like data augmentation and creating synthetic data are used. Synthetic data can be generated through simulations, algorithms like GANs (Generative Adversarial Networks), or by augmenting existing datasets. This helps strengthen the training process and improve the model's robustness.

2.2.3 Baseline Particle Classification Methods in HEP

As baseline methods to be compared to NN-based approaches proposed in this thesis, the simple cut-off and two ML methods, Support Vector Machine (SVM) and Random Forest (RF), are applied. The methods mentioned have typically been used in HEP for different tasks, including triggering [86–90]. However, when comparing and interpreting the results, it is important to consider that there are significant differences in technology, spatial resolution, and data characteristics between the proposed approach for the HGCAL under development and the existing approach used for triggering algorithms.

Further, this section analyses the simple cut-off approach and the diverse ML tools applied in the subsequent analyses. It explains some basic ML concepts and algorithms used in the thesis. ML methods will be adapted to handle HEP data, which requires sophisticated approaches to distinguish between signal and background events and identify rare processes.

2.2.3.1 Cut-off

In the feature space, linear classification is the simplest form of classification where the borders are represented by straight lines. This is a commonly used selection method in HEP named a cut-based analysis [91–93], which has the advantage of being easy to interpret. In

the process of selecting events, it is necessary to identify the physical quantities that are most distinguishing and apply cuts on those variables or their combinations. The selection process involves a series of cuts, which can be represented in tables or plots called "Cut-Flows" [93].

The primary advantage of this method is its simplicity in implementation and understanding, which makes it efficient for quickly filtering data in real-time analyses. However, in situations where the characteristics of the signal and background are similar, its inflexibility leads to low discrimination accuracy.

2.2.3.2 Support Vector Machine

Support Vector Machine (SVM) is a supervised ML algorithm used for solving complex classification and regression problems invented by Cortes and Vapnik in 1995 [94]. The aim of the SVM algorithm is to find a hyperplane in an N-dimensional space (where N represents the number of features) that effectively classifies the given data points. Two classes of data points could be separated with multiple hyperplanes, but the objective is to find a hyperplane that has the longest distance between the data points of both classes. That distance is named margin, and by maximizing it, the classifier can better generalize the future data. Support vectors are data points critical for the construction of the hyperplane. Those are the points from each class that are the closest to the hyperplane, and by using these support vectors, the margin is maximized, which increases the accuracy of the classifier. The illustration of the SVM algorithm is given in Figure 2.13.

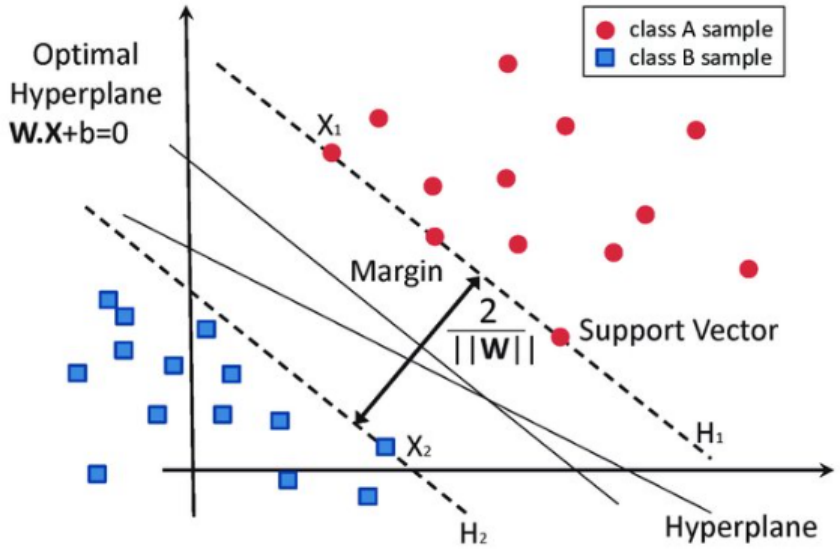


Figure 2.13. Illustration of SVM algorithm: optimal hyperplane ($WX + b = 0$) that effectively separates different data classes [95]. The largest margin $\frac{2}{\|W\|}$ can be obtained by maximizing the distance between the hyperplane and its closest point.

There are two types of SVM algorithms:

- **linear SVM**, suitable when the classes can be separated by a hyperplane. It assumes that the relationship between the features and the output classes is linear.

- **non-linear SVM**, also known as kernel SVM, allows it to find non-linear decision boundaries. It is appropriate to use when the relationship between the input features and the output classes is non-linear, as it can capture more complex relationships present in the data.

The choice between linear and non-linear SVM depends on the characteristics of the data and the problem we are dealing with. Although a nonlinear kernel often has better predictive performance, in some cases, a linear kernel needs consideration, as solving the optimization problem for a linear kernel is much faster. According to [96], using the linear kernel is sufficient if the number of features is large, as the nonlinear mapping does not provide any improvement in the performance.

In the experiment, an open-source ML library for the Python programming language named *scikit-learn* [97] is used for testing SVM classifiers. The first group of tests will be conducted using default settings for each of the kernel values: *linear*, *poly* (degree=3), *sigmoid*, and *rbf* with default settings. Subsequently, to improve model performance, the *GridSearch* will be used to find the optimal hyperparameters within the model. The parameters that will be used for model optimization are:

- ***C***: a regularization parameter with a default value of 1, controlling the trade-off between maximizing the margin and minimizing the classification error. The magnitude of the regularization is inversely related to the value of C: a smaller C value results in a wider margin, which leads to lower accuracy. When using a larger C value, the margin is more constrained, leading to a classification of more training examples correctly. However, this may potentially result in overfitting.

- ***gamma***: a kernel coefficient for '*rbf*', '*poly*', and '*sigmoid*' with a default value '*scale*' calculated as $\frac{1}{n\_features * X.var()}$. The high value of gamma values results in more complex decision boundaries but may lead to overfitting.

- ***kernel***: specifies the kernel type to be used in the algorithm.

SVM has become a widely adopted method for various applications due to its high accuracy and low computational requirements. In the HEP field, authors in [87] use SVM in search for the case of Supersymmetry at the LHC. In [88], it is used for analysis of top quark production. Already in 1999 in [98] authors used SVM and MLP for charm quark tagging and muon identification and got consistent results. SVM algorithm used to reduce the multi-jet background events in [99] achieved classification accuracy $\geq 95\%$. The application of SVM to the search for a classification of heavy quark jets is presented in [100], while Vossen's work in [101] provides insights into using SVM in HEP.

2.2.3.3 Random Forrest

Random Forest (RF) is a popular supervised ML algorithm introduced by Leo Breiman [102]. Based on the concept of ensemble learning[1], RF combines the outputs of multiple decision trees to generate a single result. The ensemble technique used in RF is Bagging (also known as Bootstrap Aggregation): decision trees are trained on various subsets of the given dataset. The final output is generated using prediction from each tree and based on the majority votes of predictions. The illustration of the RF algorithm is given in Figure 2.14 In

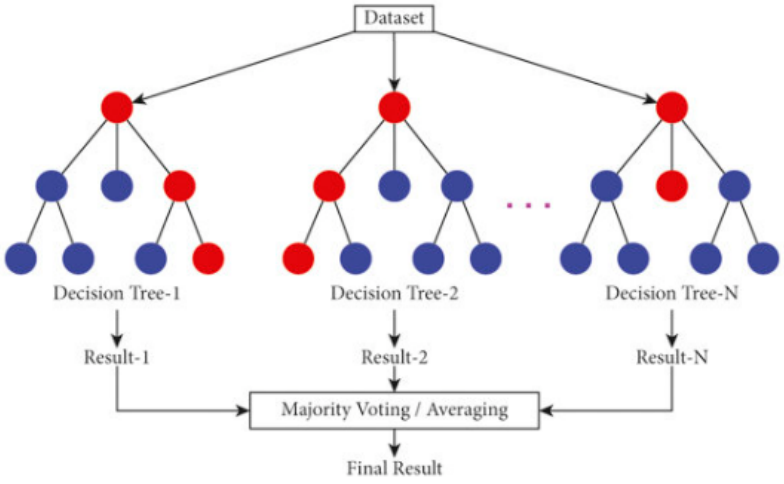


Figure 2.14. Illustration of RF algorithm: a number of decision trees are created. A final result is created by aggregating the predictions of all the trees [103].

the RF learning process, there are two major sources of randomness that reduce overfitting and improve the performance and robustness of the model. The first random component is the use of a random (bootstrapped) sample of the original training data set for each individual decision tree. It means, in the vast majority of cases, the training sets for the trees are different from one another, reducing the correlation between trees and improving the generalization of the predictions. Another element of randomness is in a (random) selection of the features considered at each node. Many good properties contribute to its widespread usage [104]: it is relatively fast to train and predict, simple to tune, can handle high-dimensional problems directly, and can be easily implemented in parallel. Aside from that, RF is a multipurpose tool; it can handle classification and regression problems, including multiclass classification. It is considered one of the most efficient algorithms, requiring almost no tuning and still providing fine control over the model that is learned.

ML library *scikit-learn* [97] is used for testing RF classifiers. It offers a robust implementation combining both algorithmic and code optimizations. Tunable parameters are available

[1]The good performance of the model is achieved by combining multiple classifiers to solve a complex problem and to improve the performance of the model.

to improve the accuracy of the RF classifier in specific situations. The hyperparameter optimization is done using the grid search, employing the next hyperparameter search space:

- ***n-estimators***: the number of trees in the forest, with default value *100*. Increasing this hyperparameter generally improves the performance of the model but also increases the computational cost of training and predicting.
- ***max_features***: maximum number of features RF considers looking for the best split, the default value is '*sqrt*' (of number of features).
- ***max_depth***: maximum number of levels in each decision tree. The default value is *None*, which means that nodes are expanded until all leaves are pure or until all leaves contain less than 2 samples.
- ***criterion***: this parameter measures the quality of the split, with default value '*gini*'.

Understanding the contribution of each feature to the (RF) predictions is typically important. By analyzing the contribution of each feature, it is possible to identify the most significant ones and eventually optimize the RF model to achieve more accurate predictions. *Scikit-learn* has a built-in method for feature importance in RF. This method can extract the most important features that contribute to the model's accuracy. Permutation feature importance is another useful *scikit-learn* method that estimates the importance of each feature by measuring the model's performance change when the feature's values are randomly shuffled. In the experiment, this is not applicable, as features are pixels corresponding to energy deposits in the particular HGCAL layer. Boruta [105] is an extension of the RF algorithm that iteratively removes the features that are proved by a statistical test to be less relevant than random probes.

RF has broad applications in HEP: in [89], it is used for particle identification in the ALICE experiment, [106] describes RF-based multiclass classification for highly skewed particle data,in [90] it is used for Higgs boson discovery. In [107], authors use an adaptation of RF algorithm to solve the problem of pulsar candidates detection. The model's performance is optimized by increasing the number of trees, resulting in an AUC of 0.9999.

2.2.4 Neural Networks

Artificial neural networks (ANNs) or, shortly, neural networks (NNs) are specific ML algorithms inspired by the complex workings of the human brain. Just like the human brain consists of interconnected neurons, ANNs consist of interconnected nodes that process and transmit information. These algorithms have the ability to learn from data, recognize complex patterns, and make decisions. By mimicking the brain's structure and functionality, ANNs have significantly contributed to the advancement of AI, enabling the development of systems that can perform tasks such as pattern recognition, prediction, and decision-making.

2.2.4.1 History and Evolution of Neural Networks

The possibility of learning from experience and the high data processing speed of the human brain were the main motivations for achieving human-like artificial intelligence. According to [108], historians and scientists alike often give credit to neuropsychiatrist Warren McCulloch and mathematician Walter Pitts for the first effort to mimic the workings of the human brain using computers in the 1940s. As a part of the research on the neurophysical characteristics of living beings, they developed a model called the Threshold Logic Unit (TLU). The TLU model "imitates" a biological neuron in the following way: the input signals are numerical values, the strength of the synapse is expressed by the weight factor w, which is multiplied by the signals and summed up in the cell body; if the obtained amount is above the defined threshold, the neuron gives an output signal. Inspired by the work of his predecessors, in [109, 110], Rosenblatt presented the classic perceptron model ten years later – the simplest form of an artificial neuron. It was later continued to be developed by Minsky and Papert, and their model was accepted under the name perceptron [111]. Hence, the term perceptron, notional as a mathematical model of a biological neuron, refers to a type of NN developed in the early days of AI. The main drawback of the perceptron is its inability to learn functions that are not linearly separable.

The perceptron, together with the artificial neuron, is a fundamental concept that plays a crucial role in the field of artificial neural networks. The perceptron processes the input signal using a linear model and produces an output signal, often used in binary classification. On the other hand, an artificial neuron uses some activation function before it produces an output, as shown in Figure 2.15. While the terms "perceptron" and "neuron" are sometimes used interchangeably, it is evident that "neuron" is a more general and complex term.

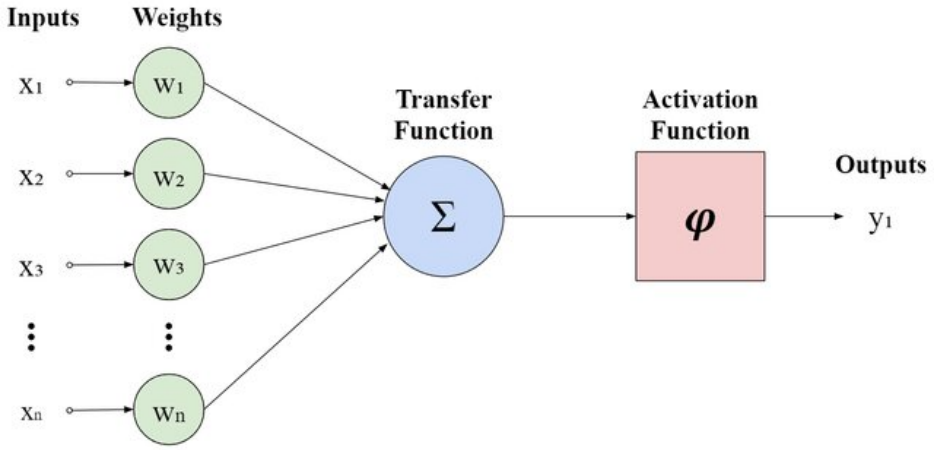


Figure 2.15. Visualization of the artificial neuron: it consists of input layer, one hidden layer, and output layer [112].

The described structure (after a process named training) is capable of solving simple classification problems, like binary classification. The parametric formula of an artificial

neuron is

$$y = \varphi(\sum_{i=1}^{n} x_i w_i + b), \tag{2.4}$$

where x is the input vector, w is the weight vector, b is the bias, and φ is the non-linear function. Weights and bias are parameters of the model that are learned during the training process. The hyperplane defined by the weights $\sum_{i=1}^{n} x_i w_i + b = 0$ creates two sub-spaces: input vectors laying on the same side of the hyperplane have the same sign. Using a specific activation function, such as sigmoid, the positive and negative values are transformed to 1 or 0, which represents the solution to the classification problem. As shown in [113], some functions, such as XOR, cannot be treated by single-layer perception. However, when the problem becomes complicated, neurons can be organized into multiple layers to form more complex neural network structures like MLP, CNN, recurrent neural networks (RNN), and many others.

2.2.4.2 Multi Layer Perceptron (MLP)

An MLP is (for historical reasons) another name for a feedforward artificial neural network. It is the most basic NN model consisting of fully connected neurons with a nonlinear activation function. According to Hornik [114], MLP is a universal function approximation; respectively, MLP has the ability (theoretically) to approximate any continuous function to an arbitrary level of accuracy using a sufficiently large number of parameters.

As shown in Figure 2.16, MLP consists of at least three main layers: an input layer, one or more hidden layers, and an output layer. The number of nodes in the input layer is determined by the complexity of the data that needs to be processed. Further, the number of output nodes depends on the number of classes the neural network should predict. The most challenging aspect of modeling a NN is determining the number and the structure of hidden layers, which depends on the complexity of the data and the problem to be solved, and is often target of optimization. In the case that MLP has two or more hidden layers, it is called a deep neural network. Each hidden layer consists of multiple neurons that receive inputs from the previous layer. Those inputs are weighted and summed together. The activation function is applied to the calculated weighted sum to enable NN to model nonlinear relationships between the inputs and outputs.

The process of training an NN involves a learning algorithm that iteratively adjusts the values of weights and biases. Before training an NN, the weights and biases are (randomly) initialized. In the forward propagation process, the input data is fed forward through the network to generate a prediction. The cost function, the difference between the calculated prediction and the ground truth from the training data, gives the error that the model made in its prediction. In backpropagation, to improve the model accuracy, the cost function is minimized using an optimization algorithm known as gradient descent. Gradually, in several steps, the network refines weights/biases values and improves performance.

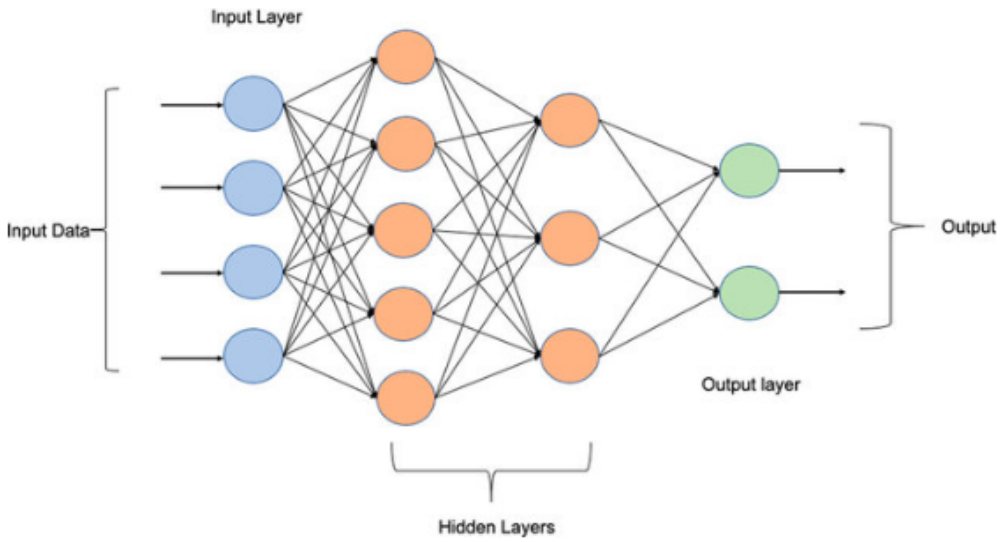


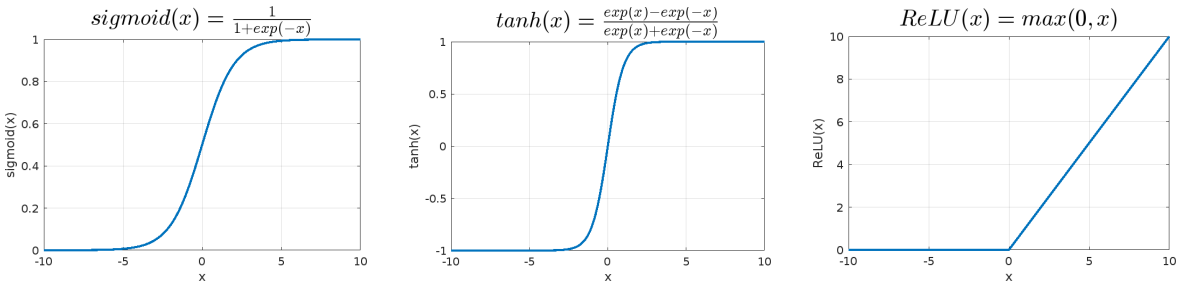
Figure 2.16. Illustration of the basic architecture of MLP: it consists of an input layer, one or more hidden layers, and an output layer [115].

Choosing the appropriate activation function is essential for ensuring effective training and performance of NNs. Several non-linear activation functions are commonly used (Figure 2.17):

- **sigmoid**: it is one of the most widely used activation functions. Sigmoid maps any input value to a range between 0 and 1, making it useful for binary classification tasks. For very large or very small input values, the gradient becomes very small, which can slow down or even stop the training process. This problem is known as the Vanishing Gradient Problem.
- **tanh**: it maps inputs between -1 and 1 and often performs better than sigmoid for hidden layers due to its zero-centered output and stronger gradient. The main disadvantage of the function is that it can also suffer from the vanishing gradient problem for very large input values.
- **Rectified Linear Unit (ReLU)**: it is the most widely used because it is computationally cheaper than alternatives and effectively solves the vanishing gradient problem during training.

2.2.4.3 Convolutional Neural Network (CNN)

In 1989, LeCun in [116] described LeNet-5, the CNN model used for the task of handwritten digit recognition. At that time, NNs were mostly applied in the scientific context, and their effectiveness in solving real-life problems was demonstrated through this work. More than



(a) Sigmoid *(b) Tanh* *(c) ReLU*

Figure 2.17. Activation functions.

a decade later, in 2012, in a [117], the authors showcased the incredible capabilities of deep CNNs in accurately classifying large-scale image datasets. Today, the paper is considered one of the breakthrough scientific papers in the ML field. Their findings revolutionized image classification and set a new standard for state-of-the-art techniques in this field. In the following years, various CNN architectures were introduced, using different design principles and optimization techniques that have transformed the field of CV,

Today, convolutional neural networks are widely used in computer vision problems such as image classification, object detection, and image segmentation. CNNs usually consist of one or more convolutional layers, followed by a pooling layer, while the network ends with one or more fully connected layers, as visible in Figure 2.18. Convolutional layers contain small matrices named filters/kernels that slide over the input image and perform convolution to learn features from input images. Each filter is designed to identify specific patterns or features, like edges, textures, or more complex structures. Pooling layers downsample the spatial dimensions of the input feature maps. It reduces computational costs in subsequent layers and makes detecting features computationally efficient. Another advantage is it helps avoid overfitting by reducing the number of parameters in the model. The two most used types of pooling layers are max pooling, which selects the maximum value in each region, and average pooling, which computes the average value. Classification is done in the fully connected layer at the end of the network.

In addition to the difference in the way data is processed, MLP and CNN require distinct types of input data due to differences in data processing. MLP necessitates flattened input data, where input features are structured as a one-dimensional vector. Furthermore, MLP inputs often require the manual creation of features, which can be a time-consuming task. In contrast, CNN is specifically designed to handle two-dimensional input data, such as images, without the need for flattening. CNN utilizes convolutional layers to maintain spatial relationships in the input data, making them particularly effective for tasks like image recognition and computer vision.

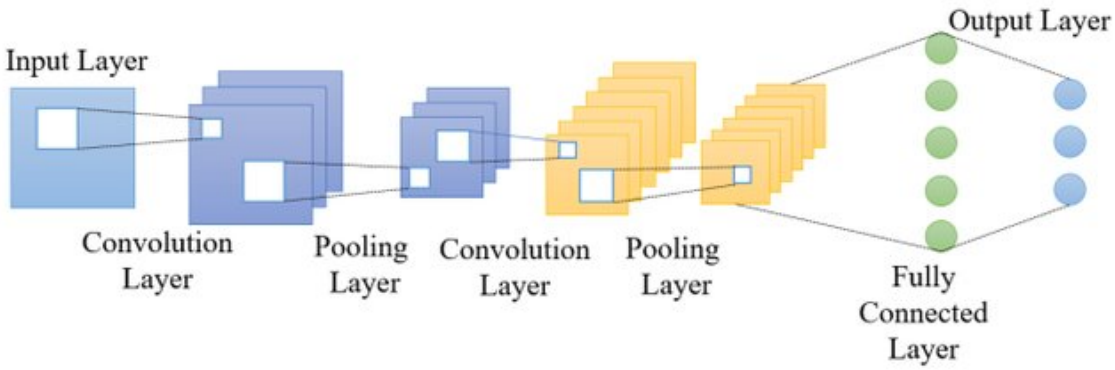


Figure 2.18. Illustration of the basic architecture of CNN: it consists of two convolutional layers, each followed by a pooling layer. Classification is performed in the fully connected layer on the end of the CNN [118].

2.2.4.4 Evaluation

Several metrics are commonly used in NN evaluation. They provide a comprehensive understanding of classifier performance and can help identify areas for potential enhancements. Although accuracy is the most commonly used measure, considering and understanding some other frequently used evaluation metrics is crucial for effective evaluation. Additional evaluation metrics usually used to illustrate the performance of the model are:

- accuracy (ACC), percentage of correct classifications, determined as

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.5}$$

- sensitivity, recall, or true positive rate (TPR)

$$TPR = \frac{TP}{TP + FN} \tag{2.6}$$

- specificity, or true negative rate (TNR)

$$TNR = \frac{TN}{TN + FP} \tag{2.7}$$

- false negative rate (FNR)

$$FNR = \frac{FN}{FN + TP} = 1 - TPR \tag{2.8}$$

- false positive rate (FPR)

$$FPR = \frac{FP}{FP+TN} = 1 - TNR \tag{2.9}$$

- F1 score

$$F1 = 2\frac{PPV * TPR}{PPV + TPR}. \tag{2.10}$$

where TP stands for the number of correctly predicted positive classes, TN is the number of correctly predicted negative classes, FP is the number of samples incorrectly predicted as positive classes, and finally, FN is the number of samples incorrectly predicted as negative classes. In some scenarios, like in the case of class imbalance, accuracy is not a good metric. Instead, in that situation, it is recommended to use the $F1$ score, which is defined as the harmonic mean of precision (PPV, where $PPV = \frac{TP}{TP+FP}$) and recall, designed to work well on imbalanced data.

Another standard technique for summarizing the classifier performance (for a balanced dataset) is to produce a receiver operating characteristic curve (ROC), created by varying the threshold values used to classify instances as positive or negative. For each threshold value, the TPR and FPR are calculated, resulting in a point on the curve. The model's performance is better if the ROC curve is closer to the top-left corner. By calculating the corresponding AUC, another evaluation metric is obtained; a larger area under the ROC curve (AUC) signifies a better classifier. In the HEP, it is common to interpret the ROC curve in terms of signal efficiency (TPR) vs. background rejection (TNR). Also, physicists are sometimes interested in other metrics, like signal efficiency at some fixed level of background rejection. When dealing with imbalanced dataset scenarios, using a precision-recall curve instead of an ROC curve is more appropriate. The former curve summarizes the trade-off between the true-positive rate and the positive predictive value for a predictive model using different probability thresholds.

Evaluation metrics are often used in conjunction to provide a more comprehensive understanding of a model's performance. For particle detector triggers, it is essential to detect as many interesting events as possible, meaning the trigger algorithm should operate at a very low FNR. On the other hand, low FPR ensures staying within bandwidth limitations without compromising the analysis's ability to achieve successful results.

In addition to the model accuracy calculated by using equation 2.2.4.4, it is common to extend a binary metric in a way that it fits to a multiclass or multilabel problems by treating the data as a collection of binary problems. By using this approach, two additional types of accuracies, "macro" and "weighted," are introduced. "Macro" accuracy is a way of calculating the mean of the binary metrics, where each class is given equal weight. It is

calculated by using the formula

$$Macro\ F1\ Score = \frac{\sum_{n=1}^{n} F1\ Score_i}{n}. \tag{2.11}$$

While macro-averaging can be useful to highlight the performance of rare classes, it assumes that all classes are equally important, which is often not the case. Therefore, macro-averaging can sometimes overemphasize the poor performance of infrequent classes. "Weighted" accuracy considers class imbalance by computing the average of binary metrics. Each class's score is weighted by its presence in the true data sample. Therefore, it is calculated by:

$$Weighted\ F1\ Score = \sum_{n=1}^{n} w_i \cdot F1\ Score_i \tag{2.12}$$

where

$$w_i = \frac{Num.\ of\ samples\ in\ class\ i}{Total\ number\ of\ samples}.$$

2.2.4.5 Classification Threshold Adjustment

In some scenarios, the default threshold (0.5) may lead to poor performance. It is the case when the predicted probabilities are not calibrated, which is a known problem for modern neural networks [119]. Adjusting the threshold could result in a more accurate outcome. Classification threshold adjustment method [120] is a method used in binary classification that involves changing the threshold at which predictions are assigned to different classes. Although the classification threshold adjustment was originally intended to be used in the context of imbalanced datasets, it can also be convenient in other scenarios. One of them is the cost-sensitive classification problem, where the cost of one type of misclassification is more important. Binary classification often requires optimizing specific metrics, such as minimizing FNR or FPR (or maximizing TNR or TPR) based on the application. As per the findings in [121], the cost of misclassifying an actual positive example into a negative one is usually more expensive than that of misclassifying an actual negative example into a positive. This observation is also relevant when it comes to the EM shower selection problem that is considered in the thesis. In other words, it is more crucial to avoid misclassifying an EM shower as background than it is to avoid misclassifying a PU/QCD as an EM shower. The first misclassification may lead to the loss of valuable information while the second case results in a useless piece of data in storage.

Therefore, when interpreting the predictions of a model, depending on the target optimization, there is a need to change the default decision threshold of 0.5. If TPR and TNR have the same importance, one way of calculating the cut-off is using Youden's J statistic ([122]), defined as follows:

$$J = Sensitivity + Specificity - 1, \tag{2.13}$$

or simplified:

$$J = TPR - FPR. \tag{2.14}$$

Youden's index is often used in combination with ROC curve analysis. The index is calculated for all points of an ROC curve, and the optimal threshold with the largest J value is chosen. If both metrics, FNR and FPR, are not equally important, using the trial and error method helps find the threshold to lower the targeted metric (which is FNR in the considered classification problem).

2.2.5 Neural Network Applications in HEP

NNs are crucial tools in HEP, used for data analysis, pattern recognition, and optimizing complex experimental processes. NNs can distinguish between different types of particles produced in high-energy collisions, such as electrons, muons, and hadrons, providing high-purity samples of a given type. They classify particles based on their energy deposits and trajectory information in particle detectors. In the traditional approach, particle selection is achieved by applying "cuts" to the handcrafted features obtained from the detector response to reject the particles not meeting the specified criterion. When the difference between various types of particles is significant, this approach is justifiable. But when the feature distributions among different particle species overlap, it can lead to unwanted outcomes. By considering multiple features and correlations between them, NNs significantly outperform simple cutoffs in discriminating between particle types [92]. The decision boundaries for particle identification are based on data and can adjust to changing conditions, resulting in improved performance. Another NN advantage is the ability to handle multiclass classification problems, allowing the identification of multiple particle types simultaneously, which is valuable in experiments with a diverse range of particles. Probabilistic outputs for particle types provided by NNs enable precise decision-making and accurate uncertainty estimation, particularly in non-binary identifications.

Today, various types of NNs are successfully used in numerous HEP applications. Below is an overview of the main areas where NNs are used, with a focus on the application of NNs in triggering:

- **Particle Identification and Classification, Event Reconstruction**

  Already in 1995, the NN approach was suggested for electron identification in the ZEUS uranium calorimeter [123]. In the [16], authors report results of image-based event classifiers Residual Net-type, using simulated, low-level detector data as inputs. In the ATLAS experiment, authors in [124] describe the usage of CNN for electron

identification. NN's input are the images of the deposited energy in the calorimeter cells. NN also uses additional input features, the high-level variables that are used by the likelihood (LLH), and other NN algorithms developed in ATLAS. Additional features include information on up to five inner detector tracks that are matched to an electron candidate during reconstruction. As a result, NN gives the probability that a reconstructed electron belongs to six classes of signals and backgrounds.

The process of interpreting electronic signals in the detector to determine the original particle characteristics and particle type is known as event reconstruction, which is closely linked to particle identification in HEP experiments. Utilizing advanced algorithms, which frequently incorporate ML techniques, can optimize and enhance the accuracy of the reconstruction process.

- **Event Simulation**

  Event simulations play a vital role in predicting the outcomes of particle collisions at the LHC. Additionally, due to planned high luminosity in upcoming colliders, the standard Monte Carlo simulation approach is becoming computationally challenging. The use of NNs can significantly enhance the efficiency and accuracy of these simulations. Rather than relying solely on traditional models, NNs can learn patterns and correlations from existing data to generate more realistic simulated events. A number of authors have proposed the use of NNs to simulate events [21, 22, 125, 126].

- **Anomaly Detection**

  Anomaly detection involves identifying data points, entities, or events that fall outside the normal range, deviating from what is standard or expected. In HEP experiments, it is a powerful tool that serves two main purposes: discovering new physics and ensuring the accuracy and reliability of experimental data. Anomalies can indicate the presence of new particles or phenomena not anticipated by existing theories. Also, using anomaly detection techniques, researchers can distinguish rare events or processes that might happen at a very low frequency from the background noise. Anomalies can reveal discrepancies between observed data and theoretical models or detector behavior, leading to improved modeling and calibration, ultimately enhancing simulation and analysis accuracy. Therefore, identifying anomalies is a key step in discovering new physics beyond the Standard Model. In HEP, various NN approaches [127–131] have been extensively studied for the problem of anomaly detection.

- **Data Compression and Reduction**

  Trigger algorithms select potentially interesting collision events and store them for further analysis. Using that approach, the full records of a small subset of the data are saved. Newer strategies suggest storing partial information about a larger fraction

of events that would eventually lead to the discovery of the existence of a new kind of particle or interaction. NNs can reduce data dimensionality while preserving relevant information, which is helpful in managing the vast amounts of data produced in high-energy physics experiments. For example, in [132], authors suggest automatical compression of the full experimental information by using the Variational Autoencoder. An ASIC implementation of a low-power, low-latency data compression algorithm based on a convolutional neural network is presented in [133]. More NN-based approaches for data compression can be find in [134–137] With the development of libraries and tools that facilitate the implementation of NN models in FPGA, their usage has become possible in restricted environments where low latency is required. In [138], authors suggest the implementation of high-performance, sub-microsecond NNs on FPGAs for ATLAS trigger application. Authors in [30] present a case study for applying NNs in jet classification. The proposed NN is implemented in FPGA using *hls4ml* and, with a latency of 75–150 ns, fits well into the CMS and ATLAS trigger constraints. Real-time jet identification using two types of recurrent neural networks, long short-term memory (LSTM) and gated recurrent unit (GRU), using the same way of FPGA implementation is given in [70]. CNN-based fast muon tracking used for muon trigger is presented in [68], and FPGA-based NN regression model for applications in the future hardware muon trigger system of the ATLAS is given in [69]. CNN is also proposed for track seed filtering in CMS high-level trigger [18]. An example of the recurrent neural network allowing the reconstruction of the energy deposited in the calorimeter and implemented in FPGA is shown in [139] and [140], while authors in [140] also present the CNN model for energy reconstruction. FPGA implementation of another NN architecture, GNN, used for charged particle tracking, is shown in [141] and [34]. A proposal for CNN usage for a real-time data acquisition and trigger system in another HEP project, the Deep Underground Neutrino Experiment (DUNE) particle detector, is given by authors in [35]. An upgrade to the L1T system in the Super Cryogenic Dark Matter Search (SuperCDMS) SNOLAB experiment is proposed in [142]: an LSTM neural network implemented on the trigger FPGA is expected to lower the trigger threshold by 22%. Authors in [143] propose a fast neural network package named ANN4FLES (containing MLP and CNN) for real-time data analysis in heavy-ion physics experiment Compressed Baryonic Matter (CBM).

It is crucial to understand that the ML models highlighted in the review are designed to make predictions for distinct applications utilizing distinct datasets and feature types. As a result, comparing the accuracy or efficiency of one model to another is not a simple task. Each model is fine-tuned to operate on a specific set of features, and therefore, evaluating their performance must be based on identical criteria. Overlooking to do so may result in misleading conclusions that do not accurately reflect the models' capabilities.

NNs have shown their immense value in the field of HEP research. These networks can efficiently process large amounts of data, identify intricate patterns, and refine analytical techniques. As a result, they have contributed significantly to a better understanding of the fundamental particles and forces. With the increasing amount of data generated by experiments, NNs have become an essential tool in their analysis, leading to the discovery of new and exciting physics beyond the limitations of the Standard Model. Also, NNs are extremely powerful when applied to low-level inputs as they get access to full information about every event. At the same time, there are other benefits from skipping the need for human preprocessing: data processing is faster, and potential biases from feature engineering are avoided.

Dealing with PU presents a significant technical challenge in the HEP experiments. This challenge is even more pronounced when luminosity increases in the incoming HL-LHC era when a PU of 140 or 200 is expected (on average, 140 to 200 additional proton-proton collisions occur simultaneously with the primary collision of interest). Scientists and engineers use advanced algorithms to extract meaningful information from collisions and reduce the effects of PU. For example, for the PF algorithms used to reconstruct and identify each particle in an event by combining the information from all the subdetectors, multiple techniques are developed to mitigate the impact of PU collisions effectively [144]. However, the presence of PU significantly complicates and hinders the process of particle identification when relying on ML techniques based on raw data.

2.3 Dataset

To conduct experiments within the thesis research, it is necessary to obtain images that provide a precise representation of particles' passage through the HGCAL. Updates to the CMS detector are ongoing, and the final HGCAL architecture has yet to be determined. As a result, real test data is not available. For the purposes of this thesis, simulated data is created using a standard tool designed for the generation of particle events needed for physics analysis.

2.3.1 Event Generation Process

Simulated data plays a crucial role in developing and enhancing detectors for various applications. In the initial stages of detector development, when real data is not available, simulated data is used to design and optimize the detector's geometry, material composition, and readout electronics. Simulated data offers several benefits over real data as it can be generated under controlled conditions, which may be difficult or impossible to achieve with real data. On the other hand, simulated data often relies on simplified models that may not fully capture the complexities of real-world data, leading to models that perform well on simulated data but poorly on real data.

To replicate the detector's behavior as accurately as possible, the simulation process involves the following steps [145]:

- **Collision generator**: This step simulates the creation of particles during a collision.
- **Geometry**: This step involves providing an exact description of all the detector's active (as sensors) and inactive (cooling, cables) elements.
- **Transport software**: This step simulates the interaction of particles with detector material, taking into account their behavior and characteristics.

Scientists involved in the CMS project for simulation and data analysis use CMS Software Components (CMSSW) [146], a software project that is open-source and represents a significant collaborative effort within the CMS collaboration and the broader HEP community. The software is designed to be flexible and modular, which makes it easy for physicists to customize and extend it for specific analyses. This modular design also allows the software to be readily adapted and updated as new technologies and techniques become available. CMSSW plays a crucial role in simulating CMS detector event data, which is essential for scientists to perform various types of analyses. CMSSW generates Monte Carlo (MC) simulation events, which are events that are simulated using statistical methods rather than being observed in nature.

The primary physics processes that are produced by programs such as Pythia [147] are used to generate these events. The software uses Geant4 [148], a tool that simulates particle interactions with the detector material to create a detailed simulation of the CMS detector. Geant4 is able to simulate particle interactions with great precision and can, therefore, provide a realistic simulation of the detector. The research dataset was created using data obtained from CMSSW version 12.1.0. Although newer CMSSW versions are now available, the one being utilized is the version accessible when the thesis experiment started.

2.3.2 Event Simulation

The chosen data to represent a signal is based on an EM shower, a cascade of particles created by an electromagnetic interaction. This shower is commonly used in experiments measuring high-energy particles, where the particles interact with the detector material and produce secondary particles that create the shower. The data collected from this shower provides valuable insights into the properties of the particles and their interactions.

The EM showers used in the experiment are obtained by simulating electrons with the transverse momentum (pt) between 2 and 200 GeV, where pt is a component of momentum perpendicular to the beam line. A minimum bias simulation is used to create the background samples. Another type of background, quantum chromodynamics (QCD) jets, is also considered. QCD jets background is generated by simulating the QCD sample with pt from 50

to 80 GeV. For the multiclass classification, another type of background will be added, pions, subatomic particles belonging to the family of mesons. In the experiment, just charged pions will be considered. Pions are generated by simulating a Single Pion sample with the *pt* between 0 and 200 GeV. All four simulated sets are obtained using a PU value of 200, adding a comparable level of additional proton-proton interactions occurring along with the primary interaction of interest.

EM showers are important processes to consider as they are produced by electrons and photons. Besides, interesting heavy particles like the Higgs boson can decay and produce electrons or photons. It is crucial to differentiate these particles from the PU, QCD jets, or pions that are more often generated.

2.3.3 Event Selection

Identifying nearby clusters of the generated electrons and positrons in an electromagnetic shower is crucial. Specifically, three-dimensional (3D) clusters within a $\Delta R < 0.2$ of the generated electrons and positrons are considered potential candidates for further analysis.

If multiple 3D clusters are located in close proximity to the generated particle, a selection process is initiated. The requirement for the cluster's *pt* is that it has to be higher than 10 GeV. Specifically, only those clusters whose *pt* falls between 80% and 120% of the *pt* of the generated particle are considered. This subset selects the 3D cluster with the highest *pt* value. This criterion ensures that the clusters are likely to be associated with the EM shower and are not due to other sources of background noise. To select the PU, 3D clusters with *pt* greater than 5 GeV are identified. Only clusters that are not in close proximity to generated photons, electrons, and positrons are chosen. A minimum distance of 0.2 in ΔR is maintained between the chosen clusters and the aforementioned particles. To identify QCD jets, 3D clusters close $(\Delta R < 0.4)$ to genjets with $pt > 30$ GeV are selected. Genjets are jets derived from generator-level Monte Carlo particles, and they contain information about fractions from different types of generated particles (charged hadrons). The selection of charged pions is similar to electron selection: 3D clusters within a $\Delta R < 0.2$ of the generated charged pions are considered. Further, if there are more 3D clusters close to the generated particle, the clusters whose *pt* is between 80% and 120% of the belonging generated particle's *pt* are selected. Finally, the cluster with the highest *pt* value is chosen.

Using the above description, 106 000 samples are generated: 31 000 for each of EM, PU, and QCD jets samples and 13 000 pion samples. Figure 2.19 shows how the samples are distributed across different classes used in the classification process. Each bar in the graph corresponds to a specific class, and its height indicates how many samples belong to that class. Although it was planned to have an equal number of all simulated particles, a larger set of simulated pions that match the detector structure used to generate the remaining samples is not available in the repository. In this study, only low-level features are used:

pt, position *(x,y,z)*, layer number, pseudorapidity (η), azimuthal angle (ϕ), and *particle ID* given by the generator.

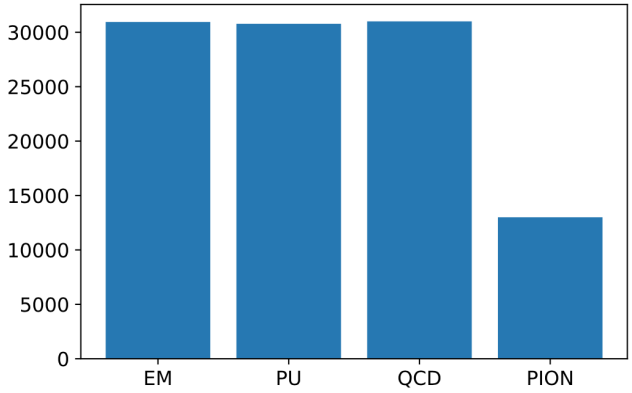


Figure 2.19. The distribution of sample numbers according to class.

2.3.4 Image Formation

Cogan's [14] study demonstrated a novel and revolutionary approach to analyzing calorimeter cells by treating the energy deposited in them as pixels of an image. Inspired by computer vision methods, this approach considers the particles detected by the physics detector as images captured on a detector playing the role of a digital camera. In more detail, the authors define jet-images using calorimeter towers as the elements of the image needed for jet tagging and classification. All previous approaches required additional preprocessing to extract physical features from the data. In contrast, this new image-based proposal does not require such preprocessing. The calorimeter cells are already treated as pixels of an image, allowing the application of advanced image-processing techniques, which can lead to more accurate and efficient analysis of the data. Therefore, in a very short time, numerous studies have been carried out using the same methodology [149–151]. In [152], authors give an overview of computer vision-based approaches to the analysis of LHC events. Treating the data collected by the detector as images has become the standard in calorimeter data processing [153, 154] and the same approach will be used in this thesis.

Particles or decay products passing through the calorimeter layers deposit energy in sensor cells grouped into TC. In the CMSSW version used in thesis (v12.1.0), improved clustering algorithms are used, enabling direct 3D clustering. The generated 3D cluster provides complete 3D information about the shower, as seen in Figure 2.20. When creating an image, if the size of the targeted image is smaller than the range of an individual 2D cluster, then not all TCs will participate in generating the image. This means that some TCs will be left out, resulting in an incomplete image. To demonstrate this, Figure 2.20 shows all the TCs in a cluster, with a black border around the cells that are involved in generating the image and a red border around the cells that are not. To satisfy L1T constraints, it is important to use

the smallest possible cluster while ensuring that the classification is done efficiently. By reducing the size of the cluster, the L1T system can process the data faster, which in turn helps to reduce the overall processing time and improve the accuracy of the results. Therefore, the goal is to find a balance between the size of the cluster and its classification efficiency.

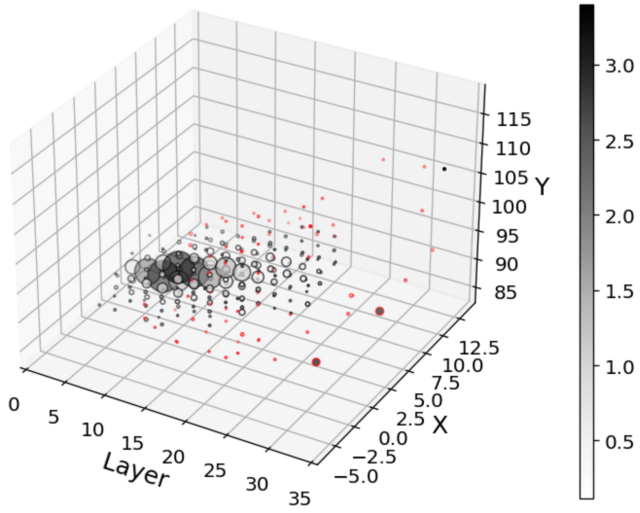
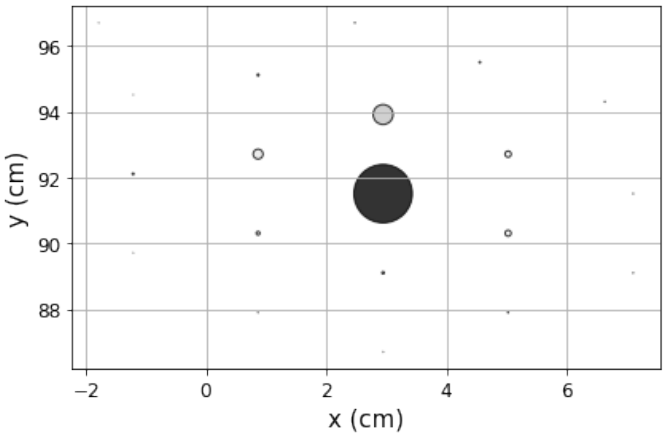


Figure 2.20. Example of 3D EM cluster where dot size and color are proportional to deposited pt. TCs participating in 5x5x36 image generation are bordered black, and those not involved in image generation have red borders.

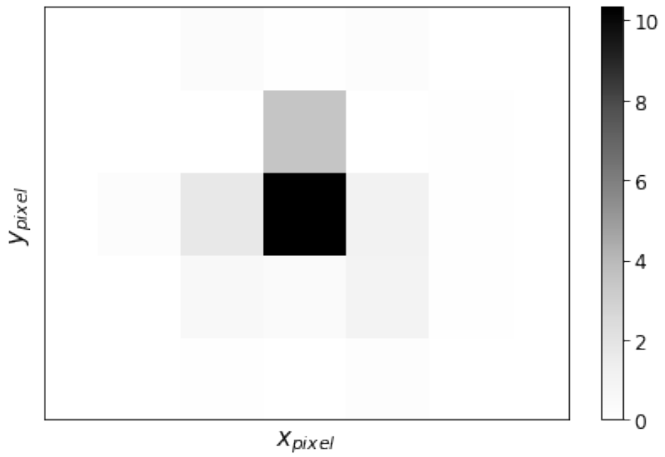
After obtaining 3D cluster data through CMSSW, virtual images are generated through a series of precise steps. First, a line is drawn between the center of mass of the 3D cluster and the center of the detector. This line represents the deposition axis. Next, the intersection point of the axis with each individual layer is identified as the center of the ROI on that layer. Figure 2.21a is a visual representation of the projection of deposited particle energies around the intersection point in one detector layer. The next step involves calculating the range along the coordinate axes based on the center and also taking into account the target size of the ROI. To summarize all the energies, a 2D histogram of 5×5 bins is applied (the target size in this thesis), with a bin size of $2 \times 2cm^2$ that is adjusted according to the size of the trigger cell. The step is shown in Figure 2.21b.

The study [155] conducted by Prvan found that increasing window size does not significantly affect model accuracy with a similar dataset.

The image generation procedure that has been described results in the creation of 3D images with a 5×5 shape. The first two parameters, i.e., 5×5, represent the window size. The third parameter, which is 36, indicates the depth of the image. In other words, the output image is composed of 36 layers, each of which displays an HGCAL layer. Since every second layer is used for triggering in the electromagnetic part of the calorimeter, it is described with the first 14 images, while the hadronic part consists of the remaining 22 images, one for each layer. Depending on the specific requirements, it is possible to use the



(a) Example of the first step in image generation: the projection of deposited particle energies (on the layer).



(b) The histogram applied on layer image.

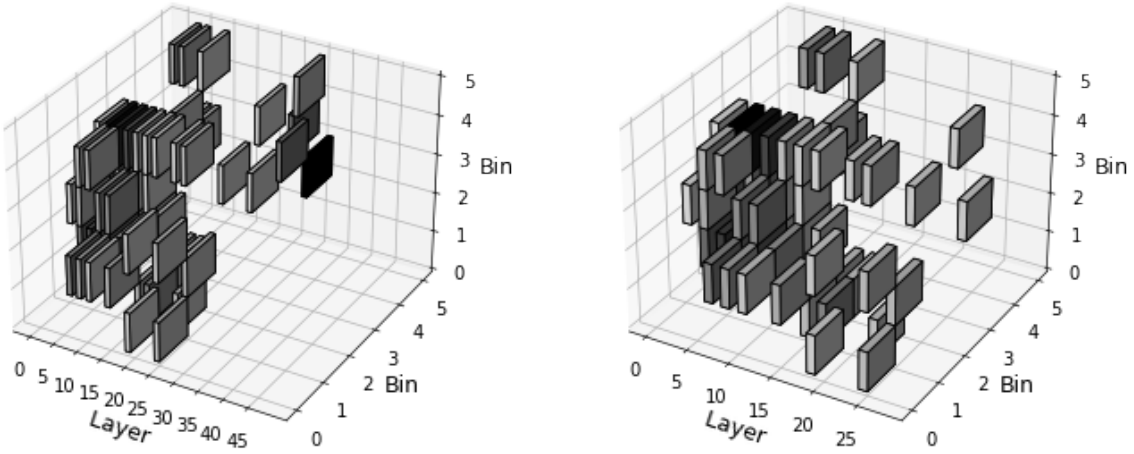
Figure 2.21. Visualizing the steps involved in creating one layer (13) image.

image of the entire endcap or just the individual CE-E or CE-H components. This flexibility allows for a more customized and efficient approach to observing the endcap, as it can be tailored to suit the specific needs of the user.

Upon analyzing the longitudinal profile following the image generation process, it was discovered that layer 29 contains a peak that does not align with the actual physical properties of the particles being observed. Instead, this peak is a result of the calibration process and does not represent an actual energy deposit in that particular layer. As a result, layer 29 will be excluded from the image.

In the thesis, the models are tested on full HGCAL image (Figure 2.22), but for some of the tested approaches, like reduction of the longitudinal profile, the HGCAL image is split into CE-E and CE-H image parts. CE-E image has a shape $5 \times 5 \times 14$ (Figure 2.22) and shows only the electromagnetic part of the calorimeter, while CE-H image with a shape $5 \times 5 \times 22$ (after layer 29 exclusion $5 \times 5 \times 21$) represents a hadronic part.

Figures 2.23a, 2.23b, 2.23c, and 2.23d represent the longitudinal profile for EM, PU,



(a) Complete HGCAL image *(b) CE-E part image*

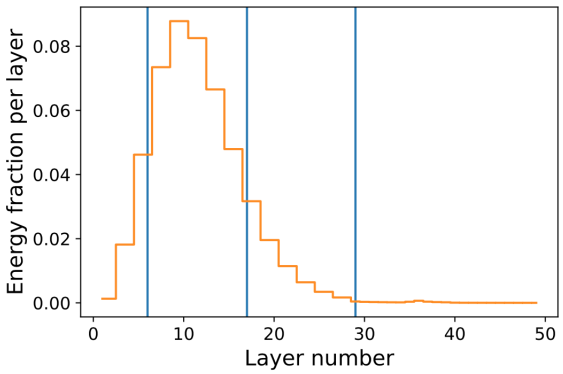
Figure 2.22. Example of generated 3D HGCAL image.

QCD, and pion showers. For easier and more detailed analysis, layers 6, 17, and 29 are specifically marked. The EM showers deposit most of its energy between channels 6 and 17, while channel 29 marks the beginning of the hadronic part. It is clear that compared with EM showers (on average), PU and QCD start showering earlier, with a peak reached before layer 10 and a higher energy deposit in the hadronic part. The average pion shower profile is significantly different from other types of showers. The pion shower shows a significantly higher proportion of the hadronic part.

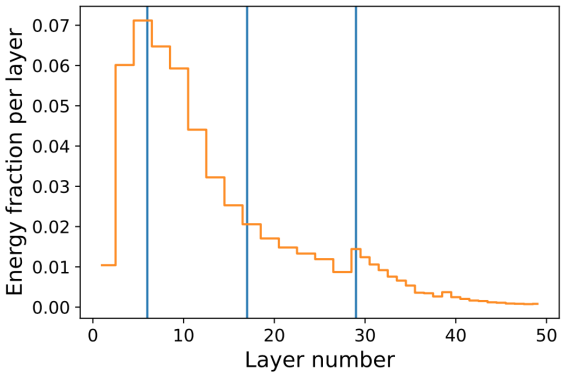
2.4 Data Reduction Methods

The effectiveness of ML algorithms largely depends on the quality of data they operate on. If data is incomplete, irrelevant, or contains extraneous information, ML algorithms may produce inaccurate or useless results. Therefore, it is essential to ensure that the data used for the ML model is of high quality and relevant to the problem at hand. Accordingly, data preprocessing is a crucial step in the ML workflow, essential for achieving accurate and reliable results. Proper preprocessing can significantly impact the performance and accuracy of ML models. A comprehensive review of data preprocessing techniques can be found in [156] and [157]. In time-sensitive applications such as L1T, it is imperative to process data quickly and efficiently, as any delay in processing can potentially result in the loss of important data. With the rise of real-time applications, new approaches to data preprocessing have emerged, and some of these methods were proposed by the authors in [158].

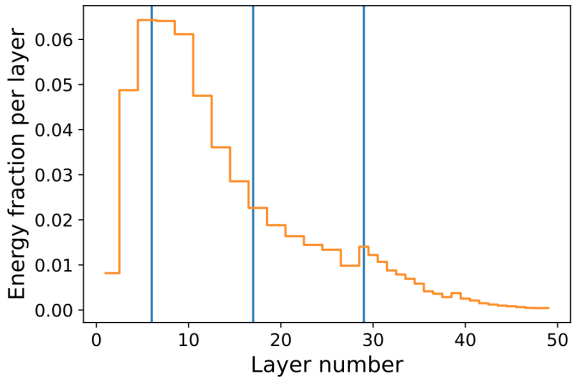
There is no universally agreed upon approach among data science researchers on the number of steps involved in this process. For example, handling outliers, according to some scientists, is a part of data cleaning, while others consider it as a separate step. Regardless



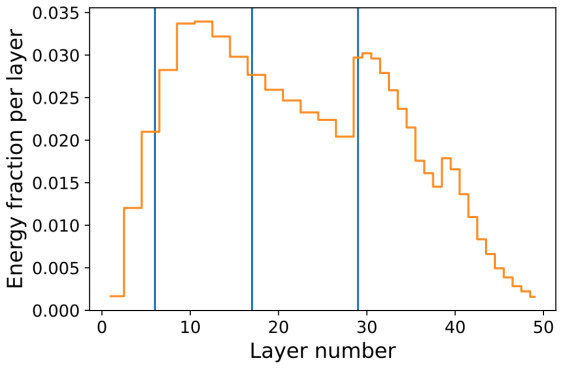
(a) EM



(b) PU



(c) QCD



(d) pions

Figure 2.23. Histogram representing average longitudinal profiles for particles/classes used in experiment.

of the approach, some of the most important steps are data cleaning, reduction, scaling, transformation, and partitioning, although the choice of applied techniques depends on the nature of the data, and the requirements of the model. Below is a detailed description of the steps used in this thesis.

The first stage in data preprocessing is data cleaning, which involves identifying and correcting inconsistencies in the dataset, like errors and missing values. In data cleaning, it is crucial to assess the entire scope of the problem and the method used rather than just the values present. For example, one of the standard techniques in data cleaning is to filter out unwanted outliers, the data points that differ significantly from other values in a dataset. Typically, values that lie far outside the expected range are considered outliers. They can negatively affect model performance, and therefore, they are removed from analysis. This technique is also known as trimming. In the experiment described in the thesis, pixel intensities of generated HGCAL images can have large variations; quite often, they span a few orders of magnitude. Particularly, the PU dataset has a considerable number of outliers that require attention to ensure accurate analysis. However, in our specific case study, these outliers are the highest energy points, providing crucial information about the observed cluster. The presence of outliers can indicate the discovery of new phenomena or anomalies, which can provide valuable insights into the fundamental nature of particle interactions. Therefore, removing them would result in a significant loss of important insights and valuable knowledge. As such, it is necessary to use alternative methods to handle these outliers that do not involve their outright removal.

One way to handle extreme values is by using capping. This method sets a maximum or minimum limit and assigns values to the data points accordingly. Another method is robust scaler normalization [159], which actually includes the next step of data preprocessing but, at the same time, takes care of outliers. It reduces the impact of outliers by scaling the data based on the median and interquartile range instead of the mean and standard deviation. Both of these techniques can effectively manage outliers and produce more accurate results. Tests are done with both techniques, and they show similar results. Capping requires minimal calculations, making it highly efficient in terms of CPU usage, and generally does not increase memory consumption. Therefore, due to its speed and resource efficiency, capping will be applied in the thesis. The next stage of data analysis involves data transformation, which entails converting raw data into a more appropriate format for analysis and modeling. This includes feature scaling (standardization or normalization), a method used to bring numerical features to a similar scale. Feature scaling is applied to enhance the numerical stability of models and expedite the training process. Another method in this step is data binning or discretization, which involves converting numerical features to discrete ones using bins. In Subsection 2.4.1.2, it will be demonstrated how a data quantization method will be used as a reduction technique, which will produce an effect similar to discretization.

2.4.1 Data Refinement

The new detector structure provides detailed collision images. However, to meet latency constraints set by L1T algorithms, the possibility of reducing the number of features through dimensionality reduction is explored. When dealing with image data, the number of pixels can be considered as the number of features; each pixel represents one feature. Even though the window size in generated HGCAL images is quite small (5×5), a high number of image channels (that correspond to HGCAL layers) are the reason for a high number of pixels - features. If the entire HGCAL image is considered, the number of pixels would be 900 for $5 \times 5 \times 36$ or 875 for $5 \times 5 \times 35$ version. Thus, the data refinement process, consisting of data selection and quantization, aims to reduce the quantity of data (i.e. data complexity) that the network needs to deal with to achieve lower inference time while retaining a similar performance level.

Two main approaches in dealing with high-dimensional data, feature extraction (FE) and feature selection (FS) both contribute to the feature transformation process by modifying the original set of features. FE is a process in which new features are created by combining the original ones. In comparison, FS methods reduce the dimensionality of the dataset by keeping the most important features and discarding the others. The study presented in [160] gives a comprehensive review of various FE methods that reduce processing time while providing higher recognition accuracy. While FE is the preferred approach in image processing, both approaches are employed due to time and resource limitations, as they reduce data dimensionality without significant performance degradation.

Data refinement is conducted in two steps: first, data selection, followed by data quantization.

2.4.1.1 Selection

The experiment utilizes ad-hoc techniques suitable for our specific physically motivated classification problem. In order to optimize computational time, two techniques are applied. The first one involves filtering out irrelevant pixel values (FS approach). The second technique engages in combining/summing particular image channels (FE approach), which reduces the overall complexity of the algorithm. Each of the presented techniques is chosen based on the combination of physics-based and logical-based reasoning. It is worth emphasizing that feature selection is applied to full-precision data, using one of the following rules:

1. **Keep energies above the threshold.**

   Analyzing HGCAL pixel values (trigger cells *pt*), shown in Figure 2.24, it is visible that most of them have values less than 2 GeV.

   The smallest energy values are considered to have minimal impact on determining if a cluster represents a signal or background. Therefore, those values will be discarded by

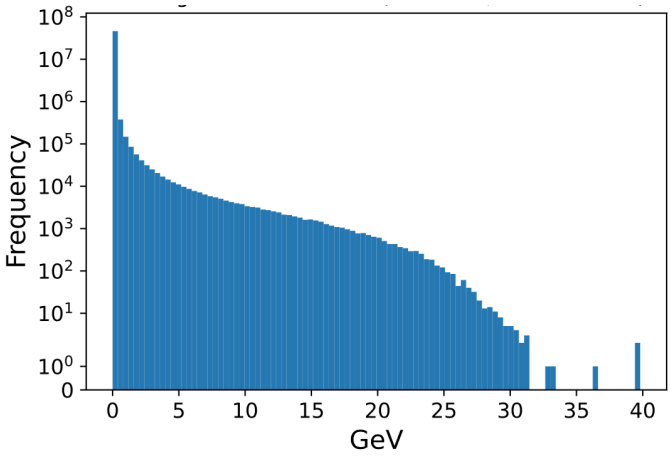


Figure 2.24. Histogram representing distribution of all dataset values in case of EM vs MIX (PU and QCD) dataset.

applying the next criteria: a fixed threshold value of 0.1, 0.5 (default), 1, and 2 GeV is applied, all *pt* values greater than the threshold are kept, and others are set to 0. By setting the proposed thresholds to be small, it is considered that the TC energies are significantly smaller than the cluster *pt*.

2. **Keep maximum layer energy (single sample per layer).**

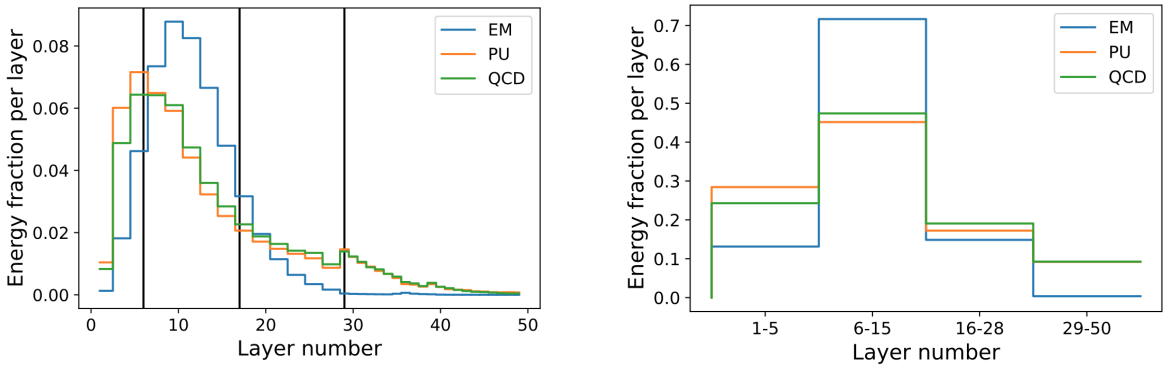
   With this approach, only the maximum energy value in each layer is kept, while other values are set to 0. The idea behind this approach is to assume that the highest *pt* deposits have the most significant role in shower classification.

3. **Capping.**

   Data capping is a statistical method in which the maximum value of a feature is set to a specific value. This technique is often used to address outliers that can otherwise distort the overall analysis. Because of its speed and resource efficiency, it is suitable for L1T algorithms. During a preliminary test, the models were implemented in FPGA using *hls4ml*, and a significant drop in accuracy was observed when dealing with non-capped data. The drop was around 10% compared to QKeras model accuracy.

4. **Reduction in longitudinal profile.**

   Andrews in [161] presented the idea of reduction of longitudinal profile based on the next approach: detector images will consist of altogether just three subdetector channels, one each for CE-E, CE-H, and one for the reconstructed tracks. Inspired by that approach, the following data refinement method is based on summing pt/pixels from particular detector layers. By summing the values of individual layers, it is possible to considerably reduce the image (channels) size, drastically decreasing the NN model's size and improving its overall performance. Two different solutions are examined:



(a) Three CE-E + CE-H layers split *(b) Summed pt in grouped layers*

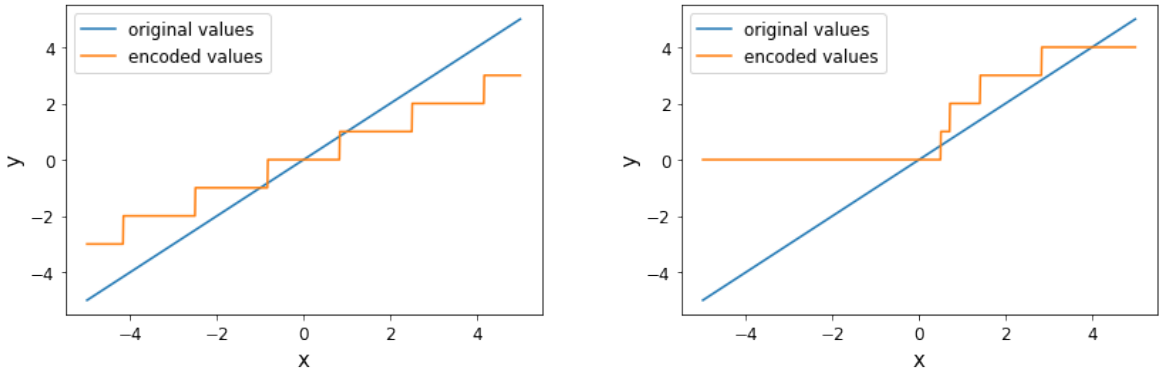
Figure 2.25. Distribution of pt across HGCAL layers for EM, PU and QCD showers used in 3Ef+Hf approach.

- Considering the fact that EM shower tends to deposit most of its energy in the CE-E section of the HGCAL, it is assumed that the information about CE-E deposit is crucial. Therefore, CE-H layers are projected in one plane, reducing the number of CE-H layers from 22 to 1. This approach is named E+Hf, and the same logic with the CE-H part is also applied in the next solution.
- It is worth noting that EM showers reach their peak between layers 10 and 15, which is later than PU and QCD showers. Based on this, the CE-E is divided into three parts. The first part is composed of layers 1-5, where the EM shower begins. The second part is composed of layers 6-15, where the majority of the transverse momentum is deposited, and the third part is composed of layers 16-28, as shown in Figure 2.25a. Each part is summed (Figure 2.25b), resulting in three 5×5 CE-E images, and the CE-H part is reduced, creating a single 5×5 CE-H image. This approach is referred to as 3Ef+Hf.
- This is the most aggressive reduction of the longitudinal profile. Here, each of the CE-E and CE-H sections is projected in one plane, resulting in two 5×5 images representing the complete HGCAL.

2.4.1.2 Quantization

Data quantization allows further reduction in resource consumption. Representing continuous values with discrete values reduces memory requirements, simplifies processing, and enhances transmission efficiency. Fixed-point arithmetic is commonly used in FPGA-based algorithms because it tends to run faster and use fewer resources compared to floating-point arithmetic. Thus FPGAs, used in L1T, can process values represented by a low number of bits much faster than those represented by multiple bits.

Two types of data quantization methods are examined in the work: linear and non-linear. The quantization levels are uniformly distributed across the signal range in linear quantiza-



(a) In the case of a uniform quantizer *(b) In the case of non-uniform quantizer*

Figure 2.26. Quantizer function $y = Q(x)$

tion. In non-linear quantization, the relationship between the quantization levels is logarithmic, which is more suitable when the range of data is large, and higher accuracy is required in the lower data range. Thus, data will be quantized in three different ways using custom quantizers:

1. **Selection bit(s).** With this approach, all non-zero data values will have a value of 1. By using just one bit for presentation for each value, memory usage can be significantly reduced. Also, the overall performance of the system will be optimized.

2. **Uniform quantizer.**

   Min-max normalization is applied on capped data, multiplied by $2^n - 1$, where n is a chosen number of bit width. The result is rounded to the nearest integer. The proposed quantizer function is visualized in Figure 2.26a.

$$Q(x) = round(\frac{x - min}{max - min} \cdot (2^n - 1)) \tag{2.15}$$

3. **Non-uniform quantizer.** The quantizer is defined as

$$Q(x) = \begin{cases} round(log_2(x)), & \text{if } x > 0 \\ -2, & \text{otherwise} \end{cases} \tag{2.16}$$

   The next step is to rescale data (using a min-max scaler), so it does not contain negative values. The non-uniform quantizer function is visualized in Figure 2.26b.

An overview of data refinement methods described in this section is given in Table 2.2.

Table 2.2. Data refinement methods.

Refinement Methods	
Selection	**Quantization**
Keep energies above the threshold	Selection bit
Keep layer maximum	Uniform quantizer
Capping	Non-uniform quantizer
Reduction of longitudinal profile	

2.5 Alternative Image Generation Process

This chapter introduces an alternative approach to generating HGCAL images used to capture the decay of particles through HGCAL layers. The quality of HGCAL images is critical for the accurate functioning of ML algorithms planned to be used for the selection of physically interesting events in L1T. Therefore, there is always a need for a new approach that could eventually improve the accuracy of the trigger algorithm by providing more accurate image data. Exhaustive tests of NN models with various reduction methods will be performed using the database generated by the traditional approach, while basic NN models will also be tested on databases generated by alternative approaches.

2.5.1 Consecutive Layers Based Approach

The main difference between the suggested approach and the traditional algorithm, described in Section 2.3.4, is a selection of the deposition axis, the direction along which *pt* is deposited by particles during interactions with a detector material. Both approaches create virtual images corresponding to each of the HGCAL layers through a series of precise steps. In the traditional approach, the deposition axis is a line connecting the center of mass of the observed 3D cluster and the center of the detector. In the subsequent step, which will remain unchanged, the point where the deposition axis intersects with each individual layer will be identified as the center of the ROI on that particular layer. Instead of the line passing through the cluster barycenter and the detector center, the emphasis would be on observing the longest sequence of layers where the observed cluster has energy deposits. This statement particularly applies to PU images, where the decay may occur in a direction other than the beamline. The visualization of the deposition axis for two EM samples is shown in Figure 2.27. It can be observed that in certain situations, the axes are nearly identical, whereas in other scenarios, they are considerably different. Hence, it is likely that the resulting HGCAL images will be quite similar in some situations, while in others, they will vary significantly. The implications of these findings are clearly visible in a generated HGCAL image, as can be seen in Figure 2.28.

The presented approach suffers from the serious limitation that there are shower shapes

for which this approach is not favorable, and the generated images do not show enough data for a valid description of the shower.

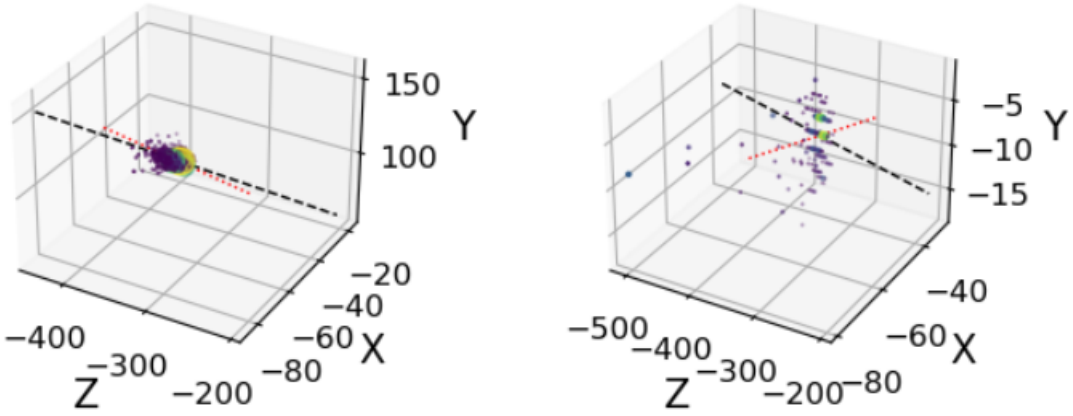
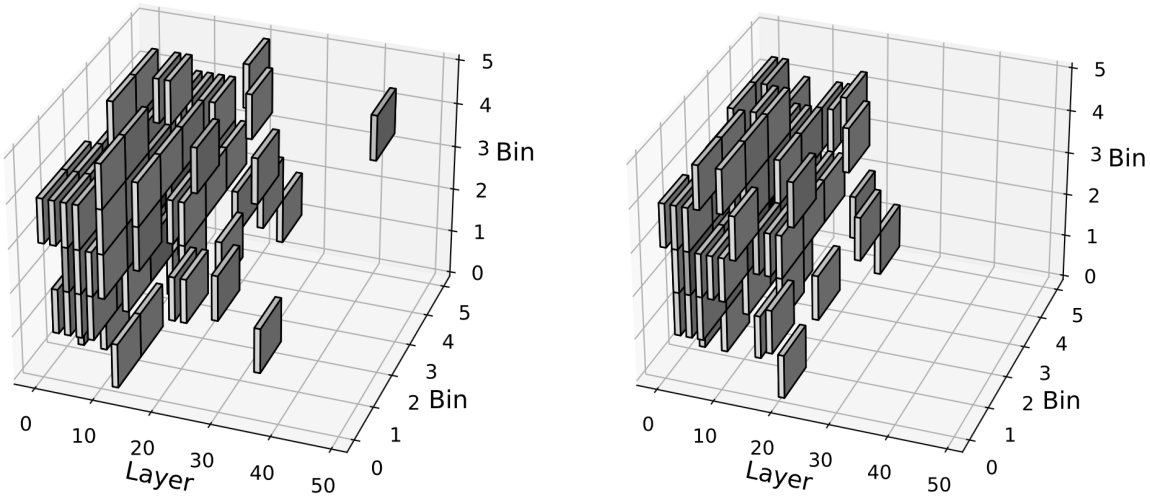


Figure 2.27. Visualization of traditional deposition axis (red) and the one based on the longest consecutive layers (black). In the left EM sample (clusterID=3139240274), the axes almost coincide, while in the right one (clusterID=3122017111), they are significantly different.



(a) Generated using traditional approach. *(b) Based on an alternative approach.*

Figure 2.28. HGCAL image (for EM clusterID=3122017111).

2.5.2 Principal Component Analysis Based Approach

Considering the importance of the choice of deposition axes in image formation, the Principal Component Analysis (PCA) is used in this approach.

PCA is a widely used method for reducing data dimensionality, first introduced in 1901 by Pearson [162]. He identified the direction (known as the first principal component) that

best approximates the set of points representing the data in terms of least squared error. Independently, Hotelling also developed the method and presented it in 1933 [163]. PCA functions by transforming the original variables into a set of new, uncorrelated variables, known as principal components, that are orthogonal to each other. The first principal component is a linear combination of the original predictor variables that captures the maximum variance of the dataset. It indicates the direction of the highest variability in the data. The first principal component represents the line that best fits the data by minimizing the sum of the squared distance between a data point and the line. Therefore, the first principal component will represent the deposition axes in this image generation approach.

The visualization of the deposition axis for two EM samples is shown in Figure 2.29. It can be observed that the axes are nearly identical to those used in the standard approach (2.3.4), which was not the case in the previous approach, as can be seen in Figure 2.27. As a result, generated HGCAL images will be very similar.

In the HGCAL project, PCA has been used for different purposes. In the TICL framework, which is a crucial component of the CMS experiment, designed to enhance the reconstruction of particle tracks and clusters, PCA is used to derive the shower propagation direction through the three principal axes [164]. In the Mask Region-Based CNN used for the HGCAL study described in [165], authors use PCA to determine the mask of the object.

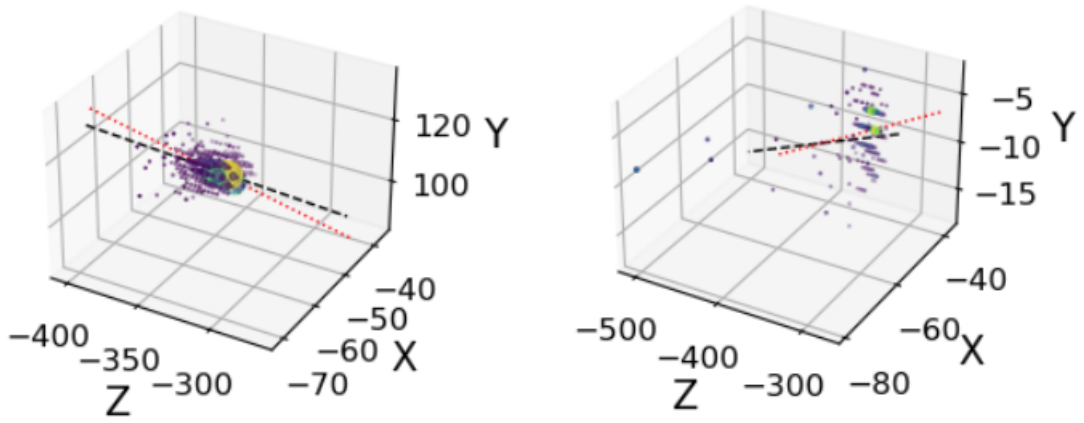


Figure 2.29. Visualization of traditional deposition axis (red) and the one based on the PCA approach (black). For both clusters, the axes almost coincide.

For future work, the intention is to investigate the use of the PCA approach in the required latency time. Research papers and articles have been published over the years, presenting various methodologies for implementing PCA in FPGAs [166, 167], although a potential problem is the speed of execution.

2.6 Model Development and Reduction Methods

The objective of this research is to develop an NN model that can accurately classify EM showers from physically uninteresting backgrounds while operating under strict constraints set by L1T. The following chapter presents the structure of NN used in the experiment.

In 1989, Cybanko in [168] proved that standard multilayer feedforward networks have the capability to approximate any continuous function of n real variables to any desired accuracy. The same year, Hornik et al. [114] showed that multilayer feedforward networks with as few as one hidden layer are universal approximators. In recent years, several state-of-the-art models like CoAtNets [169], ResNet [170], EfficientNet [171], YOLO [172], and many others have been developed. These models combine advanced techniques such as different types of convolutions, attention mechanisms, and residual connections to achieve excellent results. As shown in Figure 2.30, there is a notable trend in the growth of the number of parameters in state-of-the-art (SOTA) models. The consequence is that the high computational complexity of SOTA models and their requirements on the resources do not allow their usage in L1T conditions, where power consumption is limited, and the budget on available devices is restricted.

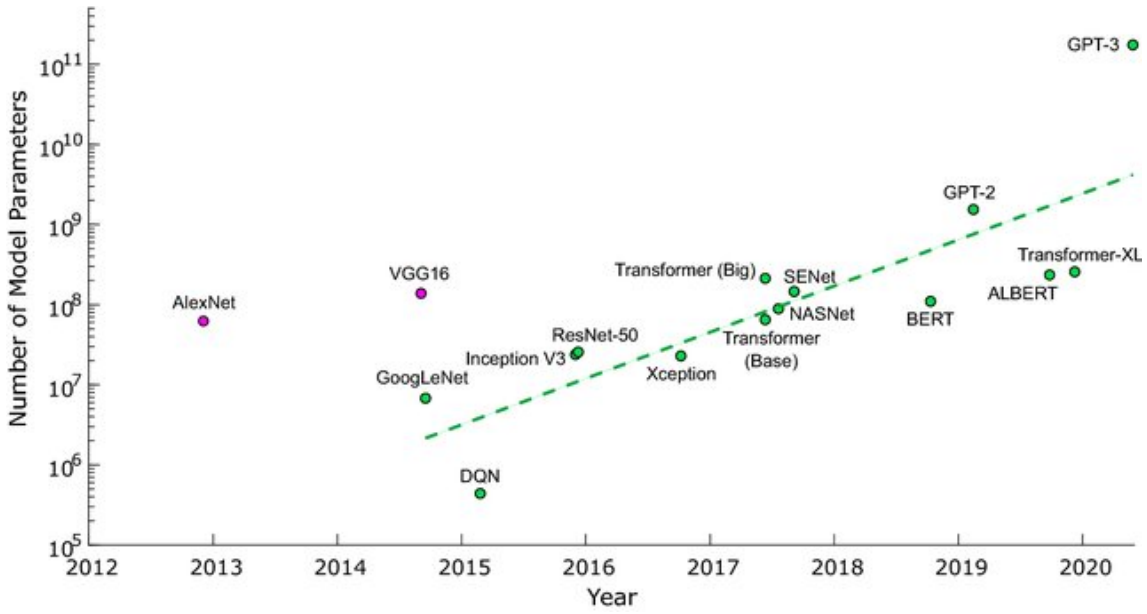


Figure 2.30. Larger neural network models require more computing power, especially in fully connected layers, but there's an ongoing effort to reduce deep neural network sizes. Despite this, there's an exponential growth in model sizes due to the need to solve increasingly complex problems with higher accuracy. The pink-colored nodes in the graph represent AlexNet and VGG16 models, now regarded as over-parameterized [173].

2.6.1 Model Architecture

The first strategy to reduce computational complexity is to limit the network size, which can be achieved by reducing the number of layers and neurons in the model. However, the model performance would be compromised in more complex tasks. Despite this limitation, it has been shown in [155, 174] that MLP with just three layers can be effectively employed in EM shower/PU classification. In both cases, the models classified the EM and PU background.

The thesis examines a similar approach, along with two other background scenarios: one with a background represented by QCD events, and another with a mixed background of both QCD and PU events, named MIX. The analysis involves two simple NN models: MLP and CNN. To ensure the optimal performance of the models, the hyperparameter optimization software framework Optuna [175] is used. By optimizing the hyperparameters, the produced models are configured in an optimal manner, leading to improved model accuracy. Optuna employs various optimization algorithms including tree-structured Parzen estimator, an algorithm that efficiently exploits the hyperparameter search space. It uses the history of previously evaluated hyperparameter configurations to sample the next set of configurations. The evaluation of these configurations is done using accuracy as preferred metric. Focusing on the most promising regions of the hyperparameter space can lead to faster convergence and improved results compared to random search or manual tuning.

The search space for the CNN consists of 6 to 16 kernels for the Conv2D layer and 8 to 18 nodes for the 1 to 2 following dense layers. This has resulted in a total of 1452 possible network combinations. On the other hand, the MLP search space involves 2 to 3 layers with 4 to 16 nodes each, generating a total of 2366 combinations. For both models, the search space contains ReLu and Sigmoid as the proposed activation functions for intermediate layers, while Softmax is used for the activation function in the output layer. The Sigmoid and ReLU activation functions were selected for their ability to introduce non-linearity and help prevent vanishing gradient problem, while Softmax can handle both binary and multi-class classification tasks. Three optimizers - Adam, RMSprop, and SGD, were chosen within the Optuna search space to balance efficiency, adaptive learning rates, and convergence speed, each with a range of learning rates. This greatly increases the number of possible combinations in the search space to 8712 for CNN and 14 196 for MLP. When multiple suggested architectures have similar accuracies, the one with the lowest number of trainable parameters is selected.

Table 2.3 displays the optimization results for the hidden layers of CNN and MLP models for every classification case. For instance, the CNN model for the EM vs. PU classification task consists of a convolutional layer with eleven 3×3 kernels, followed by one dense layer with 12 nodes. It is worth noting that the optimizer provides a fairly similar neural network structure for MLP in all classification tasks. However, the suggested CNN for EM vs. MIX is a bit larger, as it has an additional dense layer. This is expected as the classification problem

is more complicated.

For the activation function in all hidden layers, for each of the models, the optimizer suggests ReLu (while classification is performed using Softmax). For both models, the loss function is the binary cross-entropy, and optimization is conducted using the Adam algorithm, with a specific learning rate (in the range from 0.009 to 0.07) for each model and classification case obtained by the optimizer. The batch size value is 128, providing a reasonable balance between speed and accuracy, and early stopping is implemented if no progress is seen beyond fifteen epochs. Early stopping is a technique used to prevent overfitting in ML models. It works by stopping the training process if the performance on the validation set starts to decrease. Hence, it is considered a form of regularization that reduces model complexity and prevents overfitting [176].

Table 2.3. The results of model structure optimization for CNN and MLP for each classification case. The table describes hidden layers.

case/model	CNN	MLP
EM vs. PU	conv1 layer: 11 kernels dense1 layer: 12 nodes	dense1 layer: 5 nodes dense2 layer: 13 nodes
EM vs. QCD	conv1 layer: 12 kernels dense1 layer: 10 nodes	dense1 layer: 4 nodes dense2 layer: 15 nodes
EM vs. MIX	conv1 layer: 6 kernels dense1 layer: 12 nodes dense2 layer: 13 nodes	dense1 layer: 8 nodes dense2 layer: 13 nodes

The training and testing sets for all classification tasks contain balanced samples of classes. In total, the database for binary classification contains 93 000 samples. 31 000 of each class. Out of these, 27 000 samples of each type were used for training and validation, and 4 000 were used for testing. The first group of tests was conducted on the full data set without any normalization, selection, or quantization. Input images, in most cases, represent the full HGCAL profile, that is, all 36 (35) layers (as explained in Section 2.3.4). Architectures of the developed networks for EM vs MIX classification case are presented in Figure 2.31 and 2.32.

2.6.2 Model Quantization

Another strategy aiming to reduce time and memory consumption is the model quantization, a technique for reducing the precision of numerical data (in this case weights, biases, values of activation functions). The result of replacing 32-bit floating point numbers with 8-bit integers (or less) is a smaller model size. Consequently, it needs less time to load it into memory, memory consumption is reduced, and inference is faster. Using low-precision fixed integer value representation can reduce memory footprint and latency by 16× [177].

There are two main approaches to network quantization: post-training quantization and

```
          OPERATION          DATA DIMENSIONS   WEIGHTS(N)   WEIGHTS(%)

(5, 5, 35)
              Input   #####      5    5   35
             Conv2D    \|/  --------------------      1896    68.9%
(3, 3, 6)
               relu   #####      3    3    6
            Flatten   ||||| --------------------         0     0.0%
(54,)
                     #####          54
              Dense  XXXXX --------------------       660    24.0%
(12,)
               relu   #####          12
              Dense  XXXXX --------------------       169     6.1%
(13,)
               relu   #####          13
              Dense  XXXXX --------------------        28     1.0%
(2,)
            softmax   #####           2
```

Figure 2.31. ASCII diagrams showing architecture and parameters of CNN for EM vs MIX classification scenario.

```
          OPERATION          DATA DIMENSIONS   WEIGHTS(N)   WEIGHTS(%)

(875,)
              Input   #####         875
              Dense  XXXXX --------------------      7008    98.0%
(8,)
               relu   #####           8
              Dense  XXXXX --------------------       117     1.6%
(13,)
               relu   #####          13
              Dense  XXXXX --------------------        28     0.4%
(2,)
            softmax   #####           2
```

Figure 2.32. ASCII diagrams showing architecture and parameters of MLP for EM vs MIX classification scenario.

quantization-aware training. Post-training quantization is performed so that the network is trained in full precision and then quantized to lower bit widths. Although it is fast and easy to implement, it can suffer a major drawback when the precision level is set lower than 8 bits [178], making it an inappropriate option for models requiring high levels of precision. On the other hand, the quantization-aware training approach, involves quantization integrated during training, which generally gives a smaller accuracy drop compared to the previous technique. QKeras [179] and Larq [180] are libraries designed as quantization extensions of the Keras API. This thesis employs the QKeras for model quantization, which enables the development of efficient models suited for deployment on L1T resource-constrained devices.

2.6.2.1 QKeras

QKeras is a Python library that enables quantization-aware training. It uses a simple replacement for standard Keras layers, which greatly simplifies the quantization process. QKeras allows the usage of different quantization types and levels, including different activation functions, making it possible to achieve heterogeneous quantization. This library uses the concept of a straight-through estimator presented in [181]. It means the forward pass applies the quantization functions, and the backward pass adopts the quantization as the identity function to make the gradient differentiable [31]. A comprehensive research was done with different types of kernel quantizers and quantized versions of activation functions, including varying their settings (parameters). A decision is made to use the quantizer *quantized_bits* because it maintained good accuracy even with aggressive quantization and was supported by *hls4ml*. It is defined as follows

$$quantized\_bits(x) = 2^{int-b+1} clip(round(x * 2^{b-int-1}), -2^{b-1}, 2^{b-1} - 1) \tag{2.17}$$

for weights and kernels, where x is input, b is the number of bits used for the quantization, and *int* is the number of bits to the left of the decimal point. According to QKeras documentation; the syntax of this method is as follows:

$quantized\_bits(bits = 8, integer = 0, symmetric = 0, keep\_negative = 1, alpha = None, use\_stochastic\_rouding = False)$.

While the meaning of parameters *bits* and *integer* is clear, there are some additional parameters that are possible to adjust:

- *symmetric*

  The use of symmetric versions for weights and biases; this quantizer should use it to properly converge and eliminate the bias. Available values: 0 or 1.

- *keep_negative*

  Available values: 0 or 1. If the value of this parameter is 1, negative numbers are not clipped.

- *alpha*

  This parameter changes the absolute scale of the weights while keeping them discretized within the chosen bit width. Available values: $None, auto, auto\_po2$. Further details can be found in [31].

- *use_stochastic_rounding*

  When the number of bits is decreased, rounding introduces bias to the number system. To mitigate the effect of bias, Numpy rounds values to the nearest even instead of rounding up. In the [182], authors suggested using stochastic rounding, which uses

the fractional part of the number as a probability to round up or down, which can be set using this parameter.

Quantized_relu is chosen as a quantized replacement for the ReLu activation function, used in the inner layers of models, as shown in Figure 2.33, and compared to the regular ReLu function.

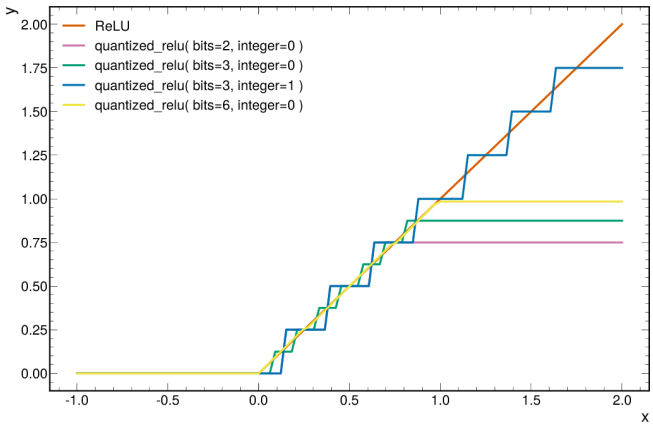


Figure 2.33. QKeras implementation of quantized_relu function in a 2- (purple), 3-bit (green and blue) and 6-bit (yellow) precision and for 0 or 1 integer bits. For comparison, the unquantized ReLU function is depicted in orange in the graph [31].

3 RESULTS AND DISCUSSION

The following chapter presents the outcomes of the model and data reduction methods described in Chapter 2. The chapter begins with the results of binary classification, followed by the results of multiclass classification. The subsequent section discusses the results obtained using alternative images, and then the chapter concludes with the presentation of results from the FPGA implementation.

3.1 Binary Classification

The following chapter presents the results of the model and data quantization and selection methods described in the previous section.

3.1.1 Base Neural Network Models

Three different binary classification tasks are conducted. In each task, the training and testing sets have an equal number of instances for each class, ensuring a balanced representation of the classes within the dataset which enables better generalization of models. There are a total of 62 000 samples in each classification task, consisting of 31 000 EM samples and 31 000 background samples (either PU or QCD). Out of these, 54 000 (87%) samples of each type were used for training and validation, and 8000 (13%) were used for testing. The initial set of experiments was performed on the dataset without any normalization, feature selection, or data quantization. Input images represent the full HGCAL profile, that is, all 36 (35) layers.

The results of baseline NN methods for classifying EM versus different types of background (PU or QCD) are shown in Table 3.1.

Table 3.1. The results for the base CNN and MLP model for EM vs. different background (PU or QCD).

	Model	Acc	Prec	Rec	F1	AUC	FPR	FNR	Param #
PU	CNN	**0.9794**	**0.9873**	**0.9712**	**0.9792**	**0.9911**	**0.0125**	**0.0288**	4702
	MLP	0.9762	0.9842	0.9680	0.9761	0.9909	0.0155	0.0320	4486
QCD	CNN	**0.9650**	**0.9723**	0.9573	**0.9647**	0.9893	**0.0272**	0.0428	4904
	MLP	0.9637	0.9603	**0.9675**	0.9639	**0.9905**	0.0400	**0.0325**	3611

According to the results shown in Table 3.1, the CNN model provides slightly better accuracy than MLP in each classification case. Although the CNN network dominance is more pronounced for EM vs PU case, the results show that based on some metrics, such as FNR, MLP performs slightly better in the EM vs QCD classification. Both models perform better at distinguishing between EM and PU clusters than between EM and QCD jets. The reason is a higher difference in average cluster energy deposits between EM and PU, as shown in Figure 3.8. In the remaining part of the thesis, for the binary classification problem, the focus is put on one vs all task, specifically EM vs. MIX. In this scenario, the background is mixed with equally distributed PU and QCD samples, which is closer to a realistic situation. The classification results are given in Table 3.2, with the corresponding ROC curve shown in Figure 3.1.

Table 3.2. The results for the base CNN and MLP model for EM vs. MIX.

Model	Acc	Prec	Rec	F1	AUC	FPR	FNR	Param #
CNN	**0.9701**	0.9688	**0.9715**	**0.9702**	**0.9919**	0.0312	**0.0285**	2753
MLP	0.9671	**0.9710**	0.9630	0.9670	0.9897	**0.0288**	0.0370	7153

The EM vs. MIX classifier, as expected, performs better than EM vs. QCD classifier, but worse than EM vs. PU.

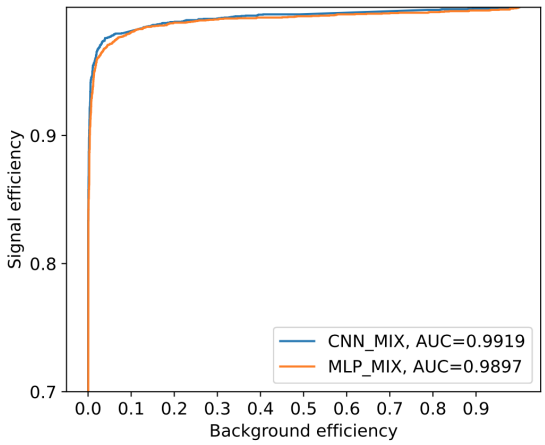
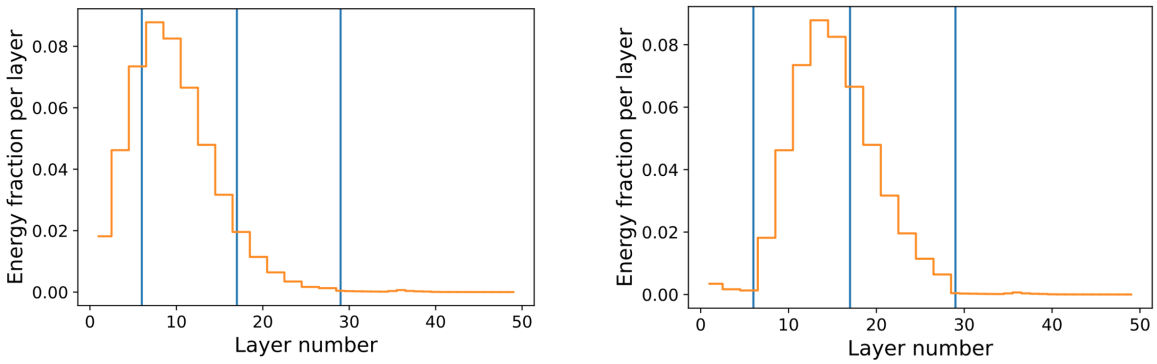


Figure 3.1. Receiver Operator Characteristic (ROC) curves for EM vs MIX classification. A calculated performance metric is the Area Under the Curve (AUC) which is the integral of the ROC curve.

Although the MLP model has almost three times more parameters than a CNN, the CNN can still consume more computational resources. The reason is convolution operations, which are computationally intensive. In contrast, MLPs use fully connected layers, where each neuron is connected to every neuron in the previous layer, which leads to a large number of parameters. Convolutional operations require complex calculations as filters are utilized to scan the input data to detect features, demanding significant memory and



(a) EM shower with CE-E profile shifted -1 layer.

(b) EM shower with CE-E profile shifted +2 layers.

Figure 3.2. Examples of shifted EM showers.

processing power. Therefore, despite having fewer parameters, CNNs often require more computational resources due to the nature of their convolutional operations.

3.1.2 Model Analysis

Model analysis involves evaluating and assessing the performance, behavior, and characteristics of an ML model. This process includes various techniques and methodologies to understand how well the model is performing, diagnose potential issues, and gain insights into its strengths and weaknesses. This section examines three critical aspects of model performance: model robustness, error analysis, and the interpretation of model decisions.

3.1.2.1 Model Robustness

To perform the robustness analysis, the tests utilize artificially introduced offsets in energy distribution profiles regarding the layer in which particular energy appears. A dataset is extended with a new set of images created by shifting CE-E energy deposits of each particle type one or two layers forward or backward. The effect of the shifting process on the EM shower longitudinal profile is shown in Figure 3.2.

The extended dataset contains 270 000 training images and 40 000 test samples, equally distributed between EM and MIX background samples. Conducted tests could be considered a type of adversarial training [183], an ML technique that trains models to be robust against adversarial examples. The examples are intentionally created with deviations from expected values (in our test, images) to mislead the model into making inaccurate predictions. Adversarial training is based on the idea that models trained on adversarial examples are more robust to real-world variations and distortions in the data. Using the adversarial approach, it is expected that the model trained on the dataset extended using the described technique can cover numerous variations and make predictions resistant to these variations. The NN models showed good robustness, maintaining accuracy for up to three layers of offsets (approx.

10% of ECAL size), as shown in Table 3.3. However, an accuracy drop was observed for offsets of four or more layers, especially in MLP.

Table 3.3. The results for the base CNN and MLP model for EM vs. MIX, dataset extended with samples shifted up to ± 2 layers.

Model	Acc	Prec	Rec	F1	AUC	FPR	FNR	Param #
CNN	**0.9728**	**0.9776**	**0.9678**	**0.9727**	**0.9903**	**0.0222**	**0.0322**	2753
MLP	0.9658	0.9643	0.9674	0.9658	0.9882	0.0358	0.0326	7153

Furthermore, in [184], we have demonstrated that the same family of NNs can adapt to change. It means that it was shown that models pre-trained on electron EM showers (which we used in our experiments) can effectively classify events containing photon EM showers. The difference between electron and photon EM showers is that electrons start showering earlier compared with photons. This results in a longitudinal profile shifted by one layer. The described findings confirm the robustness of the proposed approach.

3.1.2.2 Error Analysis

The error analysis conducted in this study revealed some potential limitations of the model. The findings suggest that the model faces difficulties in accurately predicting outcomes for low-energy EM showers, particularly in two cases: when EM starts showering in earlier layers than usual and for the EM instances wherein hadronic deposits are higher than usual. The analysis shows that the use of low-level features, such as HGCAL images and energy deposits, is not sufficient to handle these specific scenarios. Therefore, it should be considered (in some future work) to include additional features in order to achieve the desired level of accuracy.

3.1.2.3 Model Interpretation

NN model interpretation refers to understanding how the model makes decisions. In this thesis, the SHAP (Shapley additive explanations) framework [185, 186] is used to enable the models' interpretability. By using SHAP, it is possible to understand the contribution of each feature to a model's output, which helps to understand how the model makes a decision. SHAP is based on the game theory approach, where each feature is considered as a player. As a result, the framework generates SHAP values that give insight into feature importance. Shapley values fairly distribute the total payout among the players (features) based on their marginal contributions. As a replacement for the NN model, which is not easy to understand, the framework introduces a simple approximation of the original model, the so-called explanation model. Furthermore, SHAP uses Additive Feature Attribution Methods to correctly distribute the overall impact of features. The approximated output of the original model is then calculated by summing the effects of all feature attributions.

SHAP examined both MLP and CNN models, providing insights into the critical aspects that influence model predictions. After reviewing the literature and documentation for the SHAP tool, it was found that no instances of its implementation for 3D image classification were discovered. As a result, adjustments to the tool to fit HGCAL data were made, which led to the limitation of some of the (standard) SHAP functionalities. Despite this, valuable insights into the operation of the models were gained, enabling a deeper understanding of how each individual feature contributed to overall performance. The analysis helped to identify the most important features that have a significant impact on making accurate predictions. A summary of the key findings is given below:

- To enable the usage of MLP for signal/background classification based on raw detector data, the 3D HGCAL image is transformed into a 1D array of values, where each value represents a pixel. To be precise, there are $5 \times 5 \times 35 = 875$ pixels in the flattened version of the image, and these pixels can be considered as features in the classification task. In the first version of the test, the detailed position of each pixel/feature (layer, row, and column) was observed. However, the position of each pixel in a particular layer is less significant than the layer itself, therefore, in the subsequent tests, the shape values belonging to all features inside the particular layer are summed. This approach offered a more comprehensive understanding of layers' importance in the model's inference. The plot is given in Figure 3.3 showing the most significant features that affect the prediction of a single observation, along with the magnitude of the SHAP value for each feature.

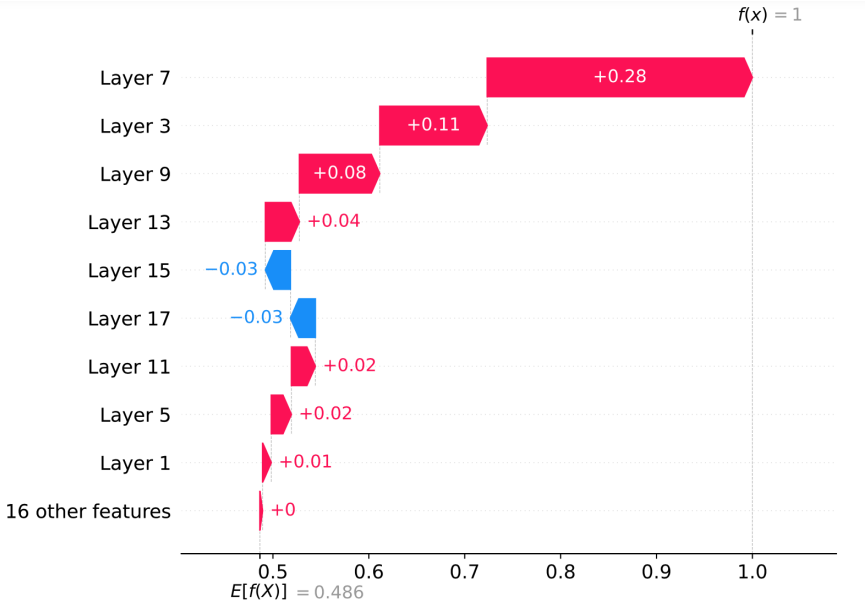


Figure 3.3. A plot showing the most influential features of the MLP model, ranked by their effect on predictions. The plot displays the SHAP value distribution for each feature, with the x-axis representing the numerical value of the feature's influence on a specific prediction.

The y-axis displays the features ranked according to their effect on the model's predictions, while the x-axis represents the range of achieved SHAP values. The plot visualizes the SHAP value for each feature, giving insight into a feature's influence on a specific prediction.

The sum of SHAP values for all the input features corresponds to the difference between the baseline (expected) model output and the current model output for the prediction being explained [187]. The color of the plot indicates whether the feature contributes (red) or withdraws (blue) to prediction. Upon closer examination of the diagram, it is noticeable that the features or pixel positions with the greatest impact on the prediction are primarily located between layers 7 and 15, with the exception of layer 3 (Figure 3.3). These are the layers in which both EM and background particles reach peak energy deposition (for reference, please see Figures 2.23a, 2.23b and 2.23c). Eventually, the NN models could be expanded with an attention mechanism to allow the models to selectively focus on relevant information, such as layers with the highest impact on prediction. This could potentially lead to improvements in performance.

- The CNN architecture was also analyzed using the SHAP framework to determine the importance of individual features. However, the SHAP documentation does not provide a mechanism for visualizing the 3D regions with the strongest influence on the model's decisions. Therefore, the custom visualization that follows standard SHAP rules is created. This visualization produces a figure wherein red shades of pixels indicate the features that lead the model to predict that an HGCAL image corresponds to an EM shower, while blue shades of pixels indicate the features that push the model away from predicting that an image belongs to the EM class. Figure 3.4 gives a visualization of an example of SHAP local interpretation, which refers to the process of calculating the SHAP values for features of a particular instance.

  This helps gain insights into which features played the most significant role in making the decision. It is evident that the majority of the (dark) red areas are concentrated in the layers where EM showers have the highest energy deposits. For reference, please see Figure 2.23a. This suggests that the model pays significant attention to these regions when predicting an EM shower.

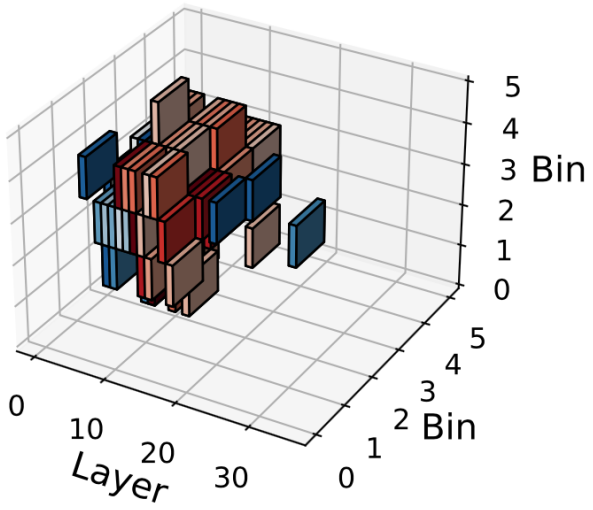


Figure 3.4. A visualization of SHAP local interpretation, calculating the SHAP values for features of an individual sample. Red-shaded pixels indicate the features that cause the model to predict that an HGCAL image depicts an EM shower, while blue-shaded pixels indicate the features that push the model away from predicting that an image belongs to a given class. (Remark: absorber layers are not shown in the figure. Therefore, there are 14 CE-E + 21 CE-H layers.).

3.1.3 Model Quantization

As explained in Section 2.6.2.1, it is decided to use QKeras's quantizer *quantized_bits* for the weights, biases, and kernels, and *quantized_relu* is for a quantized replacement for the ReLu activation function. It is important to emphasize that in this case, only the model is quantized, not the input data. To determine which level of quantization to choose, tests were performed by iterating over the quantization levels in the range from 2 to 16 (bits). The comparison of the distribution of weights for the 1st convolutional layer for models tested with different quantization levels is shown on Figure 3.5. The purpose of the tests was to examine the effect of weight/bias/kernel quantization level on model accuracy. The results of the tests revealed there is no significant difference in observed metrics, as can be seen in Table 3.4.

The comparison of accuracy and FNR for different quantization levels given in Figure 3.6 shows that accuracy is very similar for all quantization levels, while FNR varies more. Therefore, in order to reduce the memory footprint and processing requirements of the model, aggressive 2-bit quantization is chosen to be applied for both weights/biases and activation function, even if FNR is higher than, for example, for 16-bit quantized MLP model or 4-bits quantized CNN model.

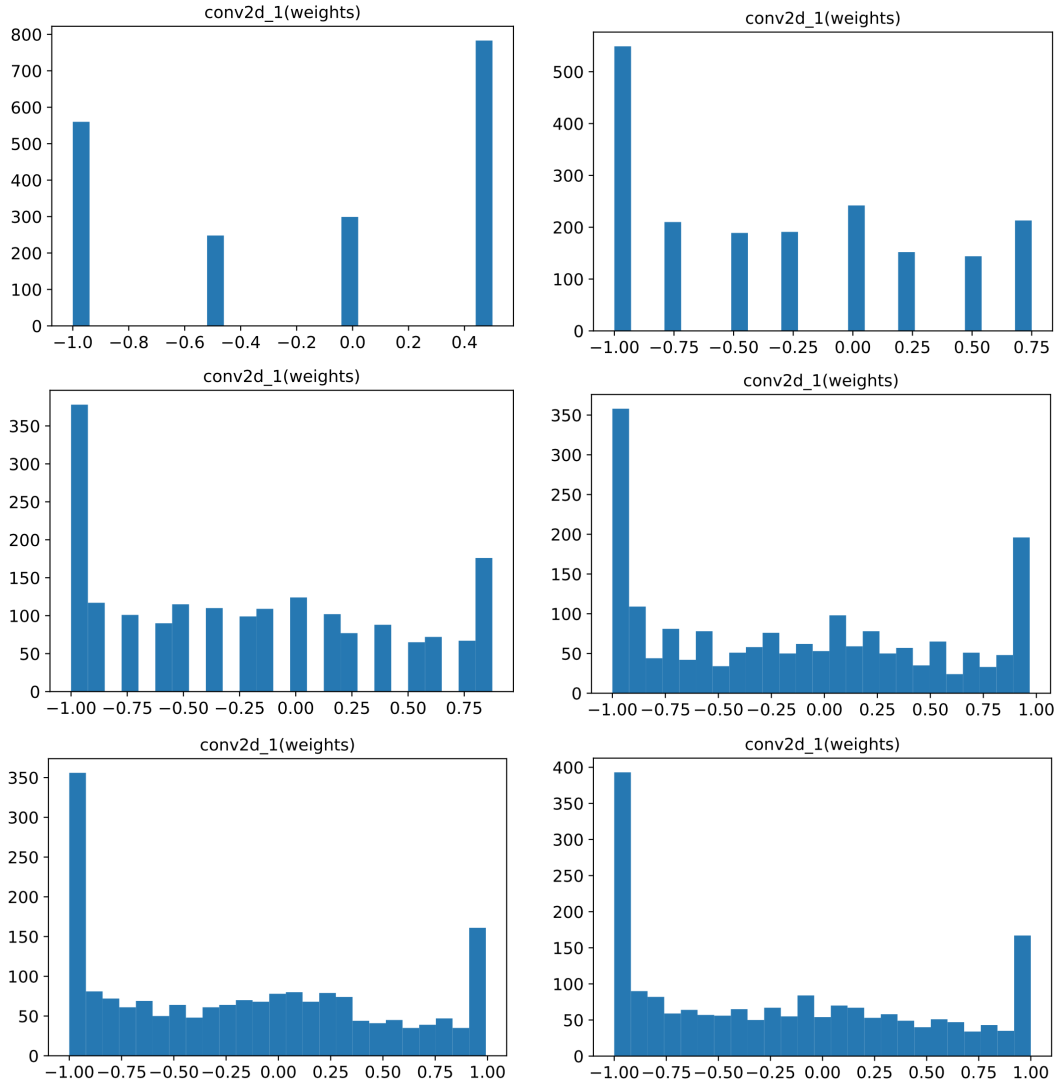


Figure 3.5. Distribution of weights for the 1st convolutional layer for each of tested model with applied different quantization levels: 2,3,4,6,8,16.

Table 3.4. The results of the CNN model and MLP for different quantization levels.

	Quant. level	Acc	Prec	Rec	F1	AUC	FPR	FNR
CNN	no quant.	0.9701	0.9688	**0.9715**	0.9702	**0.9919**	0.0312	**0.0285**
	2	0.9722	0.9807	0.9635	0.9720	0.9856	0.0190	0.0365
	3	0.9726	0.9756	0.9695	0.9725	0.9863	0.0243	0.0305
	4	0.9732	0.9752	0.9712	0.9732	0.9886	0.0248	0.0288
	6	**0.9739**	0.9832	0.9643	**0.9736**	0.9886	**0.0165**	0.0357
	8	0.9726	**0.9814**	0.9635	0.9724	0.9884	0.0182	0.0365
	16	0.9736	0.9793	0.9677	0.9735	0.9906	0.0205	0.0323
MLP	no quant.	0.9671	0.9710	0.9630	0.9670	0.9897	0.0288	0.0370
	2	**0.9720**	**0.9814**	0.9623	**0.9717**	0.9848	**0.0182**	0.0377
	3	0.9705	0.9806	0.9600	0.9702	0.9896	0.0190	0.0400
	4	0.9699	0.9772	0.9623	0.9696	0.9880	0.0225	0.0377
	6	0.9699	0.9762	0.9633	0.9697	**0.9918**	0.0235	0.0367
	8	0.9692	0.9749	0.9633	0.9691	0.9893	0.0248	0.0367
	16	0.9715	0.9775	**0.9653**	0.9713	0.9898	0.0222	**0.0348**

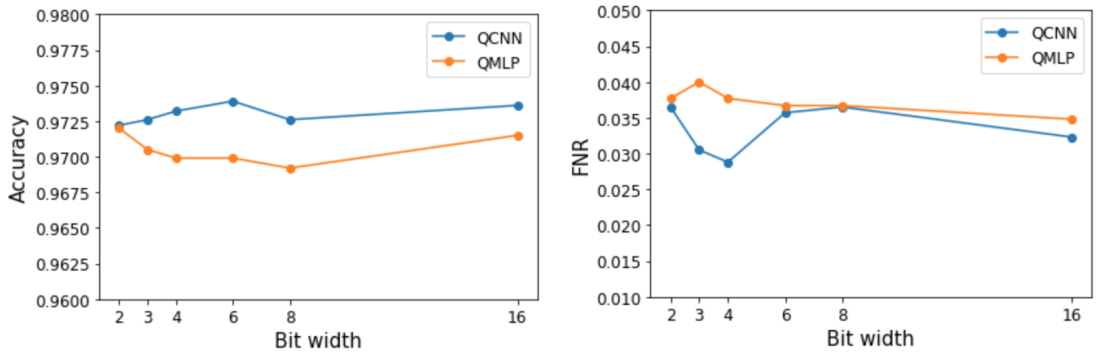


*Figure 3.6. The effect of weight/bias/kernel quantization level on CNN and MLP model: (**a**) accuracy (as a function of bit width), (**b**) FNR (as a function of bit width).*

In the rest of the thesis 2-bit quantized CNN is referred to as QCNN, and accordingly, the quantized 2-bit version of MLP is named QMLP. If there are more QCNN (QMLP) models compared, this versions of QCNN (QMLP) models are referred to as "base".

The performance of the QCNN and QMLP is summarized in Table 3.5 and Figure 3.7. The study has demonstrated that the quantized models have the same discrimination power as the baseline architectures. This is an important finding, considering the fact that trigger algorithms will be implemented on FPGAs devices with limited memory and processing abilities. Therefore, from now on, the thesis will focus only on quantized models. Comparing QCNN with QMLP, it is apparent that both models provide very similar results. However, QCNN performs slightly better in terms of all evaluated metrics. This observation confirms

that convolutional layers ensure better performance in computer vision problems.

Table 3.5. The results for the 2-bit quantized CNN and MLP models.

Model	Acc	Prec	Rec	F1	AUC	FPR	FNR	Param #
CNN	0.9701	0.9688	**0.9715**	0.9702	**0.9919**	0.0312	**0.0285**	2753
QCNN	**0.9722**	**0.9807**	0.9635	**0.9720**	0.9856	**0.0190**	0.0365	2753
MLP	0.9671	0.9710	**0.9630**	0.9670	**0.9897**	0.0288	**0.0370**	7153
QMLP	**0.9720**	**0.9814**	0.9623	**0.9717**	0.9848	**0.0182**	0.0377	7153

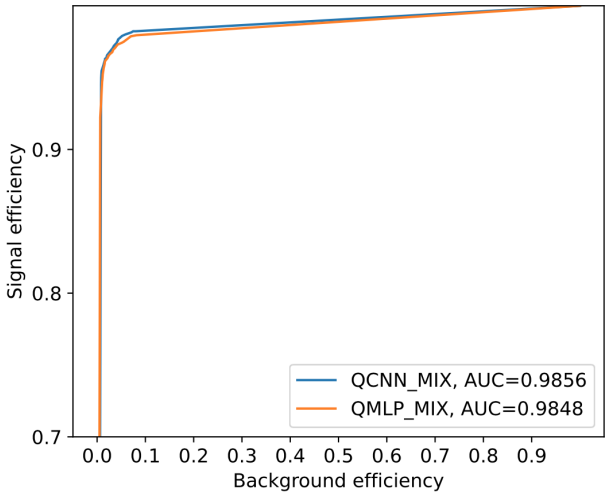


Figure 3.7. ROC curves comparison for QCNN and QMLP.

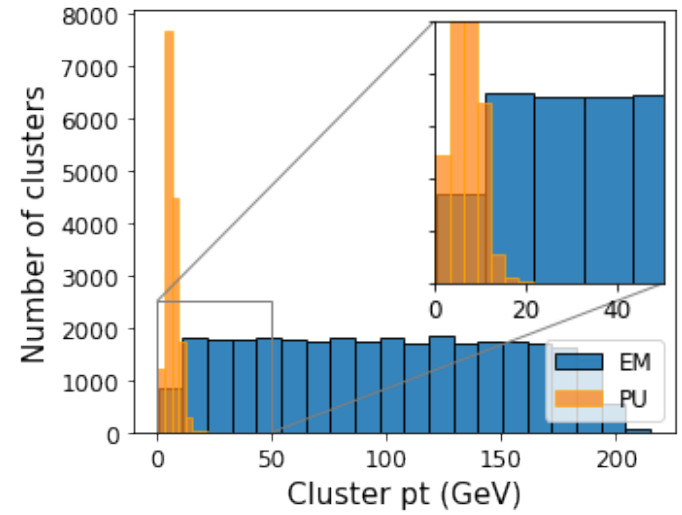
3.1.4 Comparison with Baseline Algorithms

This section evaluates the outcomes of traditional classification methods, such as cut-off, SVM, and RF, as basic algorithms for comparison with proposed NN models.

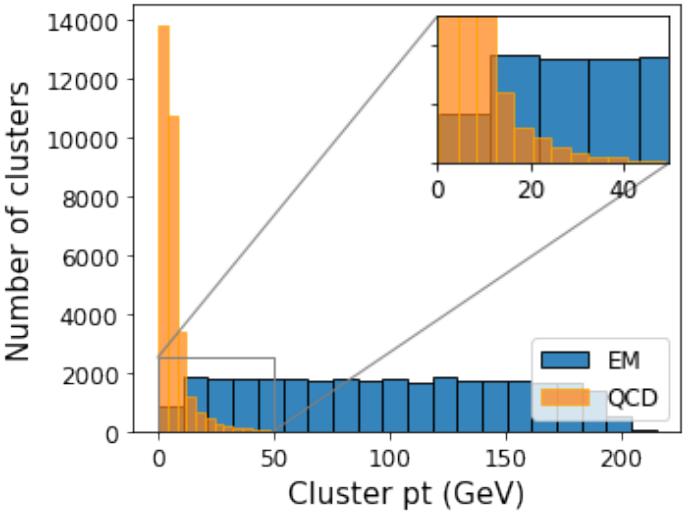
3.1.4.1 Cut-Off

In addition to the comparison with ML methods SVM and RF, NN models will be compared with a cut-off, a method of event selection often used in HEP to identify specific particles. This approach involves applying specific criteria or "cuts" on the observed events to distinguish events that correspond to the desired physical processes. The cut-based analysis is an example of linear classification, and this kind of selection is commonly used in various HEP experiments, including CMS [188]. The method applies selections to one variable at a time, which makes it robust. However, its main disadvantage is that it does not take correlations into account, which can lead to low signal efficiency and high background rate.

The data set contains ∼31 000 samples for each of the three types of clusters: EM, PU, and QCD. The comparison shown in Figure 3.8a and 3.8b highlights a significant difference



(a) For EM and PU



(b) For EM and QCD

Figure 3.8. Comparison of average clusters energies.

between the average *pt* of EM clusters and that of PU or QCD clusters. It indicates that EM clusters have higher *pt* values on average than the PU or QCD clusters.

It is possible, with high accuracy, to determine whether a cluster is EM by setting a cut-off value for its *pt* and accepting or rejecting it based on that. The reason behind this is that the low *pt* cluster is likely to be just background noise. The experiment involves using two different cut-off values. The first one is the maximum *pt* value of the PU (QCD) cluster, which is the point beyond which can be assumed that the cluster is an EM. The second cut-off value is the *pt* value of the third quartile PU (QCD) cluster, which is the point that separates the top 25% energetic of PU (QCD) clusters from the rest.

Observing EM samples, 9228 of them, or 29.82%, are low *pt* EM clusters, EM clusters with *pt* lower than maximum PU cluster *pt* that would be lost, using these criteria. Compared with QCD clusters, there are 12 527, or 40.48% low *pt* clusters. In the case of the third quar-

tile cut-off value, 222 EM clusters (0.72 %) are lost in the case of the EM/PU classification and 164 (0.53 %) in the EM/QCD classification. On the other side, using a cut-off value lower than the maximum PU (QCD) cluster *pt* would lead to the rapid growth of the number of wrongly classified PU (QCD) clusters that will then be stored unnecessarily, wasting storage space and resources.

This suggests that the L1T cannot rely solely on the cut-off applied on cluster *pt* value to determine whether to accept or reject clusters. Thus, alternative classification criteria must be employed to ensure accurate and reliable EM cluster selection.

3.1.4.2 Support Vector Machine

In the SVM part of the experiment, *scikit-learn*'s SVM module was tested with various kernels, including linear, polynomial, and radial basis function (RBF), with default parameter settings. The results showed that the selection of kernel had a major impact on the model's performance, emphasizing the significance of choosing the appropriate kernel. After analyzing the results, it is observed that the SVM model demonstrated robust performance with linear and RBF kernels (and default settings), achieving high accuracy rates and low error rates. It was found that the linear kernel performed slightly better than the RBF kernel, achieving higher accuracy rates, AUC, and lower FPR and FNR. The results for SVM using all kernel types are given in Table 3.6. Further, the combination of cross-validation and grid

Table 3.6. The results of Support Vector Machine classifier.

Kernel	Acc	Prec	Rec	F1	AUC	FPR	FNR
Linear	**0.9728**	0.9851	**0.9600**	**0.9724**	**0.9728**	0.0145	**0.0400**
Poly	0.9276	**0.9985**	0.8565	0.9221	0.9276	**0.0013**	0.1435
RBF	0.9705	0.9843	0.9563	0.9701	0.9705	0.0152	0.0437

search was used to explore different hyperparameter configurations, to find the most suitable parameters for the SVM model. The explored hyperparameter search space contains the next parameters with available values:

- *C*: 0.5, 0.1, 1, 10.
- *gamma*: 'scale', 0.1, 001.
- *kernel*: 'linear' and 'rbf'

creating altogether 24 combinations. The grid search has resulted in the following combination of tuned hyperparameters: *C*: '0.1', *gamma*: scale, *kernel*: 'linear' but with no improvement in classification accuracy. Although the best RF results achieved with the linear

kernel are comparable with QCNN results, with slightly worse AUC and FNR, it is important to remember that quantized NN enables further resource savings and increased execution speed.

3.1.4.3 Random Forest

The performance of the RF classifier with default settings has been evaluated using the same metrics as for previously tested NN models with the results presented in Table 3.7.

The hyperparameter optimization was done using the range of each of the hyperparameter values:

- *criterion*: 'gini' and 'entropy',
- *max_depth*: 8, 16, 20.
- *max_features*: 'sqrt', 'log2' or None, when the parameter value will be the same as $n\_features$.
- *n_estimators*: available values are 100, 200, 300.

Out of 54 possible combinations, the grid search has resulted in the following combination of tuned hyperparameters: *criterion*: 'gini', *max_depth*: 40, *max_features*: 'auto', *n_estimators*: 200. The achieved metrics are similar to those obtained with the predefined settings, which are shown in the same Table 3.7.

Table 3.7. The results of Random Forest classifier (with optimized parameters).

Settings	Acc	Prec	Rec	F1	AUC	FPR	FNR
default	0.9684	0.9711	0.9655	0.9683	0.9684	0.0288	0.0345
optimized	**0.9691**	**0.9723**	**0.9657**	**0.9690**	**0.9691**	**0.0275**	**0.0343**

The accomplished accuracy and AUC are lower than in the NN model's case, while FNR is slightly higher. When using an RF classifier, the importance of each feature in the model can be calculated using the feature_importance attribute calculated while the model has been trained. The resulting feature importance ranking, shown in Figure 3.9, gives an indication of which features are most crucial, which can be very useful in further analysis. Comparing with the results of the SHAP analysis shown in Figure 3.3, it is observed that some layers (7, 3, and 9) are identified as most important in both analyses/models.

The grid search was attempted to optimize the recall score, but no improvement was observed (maximizing the recall score minimizes FNR).

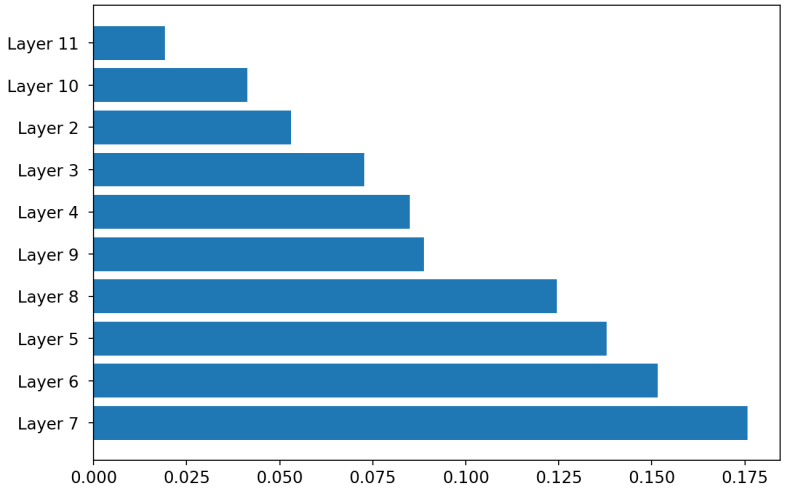


Figure 3.9. Visualization of feature analysis for optimized random forest classifier.

3.1.5 Classification Threshold Adjustment

From the results shown in Table 3.5, it is clear that two commonly used metrics to evaluate the model performance, accuracy, and AUC, are for QCNN and QMLP models above 97% (AUC above 98%), indicating good performance. Following the findings from section 2.2.4.5, the J statistic is calculated for quantized NNs. For QCNN, the largest J statistic (0.947) is achieved using a threshold value of 0.592667, and for QMLP, the maximum J value (0.945) is achieved at a threshold of 0.754915. However, this threshold value selection resulted in better FPR results but increased FNR value for both models, as seen in Table 3.8.

Table 3.8. The results of the QCNN and QML model for threshold belonging to largest J statistic.

	Thres. value	Acc	Prec	Rec	F1	AUC	FPR	FNR
QCNN	0.5 (default)	0.9722	0.9807	**0.9635**	0.9720	**0.9856**	0.0190	**0.0365**
	0.592667	**0.9731**	**0.9829**	0.9630	**0.9729**	**0.9856**	**0.0168**	0.0370
QMLP	0.5 (default)	0.9720	0.9814	**0.9623**	0.9717	**0.9848**	0.0182	**0.0377**
	0.754915	**0.9725**	**0.9834**	0.9613	**0.9722**	**0.9848**	**0.0163**	0.0387

On the other hand, aiming to optimize FNR with the trial and error approach, using a threshold value of 0.1, FNR decreases to 0.026 for QCNN and to 0.0345 for QMLP. Although the differences in achieved FNR may not look significant, it is important to remember that the volume of the data set on which the model is intended to be used is very high. Therefore, even a small decrease in FNR (increase in recall) results in a higher number of correctly classified EM showers. For example, for QCNN, the difference between the best achieved FNR and FNR obtained for the default threshold is 0.0105 (1.05%). Applied to a data set with just 1,000,000 samples, it ends with 10,500 additionally recognized EM clusters and

some of them could have a significant role in some new physical discovery. A comparison of results obtained using different thresholds is given in Table 3.9.

Table 3.9. The results of the QCNN and QMLP model for different threshold values.

	Thres. value	Acc	Prec	Rec	F1	AUC	FPR	FNR
QCNN	0.1	0.9670	0.9606	**0.9740**	0.9672	**0.9856**	0.0400	**0.0260**
	0.3	0.9691	0.9678	0.9705	0.9692	**0.9856**	0.0323	0.0295
	0.5 (default)	0.9722	0.9807	0.9635	0.9720	**0.9856**	0.0190	0.0365
	0.7	**0.9729**	0.9856	0.9597	**0.9725**	**0.9856**	0.0140	0.0403
	0.9	0.9726	**0.9904**	0.9545	0.9721	**0.9856**	**0.0092**	0.0455
QMLP	0.1	0.9704	0.9750	0.9655	0.9702	**0.9848**	0.0248	**0.0345**
	0.3	0.9710	0.9779	0.9637	0.9708	**0.9848**	0.0217	0.0362
	0.5 (default)	0.9720	0.9814	0.9623	0.9717	**0.9848**	0.0182	0.0377
	0.7	**0.9725**	0.9834	0.9613	**0.9722**	**0.9848**	0.0163	0.0387
	0.9	0.9715	**0.9861**	**0.9565**	0.9711	**0.9848**	**0.0135**	0.0435

The result now provides evidence that even though the obtained FNR (3.65% for QCNN and 3.77% for QMLP) and FPR (1.9% for QCNN and 1.82% for QMLP) are not high for the base quantized models with default threshold, they can be further optimized using the thresholding technique. It is important to notice that decreasing the number of FPs by changing the threshold leads to increasing the number of FNs, and vice versa.

3.1.6 Data Refinement

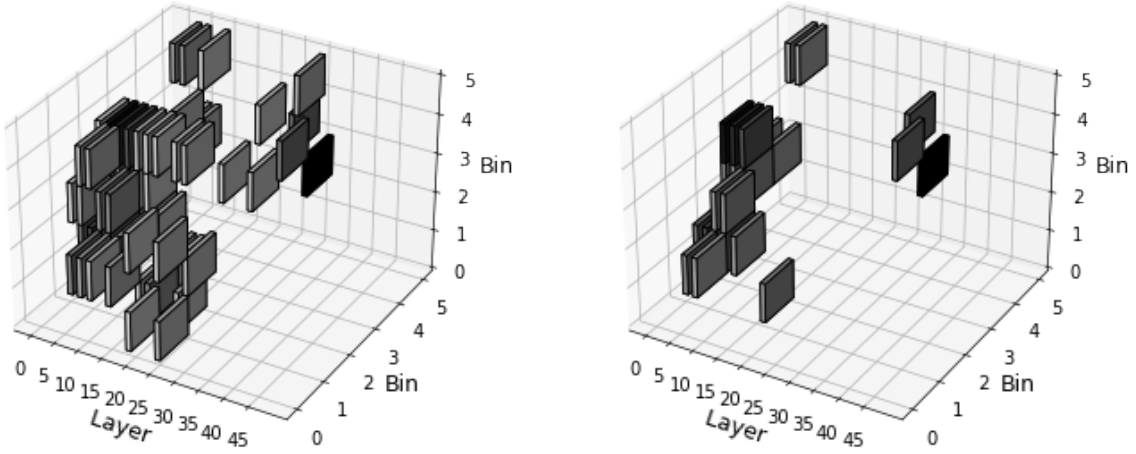
This section describes the effect of different data refinement methods on NN model performance. To ensure the effectiveness of chosen techniques, exhaustive testing was carried out. This testing covered various scenarios and conditions to determine the applicability of each technique in different situations.

3.1.6.1 Feature Selection

1. **Keep energies above the threshold.**

   Applying this method, the number of non-zero values in each dataset sample significantly drops, as can be seen in Figure 3.10. After analyzing the entire training dataset before preprocessing, consisting of full-precision values, it was found that only 11.39% of the values were non-zero. By increasing the threshold value, the percent of non-zero values decreases: for threshold value 0.1 GeV to 6.18%, for 0.5 GeV to 1.73%, for 1 GeV to 1,05%, and finally for threshold value 2 GeV to 0.62%. As shown in Table 3.10, as the threshold value increases, there is a slight drop in accuracy and

an increase in FNR value. For threshold value of 2, accuracy drops significantly and therefore, forthcoming tests are not carried out for threshold values higher than 2.



(a) Before applying the threshold value of 0.1. *(b) After applying the threshold value of 0.1.*

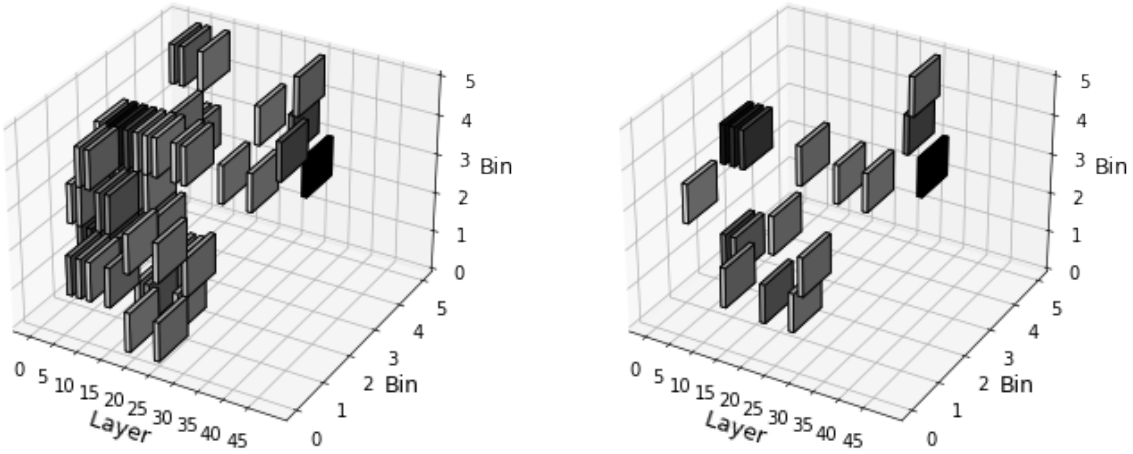
Figure 3.10. Comparison of generated 3D HGCAL image before and after applied threshold.

Table 3.10. The QCNN and QMLP results with different thresholds applied on energie/pixel values.

	Threshold (GeV)	Acc	Prec	Rec	F1	AUC	FPR	FNR
QCNN	no thres.	**0.9722**	0.9807	**0.9635**	**0.9720**	0.9856	0.0190	**0.0365**
	0.1	0.9719	**0.9856**	0.9577	0.9715	**0.9866**	**0.0140**	0.0423
	0.5	0.9699	0.9789	0.9605	0.9696	0.9837	0.0208	0.0395
	1	0.9633	0.9635	0.9630	0.9632	0.9820	0.0365	0.0370
	2	0.9511	0.9850	0.9163	0.9494	0.9602	**0.0140**	0.0838
QMLP	no thres.	**0.9720**	**0.9814**	0.9623	**0.9717**	0.9848	**0.0182**	0.0377
	0.1	0.9701	0.9786	0.9613	0.9699	0.9798	0.0210	0.0387
	0.5	0.9660	0.9667	**0.9653**	0.9660	**0.9851**	0.0333	**0.0348**
	1	0.9639	0.9727	0.9545	0.9635	0.9813	0.0267	0.0455
	2	0.9497	0.9759	0.9223	0.9483	0.9615	0.0227	0.0777

2. **Keep maximum layer energy (single sample per layer).**

   The effect of keeping layer maximum value on HGCAL image can be seen in Figure 3.11.



(a) Before applying the highest in layer approach.

(b) After applying the highest in layer approach.

Figure 3.11. Comparison of generated 3D HGCAL image before and after applied highest in layer approach.

The models that keep only one value per image layer show no significant decrease in classification accuracy compared to the base quantized versions, according to results shown in Table 3.11

Table 3.11. The results for the keep layer maximum approach for QCNN and QMLP.

	Model	Acc	Prec	Rec	F1	AUC	FPR	FNR
QCNN	base	**0.9722**	**0.9807**	0.9635	**0.9720**	**0.9856**	0.0190	0.0365
	layer max.	0.9689	0.9718	**0.9657**	0.9688	0.9847	0.0280	**0.0343**
QMLP	base	**0.9720**	**0.9814**	**0.9623**	**0.9717**	**0.9848**	**0.0182**	**0.0377**
	layer max.	0.9675	0.9761	0.9585	0.9672	0.9827	0.0235	0.0415

3. **Capping.**

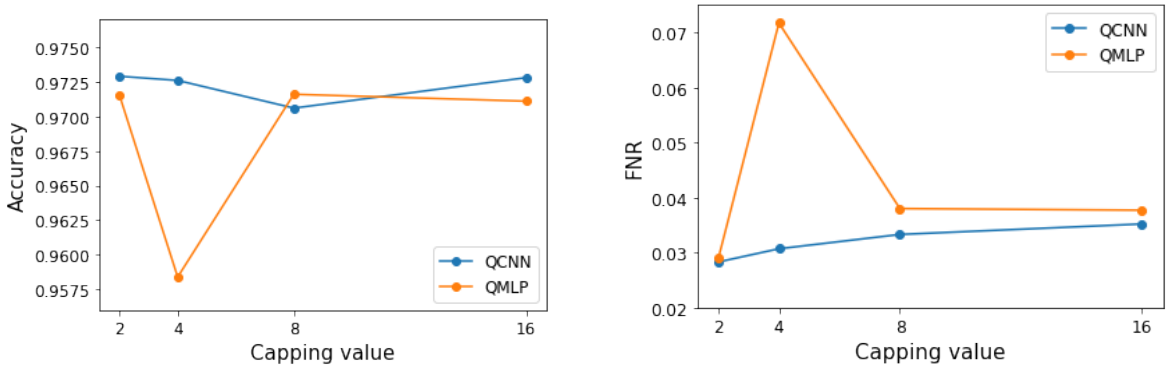
   Data capping is applied by using values (2, 4, 8, 16). For QMLP, there is no notable difference in metrics, but for QCNN, there is an improvement in FNR for all capping values, no matter which capping value is chosen (Table 3.12).

   Due to resource constraints, hardware implementations, including FPGAs, typically use fixed-point arithmetic rather than floating-point arithmetic. By limiting the range of input data values, the fixed-point representation can accurately represent all possible input values without loss of precision. Therefore, NN models with applied data capping will be chosen for FPGAs implementation. The impact of the applied capping value on the model's accuracy and FPR, shown in Figure 3.12, manifests an anomaly

Table 3.12. The results for the QCNN and QMLP for different capping values.

	c_value (GeV)	Acc	Prec	Rec	F1	AUC	FPR	FNR
QCNN	(base)	0.9722	**0.9807**	0.9635	0.9720	0.9856	**0.0190**	0.0365
	2	**0.9729**	0.9739	**0.9718**	**0.9728**	0.9864	0.0260	**0.0283**
	4	0.9726	0.9758	0.9692	0.9725	**0.9871**	0.0240	0.0307
	8	0.9706	0.9743	0.9667	0.9705	0.9850	0.0255	0.0333
	16	0.9728	0.9804	0.9647	0.9725	0.9867	0.0192	0.0352
QMLP	(base)	**0.9720**	0.9814	0.9623	**0.9717**	**0.9848**	0.0182	0.0377
	2	0.9715	0.9720	**0.9710**	0.9715	0.9822	0.0280	**0.0290**
	4	0.9584	**0.9878**	0.9283	0.9571	0.9835	**0.0115**	0.0717
	8	0.9716	0.9809	0.9620	0.9713	0.9801	0.0187	0.0380
	16	0.9711	0.9796	0.9623	0.9709	0.9842	0.0200	0.0377

in the case of capping value 4. There is a significant drop in accuracy and an increase in FNR value, which was not noticed when tests were performed with the dataset containing fewer samples.



(a) Model accuracy as a function of capping value.

(b) Model FNR as a function of capping value.

Figure 3.12. Model accuracy and FNR as a function of capping value.

4. **Reduction in longitudinal profile.**

This approach is applied only when QCNN is used as a classifier. The HGCAL image contains $5 \times 5 \times 35 = 875$ pixels per image, resulting in 2753 parameters for QCNN (respectively 7153 for QMLP). Following the procedures described in Section 4, an attempt is made to reduce the number of model parameters. The proposed solutions have been analyzed, and the outcomes are presented in Table 3.13.

- The QCNN model using a full HGCAL image has 2753 parameters, and with this approach (E+Hf), the number of parameters is reduced to 1673. Despite the

Table 3.13. The results for the QCNN applied after reduction of the longitudinal profile.

model	Acc	Prec	Rec	F1	AUC	FPR	FNR	Param #
base	**0.9722**	**0.9807**	0.9635	**0.9720**	0.9856	**0.0190**	0.0365	2753
E+Hf	0.9710	0.9705	**0.9715**	0.9710	0.9854	0.0295	**0.0285**	1673
3Ef+Hf	0.9692	0.9796	0.9585	0.9689	**0.9876**	0.0200	0.0415	1079
Ef+Hf	0.9570	0.9505	0.9643	0.9573	0.9808	0.0503	0.0357	971

fact that the model based on the reduced image uses just ∼60.77% of a complete number of parameters working with a complete HGCAL image, it is still effective in producing accurate results.

- The approach 3Ef+Hf replaces the full HGCAL image ($5 \times 5 \times 35$) with a $5 \times 5 \times 4$ image, decreasing the NN parameter numbers to 1079. In this scenario, the reduced model uses ∼39.19% of a complete number of parameters. The accuracy is slightly decreased, but there is a notable decrease in FNR.
- This selection method, named Ef+Hf, is based on two 5×5 image, and uses 971 parameters(∼ 35.27%). A more serious drop in accuracy can be noticed, while FNR is slightly better than in the 3Ef+Hf case.

3.1.6.2 Quantization

In this section, different data quantization approaches are presented.

1. **Selection bit(s).**

   It is clear that the selection bit method would drastically reduce the amount of data needed for acceptance/rejection decisions, using one bit for each pixel presentation instead of floating point format. The drawback is poor accuracy, especially for QMLP, and a significant increase in FPR for both models, as could be seen in Table 3.14.

Table 3.14. The results of the selection bit method for the QCNN and QMLP.

	Model	Acc	Prec	Rec	F1	AUC	FPR	FNR
QCNN	base	**0.9722**	**0.9807**	**0.9635**	**0.9720**	**0.9856**	**0.0190**	**0.0365**
	select. bit	0.9429	0.9562	0.9283	0.9420	0.9669	0.0425	0.0717
QMLP	base	**0.9720**	**0.9814**	**0.9623**	**0.9717**	**0.9848**	**0.0182**	**0.0377**
	select. bit	0.9285	0.9180	0.9410	0.9294	0.9662	0.0840	0.0590

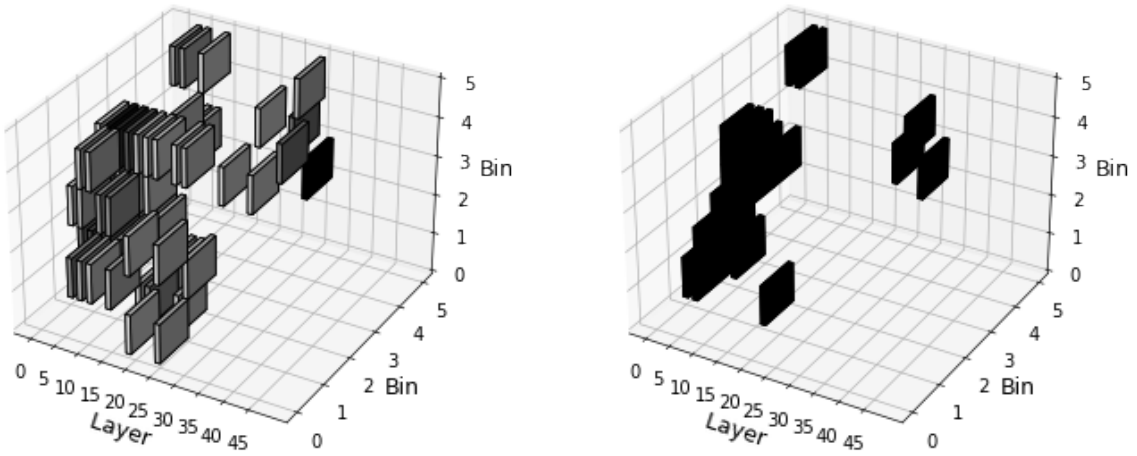
However, acceptable results are achieved when data thresholding is applied prior selection bit. According to performance metrics shown in Table 3.15, increasing the threshold value leads to improvement in accuracy. A scenario with a threshold value of 0.5

followed by the selection bits method for QCNN gives results comparable with those achieved with complete full precision data. For threshold value 1, QCNN achieves slightly better accuracy, but FNR is decreased, while QMLP behaves very similarly for threshold values 0.5 and 1.

Table 3.15. The results of the selection bit method applied after thresholding for the QCNN and QMLP.

	Threshold (GeV)	Acc	Prec	Rec	F1	AUC	FPR	FNR
QCNN	/ (base)	0.9722	0.9807	0.9635	0.9720	0.9856	0.0190	0.0365
	0.1	0.9577	0.9633	0.9517	**0.9575**	0.9794	0.0362	0.0483
	0.5	0.9667	0.9696	**0.9637**	0.9666	**0.9834**	0.0302	**0.0362**
	1	**0.9686**	**0.9813**	0.9555	0.9682	0.9781	**0.0182**	0.0445
QMLP	/ (base)	0.9720	0.9814	0.9623	0.9717	0.9848	0.0182	0.0377
	0.1	0.9495	0.9636	0.9343	0.9487	0.9738	0.0352	0.0658
	0.5	0.9653	**0.9839**	0.9460	0.9646	**0.9819**	**0.0155**	0.0540
	1	**0.9657**	**0.9839**	**0.9470**	**0.9651**	0.9807	**0.0155**	**0.0530**

The comparison in Figure 3.13 demonstrates the difference between the full HGCAL image and an image where each pixel is described using 1-bit.



(a) Before applying the selection bit approach preceded by thresholding 0.1.

(b) After applying the selection bit approach preceded by thresholding 0.1.

Figure 3.13. Comparison of generated 3D HGCAL image before and after applied selection bit approach.

2. **Uniform quantizer.**

Before applying uniform quantization to HGCAL images, it is important to perform a mandatory step - capping. In the (uniform) quantization process, a continuous range

of input values is mapped to a finite number of output values. In the scenario when the input range is too wide, the low values will be lost, becoming zero after applied quantization. By capping the range of input values, the quantization process is more effective in preserving the details of the original image. The impact of the applied capping value on the different levels of quantization is shown in Table 3.16 and Table 3.17. For both models, the AUC value does not vary a lot, regardless of the combination of capping and bit-width values. However, for capping values 8 and 16, accuracy decreases while the FNR value grows. From the results, it is clear that models with 2 and 3 input data bit width outperform models with wider data precision.

Fig 3.14 shows model accuracies and FNR depending on the capping value applied before uniform quantization.

Table 3.16. Uniform quantizer with different capping values for various quantization levels, for the QCNN.

Bit width	Cap	Acc	Prec	Rec	F1	AUC	FPR	FNR
(base)	/	0.9722	0.9807	0.9635	0.9720	0.9856	0.0190	0.0365
2	2	0.9701	0.9736	**0.9665**	0.9700	0.9832	0.0262	0.0335
3	2	0.9666	0.9656	0.9677	0.9667	0.9853	0.0345	**0.0323**
4	2	0.9705	0.9765	0.9643	0.9703	0.9877	0.0232	0.0357
8	2	0.9403	0.9431	0.9370	0.9401	0.9715	0.0565	0.0630
2	4	0.9679	0.9761	0.9593	0.9676	0.9830	0.0235	0.0408
3	4	**0.9724**	0.9790	0.9655	**0.9722**	0.9880	0.0208	0.0345
4	4	0.9714	0.9777	0.9647	0.9712	0.9878	0.0220	0.0352
8	4	0.9454	0.9694	0.9197	0.9439	0.9727	0.0290	0.0803
2	8	0.9616	0.9827	0.9397	0.9608	0.9734	0.0165	0.0602
3	8	0.9685	0.9728	0.9640	0.9684	**0.9861**	0.0270	0.0360
4	8	0.9698	0.9776	0.9615	0.9695	0.9883	0.0220	0.0385
8	8	0.9509	0.9562	0.9450	0.9506	0.9768	0.0432	0.0550
2	16	0.9407	**0.9903**	0.8902	0.9376	0.9448	**0.0088**	0.1098
3	16	0.9667	0.9819	0.9510	0.9662	0.9767	0.0175	0.0490
4	16	0.9688	0.9730	0.9643	0.9686	0.9883	0.0267	0.0357
8	16	0.9616	0.9693	0.9535	0.9613	0.9833	0.0302	0.0465

Based on the results shown in Fig 3.6, both QCNN and QMLP achieve better accuracy and FNR for a lower number of chosen bit widths (2-4) for data quantization. The capping values 2 and 4 provide better results compared to 8 and 16. These unexpected results could potentially be explained as the outcome of the reduced precision acting as a form of regularization. Therefore, 2 is chosen as the capping value for QCNN and 4 for QMLP whenever uniform quantization is used.

Table 3.17. Uniform quantizer with different capping values for various quantization levels, for the QMLP.

Bit width	Cap	Acc	Prec	Rec	F1	AUC	FPR	FNR
(base)	/	0.9720	0.9814	0.9623	0.9717	0.9848	0.0182	0.0377
2	2	0.9684	0.9759	0.9605	0.9681	0.9781	0.0238	0.0395
3	2	0.9619	0.9652	0.9583	0.9617	0.9840	0.0345	0.0418
4	2	0.9605	0.9663	0.9543	0.9603	0.9838	0.0333	0.0457
8	2	0.9395	0.9478	0.9303	0.9389	0.9677	0.0512	0.0698
2	4	0.9601	0.9499	**0.9715**	0.9606	0.9844	0.0512	**0.0285**
3	4	0.9649	0.9728	0.9565	0.9646	0.9868	0.0267	0.0435
4	4	0.9627	0.9758	0.9490	0.9622	0.9857	0.0235	0.0510
8	4	0.9439	0.9411	0.9470	0.9440	0.9703	0.0592	0.0530
2	8	0.9615	0.9800	0.9423	0.9607	0.9751	0.0192	0.0578
3	8	0.9665	0.9630	0.9702	0.9666	0.9874	0.0372	0.0297
4	8	**0.9686**	0.9711	0.9660	**0.9685**	**0.9887**	0.0288	0.0340
8	8	0.9505	0.9597	0.9405	0.9500	0.9726	0.0395	0.0595
2	16	0.9340	**0.9974**	0.8702	0.9295	0.9461	**0.0022**	0.1298
3	16	0.9639	0.9798	0.9473	0.9633	0.9813	0.0195	0.0527
4	16	0.9681	0.9708	0.9653	0.9680	0.9875	0.0290	0.0348
8	16	0.9559	0.9667	0.9443	0.9554	0.9800	0.0325	0.0558

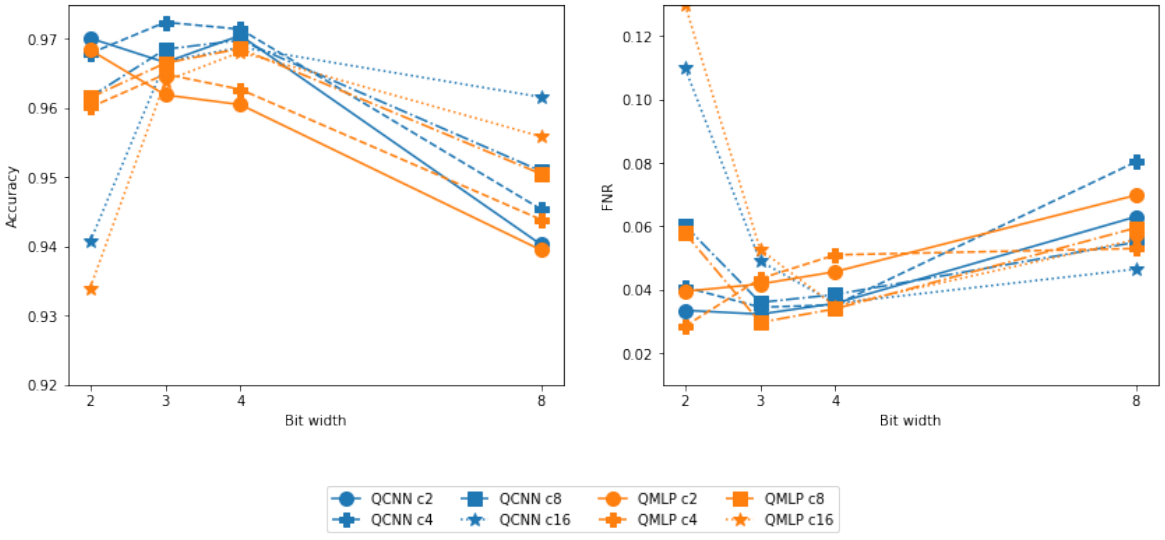
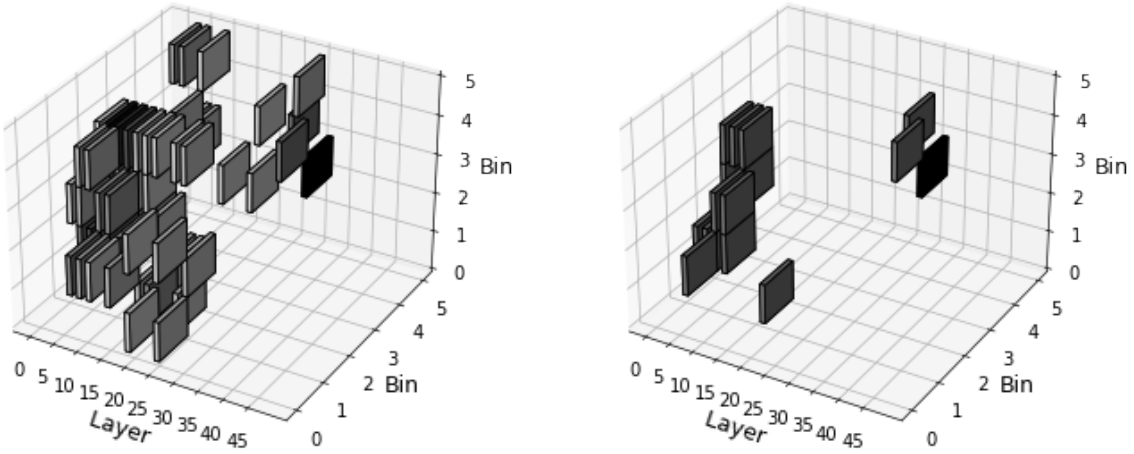


Figure 3.14. Model accuracy and FNR as a function of data bit width and capping value.

A comparison of the full HGCAL image and image with a 4-bit uniformly quantized data can be seen in Figure 3.15.

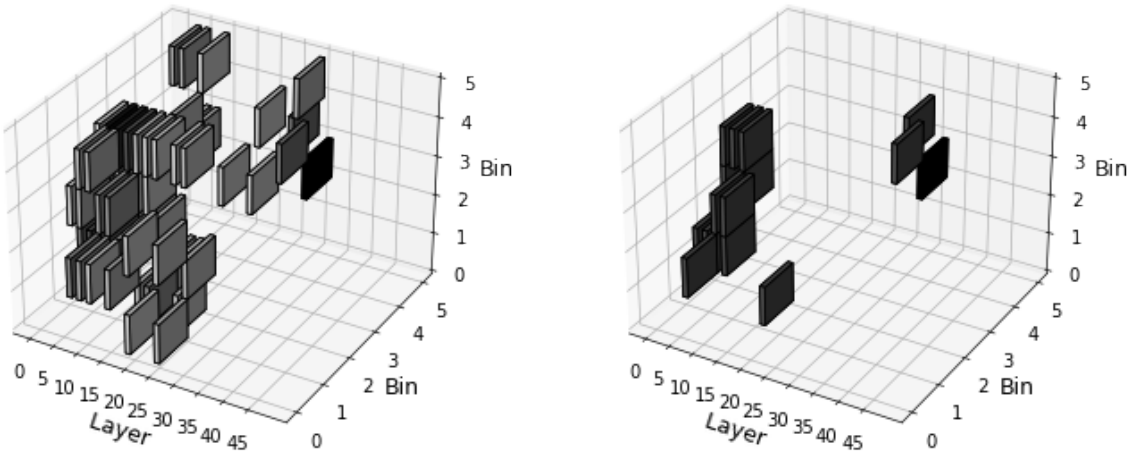
3. **Non-uniform quantizer.** Observation of the difference between a full HGCAL image



(a) Before 4-bit uniform quantization. *(b) After 4-bit uniform quantization.*

Figure 3.15. Comparison of generated 3D HGCAL image before and after applied uniform quantization.

and an image with non-uniformly quantized data is shown in Figure 3.16.



(a) Before non-uniform quantization. *(b) After non-uniform quantization.*

Figure 3.16. Comparison of generated 3D HGCAL image before and after applied non-uniform quantization.

The models based on non-uniformly quantized data with results in Table 3.18 show results similar to scenarios with uniformly quantized input. It is important to note that in the case of non-uniform quantization, the maximum number of bits is not predetermined. For QCNN, the results are comparable to 3-bit uniform quantization with a capping value of 8, and in the case of QMLP, results are comparable with those achieved with 3-bit quantization and a capping value of 8.

Table 3.18. The results of nonuniform quantizer.

	Model	Acc	Prec	Rec	F1	AUC	FPR	FNR
QCNN	base	**0.9722**	**0.9807**	0.9635	**0.9720**	0.9856	**0.0190**	0.0365
	nonuni. quant.	0.9686	0.9723	**0.9647**	0.9685	**0.9859**	0.0275	**0.0352**
QMLP	base	**0.9720**	**0.9814**	**0.9623**	**0.9717**	**0.9848**	**0.0182**	**0.0377**
	nonuni. quant.	0.9676	0.9739	0.9610	0.9674	0.9810	0.0257	0.0390

In this chapter, various data reduction methods based on selection and quantization are explored, and their effects on the accuracy of the algorithms are examined. The results showed that some of the individual methods, for example, capping and uniform quantization, achieve accuracy comparable to the accuracy obtained when the algorithms use the entire data with full precision. On the other hand, most longitudinal profile reduction methods (except E+Hf) and non-uniform quantization resulted in decreased accuracy, while the thresholding and selection-bit approach suffer from increased FNR. These findings provide insight into the potential advantages and limitations of applying different data reduction techniques in L1T algorithms. For a more detailed analysis and implementation in selected hardware, it is desirable to choose methods whose processing would be the fastest, with minimal resource consumption, in order to meet set hardware and latency requirements. Some combinations of uniform quantization with capping maintained high accuracy and reduced memory consumption, making it the most suitable method for further experiments due to its balance of performance and efficiency.

3.2 Multiclass Classification

In this chapter, an NN model to distinguish between four different categories (EM showers, PU, QCD jets, and pion samples) of particle showers resulting from proton-proton collisions are considered. The overview of the class distribution in the dataset given in Figure 2.19, helps to understand the balance or imbalance of class representation. This information is crucial for understanding the performance of classification models and guiding decision-making processes during model training and evaluation. According to Figure 2.19 in the experiment, there is a lack of pion samples. As a result, this class may be harder to predict accurately compared to the other classes (although, in real-life detector analysis, the EM showers will be the class overwhelmed with different kinds of uninteresting decay products). Imbalanced classes pose a significant challenge to achieving accurate results in ML as they cause different difficulties. One of the main issues is that most standard ML algorithms, including NNs, operate by trying to maximize classification accuracy. This can lead to misleading information about the classifier's performance, as it may appear to perform

well based on overall accuracy but actually perform poorly on the minority class. Studies on class imbalance classification have gained more emphasis in recent years as many real-world applications suffer from these phenomena [189–193].

3.2.1 Base four-class Classification

In order to achieve the best possible performance of the models, as it is done in binary classification cases, the hyperparameter optimization software framework Optuna [175] is used. The CNN model search space is defined by a range of 8 to 32 kernels for the 1-2 Conv2D layers and 8 to 32 nodes for the subsequent 1 to 3 dense layers. On the other hand, the MLP model search space involves 2 to 6 layers with 8 to 32 nodes each. Both models use ReLu and sigmoid as the suggested activation functions for intermediate layers, with Softmax being used for the activation function in the output layer. Additionally, the search space consists of three optimizers - Adam, RMSprop, and SGD, each with a variety of learning rates to choose from. When multiple architectures have similar accuracies, the one with the fewest trainable parameters is chosen.

In the case of CNN, the best results are achieved for the model with two convolutional layers with 30 and 17 kernels, followed by two dense layers with 26 nodes each. For MLP, Optuna suggests a five-layer model with a number of nodes as follows: 22, 8, 32, 24, and 22. Optimizer Adam is selected in both cases as the most favorable optimizer, with a learning rate value of 0.0083293 for CNN and 0.00719526 for MLP. Architectures of the developed multiclass classifiers are presented in Figure 3.17 and 3.18.

```
          OPERATION          DATA DIMENSIONS   WEIGHTS(N)   WEIGHTS(%)

(5, 5, 35)
              Input   #####      5    5   35
             Conv2D    \|/  --------------------      9480    61.7%
(3, 3, 30)
               relu   #####      3    3   30
             Conv2D    \|/  --------------------      4607    30.0%
(1, 1, 17)
               relu   #####      1    1   17
            Flatten   ||||| --------------------         0     0.0%
(17,)
                      #####          17
              Dense   XXXXX --------------------       468     3.0%
(26,)
               relu   #####          26
              Dense   XXXXX --------------------       702     4.6%
(26,)
               relu   #####          26
              Dense   XXXXX --------------------       108     0.7%
(4,)
            softmax   #####           4
```

Figure 3.17. ASCII diagrams showing architecture and parameters of CNN for multiclass classification scenario.

According to the metrics presented in Table 3.19, it can be observed that both models have achieved a considerably low accuracy of around 74%. Further analysis is required to identify the potential factors affecting the performance of the models and to improve their accuracy. The confusion matrix displayed in Figure 3.19 reveals that both models exhibit

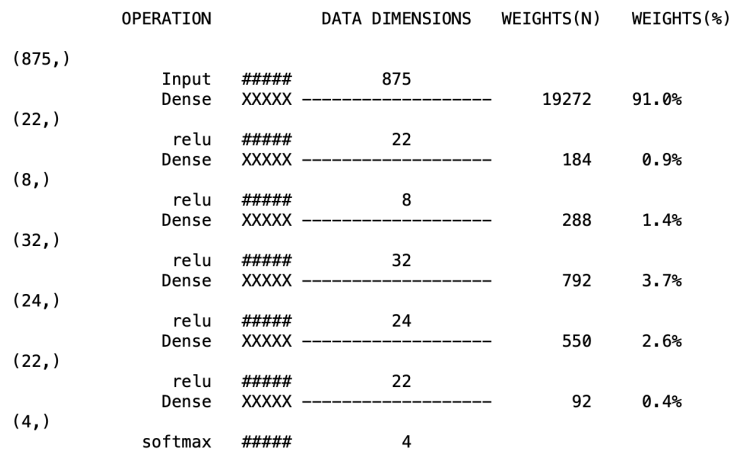


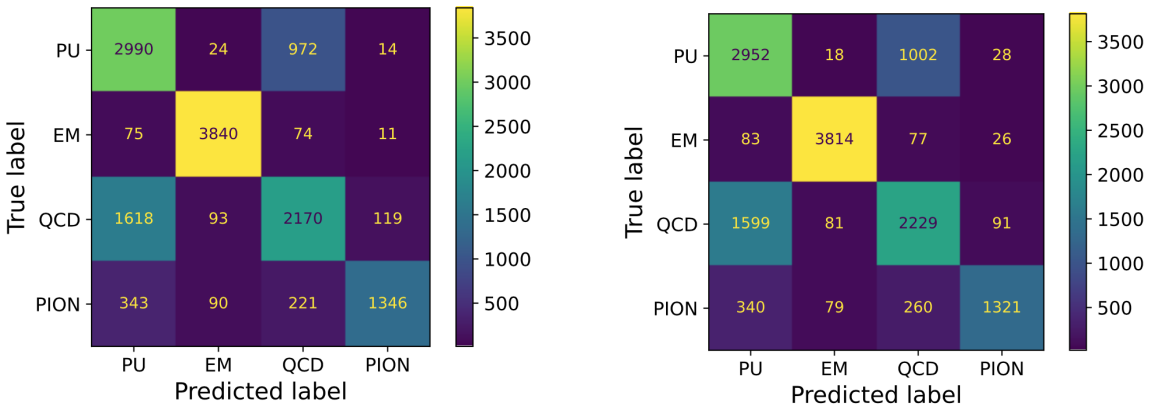
Figure 3.18. ASCII diagrams showing architecture and parameters of MLP for multiclass classification scenario.

significant difficulty in accurately classifying QCD jets and PUs. Specifically, the models tend to misclassify QCD jets as PU and, opposite, misclassify PU as QCD.

Table 3.19. The results of four-class CNN and MLP classification.

		Precision	Recall	F1-score
CNN	PU	0.5949	0.7475	0.6625
	EM	0.9489	0.9600	0.9544
	QCD	0.6314	0.5425	0.5836
	PION	0.9034	0.6730	0.7713
	Accuracy			0.7390
	Macro avg	0.7696	0.7308	0.7430
	Weighted avg	0.7505	0.7390	0.7389
MLP	PU	0.5935	0.7380	0.6579
	EM	0.9554	0.9535	0.9545
	QCD	0.6247	0.5573	0.5891
	PION	0.9011	0.6605	0.7623
	Accuracy			0.7369
	Macro avg	0.7687	0.7273	0.7409
	Weighted avg	0.7498	0.7369	0.7379

Based on Figure 3.20, it is evident that there are only minor differences in the longitudinal profiles of the two classes, PU and QCD. These findings are of great importance, especially considering the fact that the longitudinal profile is a critical feature used in NN models. It suggests that these two classes cannot be distinguished without using additional features.



(a) For CNN *(b) For MLP*

Figure 3.19. Confusion matrix for four-class classification.

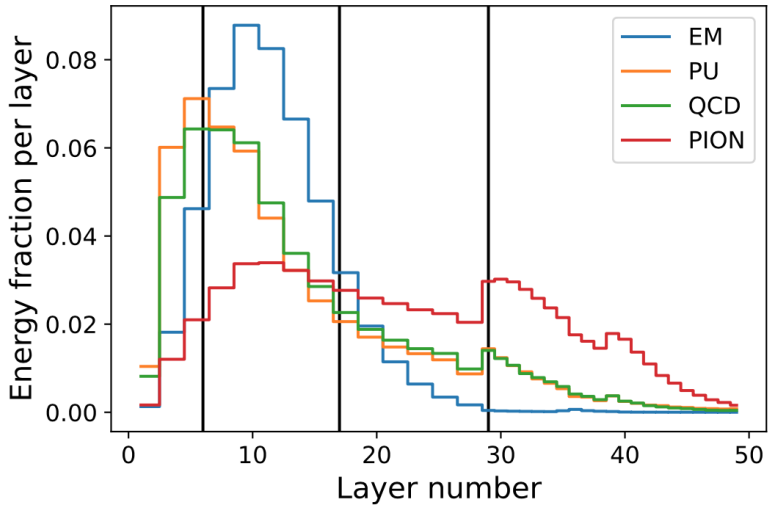


Figure 3.20. Average distribution of pt across HGCAL layers for EM, PU, QCD, and pions showers.

3.2.2 Four-class Classification with Additional Features

Given that PU/QCD misclassifications occur frequently, this indicates the need for model refinement to improve accuracy in distinguishing between jet types. As models are already optimized and the features used are actually HGCAL energy deposits, additional features have been created based on them. Several methods have been tried in order to achieve better performance, but only one significantly improved accuracy. The main idea is to compare the average longitudinal profile of each particle with the profile of every shower sample. That way, four new values are added to each shower sample, indicating the measure of similarity with each of the considered particles. By adding new features, the input data can no longer be viewed as an image. Therefore, in the CNN model, the convolutional layers will process an HGCAL image, and the additional features will be added immediately after flattening,

before a dense layer. Two different measures of similarity are calculated:

- **The Chi-Square Test**

  The chi-square test is a statistical technique used to measure the goodness of fit between observed and expected frequency distributions [194], calculated by

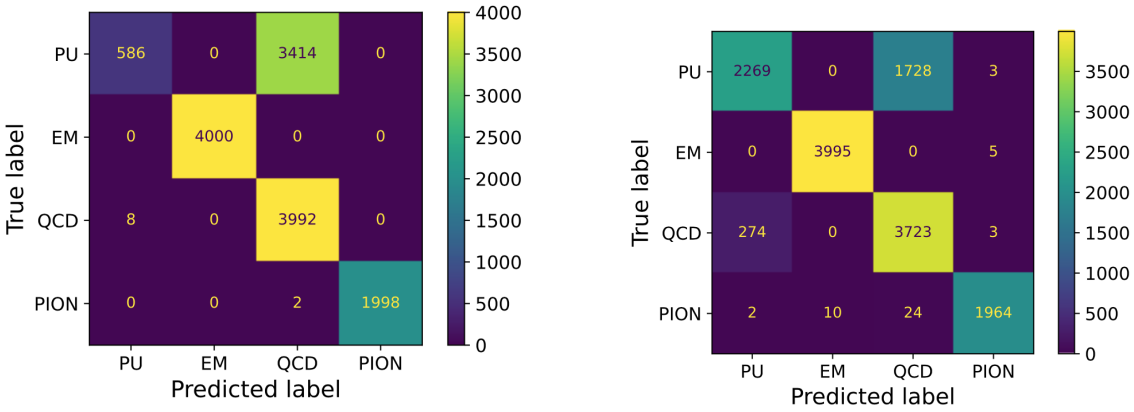
$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i} \tag{3.1}$$

  where O_i is the observed frequency in bin i, E_i is the expected frequency in the same bin i, and n is the number of bins. It is commonly used to compare two histograms and quantify the degree of similarity between them. If the chi-square statistic is low, it indicates a high level of similarity between the histograms. On the other hand, if the chi-square statistic is high, it suggests a significant difference in distribution. In ML, the chi-square test is an often-used method of feature selection (conditionally speaking, it is a dimensionality reduction method) [195], although here it is used to create additional features. The classification report for the previously created MLP model, now trained with additional features, is shown in Table 3.20 with the corresponding confusion matrix in Figure 3.21.

Table 3.20. The results of four-class CNN and MLP classification, with additional Chi-square-based calculated features.

		Precision	**Recall**	**F1-score**
CNN	PU	0.9865	0.1465	0.2551
	EM	1.0000	1.0000	1.0000
	QCD	0.5389	0.9980	0.6999
	PION	1.0000	0.9990	0.9995
	Accuracy			0.7554
	Macro avg	0.8814	0.7859	0.7386
	Weighted avg	0.8644	0.7554	0.7013
MLP	PU	0.8916	0.5673	0.6934
	EM	0.9975	0.9988	0.9981
	QCD	0.6800	0.9307	0.7859
	PION	0.9944	0.9820	0.9882
	Accuracy			0.8536
	Macro avg	0.8909	0.8697	0.8664
	Weighted avg	0.8761	0.8536	0.8490

According to results presented in Table 3.20 and Figure 3.21, there is an improvement in accuracy in more than 10% for MLP model. The ability to distinguish between



(a) For CNN *(b) For MLP*

Figure 3.21. Confusion matrix for four-class classification, with additional Chi-square-based calculated features.

PU and QCD jets has significantly improved. In the case of CNN, EM showers and PIONs are classified without errors, but most PUs are recognized as QCD. The reason is the high similarity between average PU and QCD longitudinal profile, which is more strongly emphasized when additional features are added later, when the number of parameters is significantly smaller

- **Chebyshev distance**

  Chebyshev distance [196] is also known as maximum value distance or $L\infty$ distance. The Chebyshev distance formula calculates the distance between two vectors p and q by finding the greatest difference between their corresponding components along any coordinate dimension:

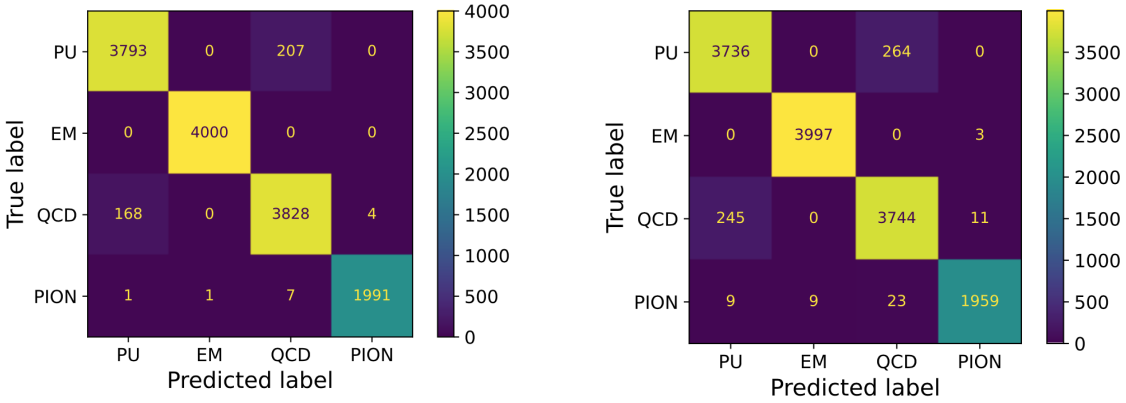
$$d(p,q) = \max_i(\left|p_i - q_i\right|). \tag{3.2}$$

  In addition to being used to determine the similarity between objects, it is also a useful metric for identifying outliers [197] and for data clustering [198, 199]. The classification report for the CNN and MLP models, with additional features calculated using this approach, shown in Table 3.21 with the corresponding confusion matrix in Figure 3.22 reveals an even higher improvement of accuracy, with $F1-score$ over 93% in all cases. Comparing with Chi-square-based results given in Table 3.20, it is clear that models in this case do not have problems recognizing QCD jets.

The results suggest that using both chi-square test and Chebyshev distance features improves the model's performance as compared to the baseline model. However, when the MLP model is trained using Chebyshev distance features, it exhibits significantly better results, $\sim$9% (CNN), and $\sim$96% (MLP), while chi-square model accuracy is $\sim$75% (CNN) and $\sim$85% (MLP). This is because the Chebyshev distance is more robust in capturing distributional differences, which allows the NN model to identify subtle variations between histograms,

Table 3.21. The results of four-class CNN and MLP classification, with additional Chebyshev-distance-based calculated features.

		Precision	Recall	F1-score
CNN	PU	0.9573	0.9483	0.9528
	EM	0.9998	1.0000	0.9999
	QCD	0.9471	0.9570	0.9520
	PION	0.9980	0.9955	0.9967
	Accuracy			0.9723
	Macro avg	0.9755	0.9752	0.9753
	Weighted avg	0.9723	0.9723	0.9723
MLP	PU	0.9363	0.9340	0.9352
	EM	0.9978	0.9992	0.9985
	QCD	0.9288	0.9360	0.9324
	PION	0.9929	0.9795	0.9862
	Accuracy			0.9597
	Macro avg	0.9640	0.9622	0.9631
	Weighted avg	0.9598	0.9597	0.9598



(a) For CNN *(b) For MLP*

Figure 3.22. Confusion matrix for four-class classification, with additional Chebyshev-distance-based calculated features.

leading to more precise predictions.

These findings underscore the efficacy of integrating histogram comparison features, particularly those derived from Chebyshev distance, into NN models for enhanced performance. By using these additional features, the NN model achieves superior classification accuracy and robustness, highlighting the importance of feature engineering in ML applications.

Additional tests are conducted with a quantized version of MLP using a dataset extended with the Chebyshev distance-based feature. 2-bit aggressive quantization is applied to the

MLP weights, biases, and activation functions, resulting in a significant drop in accuracy ($F1-score$ is $\sim$82%). If 4-bit quantization of the model is applied, the results are comparable to those obtained with the model in full precision ($F1-score$ is $\sim$95%).

3.2.3 Joined PU/QCD Background Approach (three-class Classification)

As neither PU nor QCD are among the particles of interest, they could be regarded as a single, mixed background for the next phase of the experiment. According to Figure 3.23, it can be observed that the distribution of classes in this case is even more unbalanced. Optuna

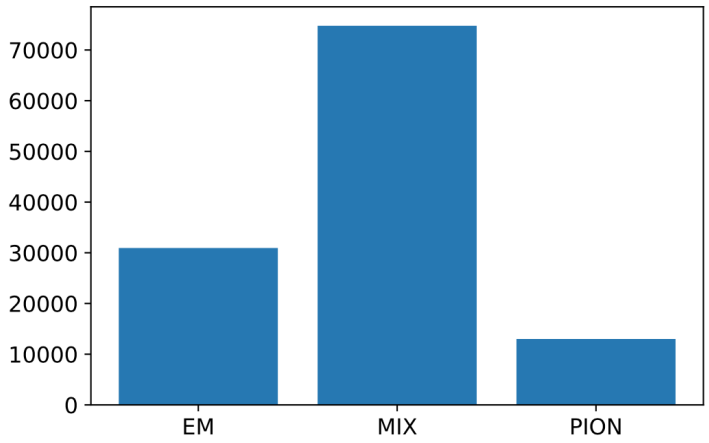


Figure 3.23. The distribution of sample numbers according to class, with PU and QCD joined in MIX background.

is launched on the same search space as before (Section 3.2.1) to determine the optimal CNN and MLP model configuration for a new dataset. The optimization results reveal that, for a three-class classification, both models will have a significant reduction in complexity. CNN model consists of a single convolutional layer with 17 kernels, followed by one dense layer with 28 nodes, as presented in Figure 3.24. Based on Optuna's recommendations, an MLP model contains three layers. The suggested number of nodes for each layer is 9, 32, and 21, respectively, as shown in Figure 3.25.

Classification reports providing detailed insights into the model's performance are generated after conducting tests on the developed models. These reports are presented in Table 3.22 with the corresponding confusion matrix in Figure 3.26.

Despite achieving better results than the baseline models, the performance of both NN models trained on the mixed background dataset is inferior to that of the models with integrated histogram comparison features, particularly those derived from Chebyshev distance. This indicates that integrating histogram comparison features offers valuable information for classification, resulting in improved model performance.

```
           OPERATION          DATA DIMENSIONS   WEIGHTS(N)   WEIGHTS(%)

(5, 5, 35)
               Input   #####      5    5   35
              Conv2D    \|/  --------------------      5372    55.0%
(3, 3, 17)
                relu   #####      3    3   17
             Flatten   ||||| --------------------         0     0.0%
(153,)
                       #####          153
               Dense   XXXXX --------------------      4312    44.1%
(28,)
                relu   #####           28
               Dense   XXXXX --------------------        87     0.9%
(3,)
             softmax   #####            3
```

Figure 3.24. ASCII diagrams showing architecture and parameters of CNN for joined multiclass classification scenario.

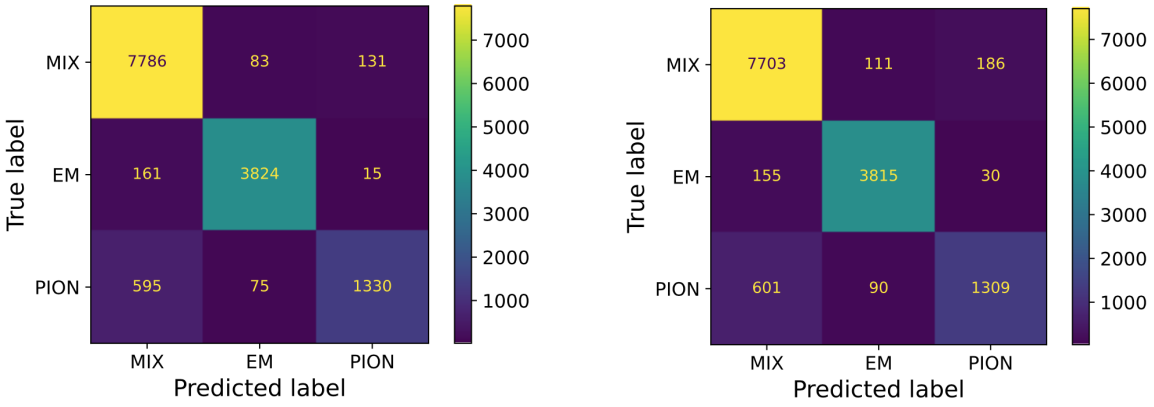
```
           OPERATION          DATA DIMENSIONS   WEIGHTS(N)   WEIGHTS(%)

(875,)
               Input   #####          875
               Dense   XXXXX --------------------      7884    88.0%
(9,)
                relu   #####            9
               Dense   XXXXX --------------------       320     3.6%
(32,)
                relu   #####           32
               Dense   XXXXX --------------------       693     7.7%
(21,)
                relu   #####           21
               Dense   XXXXX --------------------        66     0.7%
(3,)
             softmax   #####            3
```

Figure 3.25. ASCII diagrams showing architecture and parameters of MLP for joined multiclass classification scenario.

Table 3.22. The results of three-class CNN and MLP classification.

		Precision	Recall	F1-score
CNN	MIX	0.9115	0.9732	0.9414
	EM	0.9603	0.9560	0.9582
	PION	0.9011	0.6650	0.7652
	Accuracy			0.9243
	Macro avg	0.9243	0.8647	0.8883
	Weighted avg	0.9240	0.9243	0.9210
MLP	MIX	0.9106	0.9629	0.9360
	EM	0.9500	0.9537	0.9518
	PION	0.8584	0.6545	0.7427
	Accuracy			0.9162
	Macro avg	0.9063	0.8570	0.8769
	Weighted avg	0.9144	0.9162	0.9129



(a) For CNN *(b) For MLP*

Figure 3.26. Confusion matrix for three-class classification.

Therefore, supplementary tests with integrated additional features are performed for three-classes classification and binary classification problem EM vs. MIX. In this scenario, only a comparison with the average EM profile is utilized as an additional feature, resulting in a 98% accuracy in three-class classification, and even over 99% in EM vs. MIX binary classification problem.

3.3 Alternative images

The base NN classification models presented in Section 3.1.1 are tested with new datasets. First, results are given for the Consecutive Layers Approach.

After carefully comparing the classification metrics in Table 3.1 and Table 3.23, it is evident that in the EM vs. PU classification, the CNN model trained on the alternative dataset achieves results that are comparable to the model trained on the base dataset. However, the MLP results are slightly inferior in comparison. Furthermore, upon analyzing Table 3.1 and Table 3.23, as well as Table 3.2 and Table 3.24, one can come to the same conclusion for EM vs QCD and EM vs MIX background classification tasks. More specifically, the CNN model trained on the alternative dataset outperforms the model trained on the base dataset in EM vs. QCD classification, with a slightly better false negative rate. This observation suggests that the alternative dataset can be a more effective and efficient option for training CNN models for EM vs. QCD classification tasks. On the other hand, quantized versions of alternative models, as shown in Table 3.25, achieve slightly worse performance then quantized based versions.

Although the results of our alternative HGCAL image-trained models did not outperform those trained on traditionally created images, our findings indicate that it is crucial to investigate ways of improving the quality of HGCAL images in order to enhance classification accuracy.

Table 3.23. The results for the base CNN and MLP model for EM vs. different backgrounds (PU or QCD).

	Model	Acc	Prec	Rec	F1	AUC	FPR	FNR	Param #
PU	CNN	**0.9725**	**0.9881**	**0.9718**	**0.9798**	**0.9912**	**0.0259**	**0.0283**	4702
	MLP	0.9665	0.9842	0.9667	0.9754	0.9845	0.0341	0.0333	4486
QCD	CNN	**0.9622**	**0.9683**	0.9607	**0.9645**	**0.9881**	**0.0361**	0.0393	4904
	MLP	0.9512	0.9460	**0.9635**	0.9547	0.9828	0.0629	**0.0365**	3611

Table 3.24. The results for the base CNN and MLP model for EM vs. MIX.

Model	Acc	Prec	Rec	F1	AUC	FPR	FNR	Param #
CNN	**0.9665**	**0.9673**	**0.9673**	**0.9673**	**0.9854**	**0.0343**	**0.0328**	2753
MLP	0.9608	0.9650	0.9583	0.9616	0.9854	0.0364	0.0418	7153

Table 3.25. The results for the QCNN and QMLP model for EM vs. MIX.

Model	Acc	Prec	Rec	F1	AUC	FPR	FNR	Param #
QCNN	**0.9614**	**0.9741**	0.9497	**0.9618**	0.9794	**0.0265**	0.0503	2753
QMLP	0.9594	0.9644	**0.9560**	0.9602	**0.9805**	0.0370	**0.0440**	7153

The same tests are conducted with a Principal Component Analysis Approach dataset. Upon detailed examination of the classification metrics in Table 3.1, Table 3.23, and Table 3.26, it is evident that the PCA approach does not outperform either the model trained on the base dataset or the consecutive layers based model. Furthermore, comparing Table 3.2, Table 3.24, and Table 3.27, specifically the EM vs MIX background classification task, it can be assumed that the PCA-based model slightly outperforms the model trained on the consecutive layers-based dataset, but not the base model. The same conclusion is applicable to quantized versions of models (by comparing Table 3.5, Table 3.25, and Table 3.28), with the quantized version of the PCA based approach having the lowest FNR.

Table 3.26. The results for the base CNN and MLP model for EM vs. different backgrounds (PU or QCD).

	Model	Acc	Prec	Rec	F1	AUC	FPR	FNR	Param #
PU	CNN	**0.9717**	**0.9881**	**0.9730**	**0.9805**	**0.9891**	**0.0320**	**0.0270**	4702
	MLP	0.9634	0.9808	0.9690	0.9748	0.9851	0.0518	0.0310	4486
QCD	CNN	**0.9610**	**0.9757**	0.9630	**0.9693**	**0.9889**	**0.0424**	0.0370	4904
	MLP	0.9590	0.9706	**0.9650**	0.9678	0.9835	0.0517	**0.0350**	3611

It is important to note that tests were conducted using a smaller number of generated images for both alternative approaches. As no improvements were observed, further tests

Table 3.27. The results for the base CNN and MLP model for EM vs. MIX.

Model	Acc	Prec	Rec	F1	AUC	FPR	FNR	Param #
CNN	**0.9667**	**0.9718**	**0.9657**	**0.9688**	**0.9883**	**0.0323**	**0.0343**	2753
MLP	0.9641	0.9693	0.9635	0.9664	0.9841	0.0352	0.0365	7153

Table 3.28. The results for the QCNN and QMLP model for EM vs. MIX.

Model	Acc	Prec	Rec	F1	AUC	FPR	FNR	Param #
QCNN	**0.9689**	**0.9772**	**0.9645**	**0.9708**	**0.9855**	**0.0260**	**0.0355**	2753
QMLP	0.9614	0.9694	0.9583	0.9638	0.9820	0.0349	0.0418	7153

were not carried out.

The current quality of HGCAL images may be a factor that limits the performance of models trained on them. Therefore, to enhance L1T functionality, we recommend further research to identify techniques that can be used to improve the quality of HGCAL images, which could lead to more accurate classification results in the future.

3.4 FPGA Implementation

In previous sections, a comprehensive set of tests using different types of data refinement techniques used for CNN and MLP models was presented. Based on the result of the examination of their performance, the current focus is on choosing a subset of the most successful models to be deployed on a dedicated L1T device, FPGA Xilinx Virtex UltraScale, part number xcvu13p-fhgb2104-2-e using *hls4ml*.

3.4.1 FPGA - Basic Concepts

FPGAs are a type of integrated circuits that provide programmers the ability to define and configure digital logic functions within the chip itself according to their specific use case requirements. This makes them extremely flexible and applicable to different applications, such as digital signal processing, machine learning, and many others.

FPGAs contain different building blocks (Figure 3.27) that are connected together based on the specific requirements [28]:

- Logic cells and LUTs, are used to perform arbitrary functions on small bit-width inputs. They can be used for boolean operations, arithmetic, and small memories.
- Flip-Flops (FFs), used to register data in time with the clock pulse.
- Digital Signal Processors (DSPs) are specialized units for multiplication and arithmetic. Compared with LUTs, they are more efficient for these types of operations. In

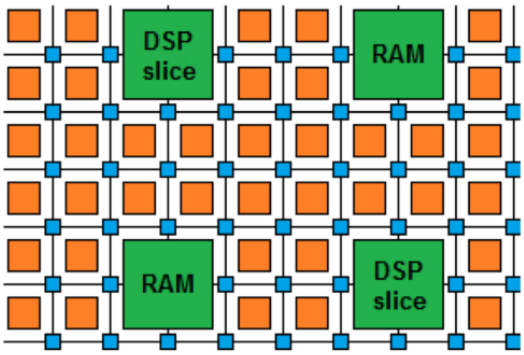


Figure 3.27. Diagram showing main FPGA building blocks [28].

NN implementation, DSPs are often limiting resource. It is important to keep in mind that the number of DSPs needed to perform the multiplication depends on the data precision used, so it is possible to influence the required number of DSPs by configuring the precision.

- BRAMs, small and fast memories (RAMs, ROMs, FIFOs). Memories using BRAMs are generally considered more efficient than using LUTs.

In addition, FPGAs use specialized blocks for I/O, which makes them popular in embedded systems and HEP triggers.

3.4.2 hls4ml

The L1T algorithms, including those suggested in this thesis, should be designed to run on FPGA devices. However, FPGAs are usually programmed using Hardware Description Languages (HDLs) such as VHSIC Hardware Description Language (VHDL) or Verilog (usually used for FPGAs), making it difficult to develop ML models. Therefore, scientists have developed HLS, a technology that takes high-level code in languages like C and C++ and automatically synthesizes it into RTL (Register Transfer Level) descriptions written in HDLs such as Verilog or VHDL. HLS aims to make hardware design more accessible to software engineers by raising its abstraction level. *hls4ml* [29, 30] is a Python library that efficiently and quickly translates ML models to HLS code. The library was originally created to implement relatively small NNs in FPGAs for the L1T, but the scope of focus has significantly expanded since then. Figure 3.28 illustrates a typical workflow needed to develop a NN model and implement it in FPGA. The red part of shema indicates the standard workflow required to develop an NN model for a specific purpose, usually including tools/libraries such as Keras and PyTorch. The blue section describes the *hls4ml* workflow, which starts with a model translation into an *hls* project that is then synthesized and implemented to run on an FPGA.

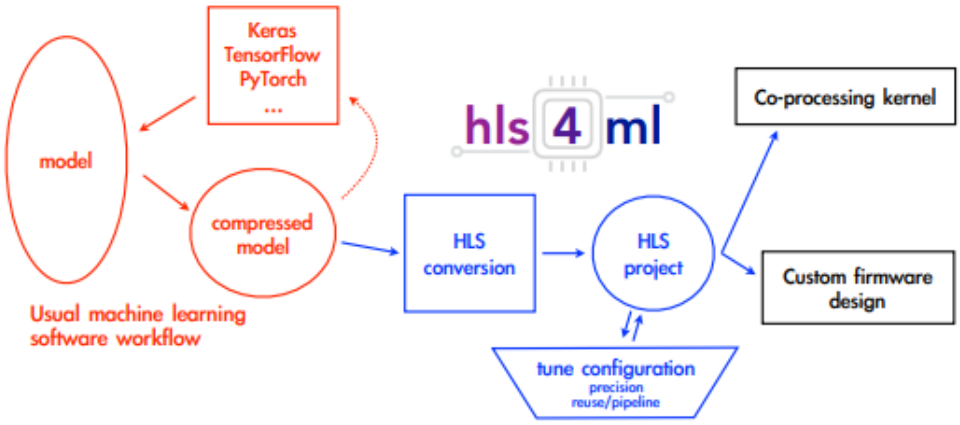


Figure 3.28. A schema representing the typical workflow for converting a model into an FPGA implementation using hls4ml [30].

Figure 3.29 visualize how FPGA uses resources in neural network inference: activation function values are precomputed and stored in BRAMs, multiplication is calculated using DSPs, while the bias values are stored in logic cells.

$$\mathbf{x}_n = g_n(\mathbf{W}_{n,n-1}\mathbf{x}_{n-1} + \mathbf{b}_n)$$

precomputed and stored in BRAMs — DSPs — logic cells

Figure 3.29. Visualization of FPGA building blocks tasks in NN inference [28].

The way in which algorithms are processed on FPGAs differs from their execution on traditional CPUs. FPGAs have the capability to carry out multiple independent operations simultaneously, allowing them to achieve processing speeds of trillions of operations per second. This is accomplished at a low power cost compared to CPUs and GPUs. However, these operations use dedicated resources on the FPGA and cannot be changed while running. To develop an efficient FPGA implementation, a crucial aspect is to effectively manage the utilization of FPGA resources while simultaneously meeting the latency and throughput objectives of the intended algorithm. Some key metrics to consider for an FPGA implementation are:

- **latency**, refers to the total time required for a single iteration of the algorithm to complete, measured in units of "clocks."
- **initiation interval**, the number of clock cycles required before the algorithm can accept a new input, is often expressed as "II" and is inversely proportional to the inference rate or throughput.

- **resource usage**, it is common to express this usage in terms of several categories: BRAMs, DSPs, FFs, and LUTs.

The *hls4ml* offers several configurable parameters allowing users to customize their applications' latency and resource usage. One of the ways to save resources and reduce latency is by decreasing precision. For example, for many applications, the limiting FPGA resource is the number of DSPs that are primarily used for multiplication. Different precision settings use different numbers of DSPs needed for performing multiplication. Another example is the usage of the "reuse factor" parameter, which enables parallelization and resource reuse, determining the number of times each multiplier is used to compute a layer of neuron's values. It is evident that a low reuse factor results in the lowest latency and the highest throughput but consumes the most resources. On the other hand, a high reuse factor saves resources while leading to longer latency and lower throughput.

3.4.3 Neural Network Model Selection

The primary goal of the selection is to choose the most suitable models that, in addition to good performance metrics, offer the possibility of efficient hardware resource usage due to their characteristics. Therefore, the selection process is performed in a way that each subsequent model implemented in the FPGA enables a noticeable improvement in its performance or savings in resource consumption compared to its predecessor. This can be achieved through various techniques, such as optimizing the models by using model and/or data quantization combined with capping and utilizing the *hls4ml* features like adjustment of precision and reuse factor. The goal is to choose previously created models that not only meet the performance requirements but also use resources sparingly, bearing in mind that FPGAs are shared among a number of L1T algorithms. The process of achieving this objective involves rigorous testing and analysis, which helps in identifying the minimum resource consumption without impacting model accuracy.

Consequently, the models shown in Table 3.29 are selected for *hls4ml* implementation. We start with a base CNN model that will provide the initial estimation of resource consumption. The next one is QCNN, CNN with weights, biases, and activations quantized on 2 bits, which helps in decreasing computational resources. After that follows a model named QCNN_U2, which, in addition to the 2-bit quantized model, also applies 2-bit quantization to inputs, which enables additional saving of resources.

According to the *hls4ml* documentation, each layer in the model should have a maximum of 4096 trainable parameters to achieve the lowest possible latency. This limitation is due to fixed constraints in the Vivado compiler. If the model stays within this limit, the strategy = 'latency' can be used, which will result in lower latency compared to using strategy = 'resource'. Given the above, the MLP models using the complete HGCAL image are not possible to implement with a 'latency' strategy.

Table 3.29. The NN models for EM vs. MIX. classification selected for FPGA implementation.

Model	Acc	Prec	Rec	F1	AUC	FPR	FNR	Param #
CNN	0.9701	0.9688	0.9715	0.9702	0.9919	0.0312	0.0285	2753
QCNN	0.9722	0.9807	0.9635	0.9720	0.9856	0.0190	0.0365	2753
QCNN_U2	0.9701	0.9736	0.9665	0.9700	0.9832	0.0262	0.0335	2753

3.4.4 Implementation Results

The classification results obtained after the implementation of selected models, together with the used precision settings, are shown in Table 3.30. In addition to standard NN (Keras) accuracy, the *hls4ml* report also gives the *hls* accuracy of implemented models. Accuracy of *hls* implementation - *hls* accuracy is affected not only by the original NN design but also by the quality of the optimization settings applied during the synthesis process. hls4ml offers a number of implementation improvement options that use different parameter settings. Some of them used in the experiment are:

- **Reuse factor**: this parameter defines the number of times that each multiplexer will be used. It is clear that in fully parallel implementation, each multiplexer is used just once, which requires more resources. Therefore, depending on NN size, due to the limited resources it is often not possible to run the model in fully parallel implementation. It is also clear that the requirement for resources is inversely proportional to latency.
- **Precision**: precision in *hls4ml* refers to the numerical representation of used data; it defines the precision of inputs, outputs, weights, and biases. It is denoted by $< X, Y >$, where Y is the number of bits representing the integer part, and X is the total number of bits. By choosing a low precision for a model, FPGA resource usage can be reduced. However, it can lead to decreased model performance if not selected appropriately.

Table 3.30. The Keras and hls accuracy for selected models.

Model	Keras Acc	*hls* Acc	Extended precision	New *hls* Acc
CNN	0.970125	0.686	<28,10>	0.97
QCNN	0.97325	0.874125	<22,8>	0.973
QCNN_U2	0.970125	0.829375	<18,8>	0.969625

Considering the fact that default *hls4ml* precision is <16,6>, it is obvious that for all models, it was necessary to increase the required accuracy, which results in an increase in resource consumption. However, with the non-quantized CNN model, this increase was even

up to <28,10>, which indicates that more resources are required to maintain a higher level of precision.

3.4.5 Estimation of FPGA Resource Usage

Table 3.31 records the FPGA implementation results for each of the selected models, estimating the utilization of logic resources provided by the HLS tool after synthesis, using the default clock period of 5 ns. In the first phase of the experiment, we evaluated the precision of the models at the model level. This was done by ensuring that the HLS accuracy matched or was comparable to the Keras accuracy. We did not optimize the models by adjusting the precision at the layer level. Instead, we ensured that the overall precision was sufficient for our purposes without worrying about optimizing resource consumption.

Table 3.31. The results of FPGA implementation for selected models model with precision determined on model level. The table also includes the total number of DSPs, FFs, and LUTs, allowing for a direct comparison with the utilized values.

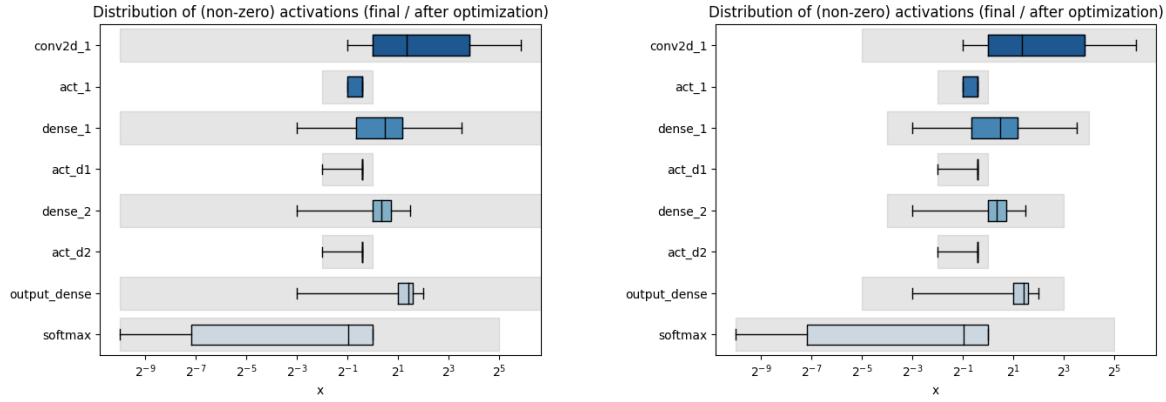
Model	**Time**	**Interval**	**Latency**	**DSP**	**FF**	**LUT**
	[ns]	**[Cycles]**	**[μs]**	**# (%)**	**# (%)**	**# (%)**
				12 288	**3 456 000**	**1 728 000**
CNN	4.374	34	0.170	5436 (∼44.24)	91 779 (∼2.65)	206 234 (∼11.93)
QCNN	4.350	34	0.170	2 (∼0.02)	20 573 (∼0.6)	72 358 (∼4.19)
QCNN_U2	4.357	34	0.170	2 (∼0.02)	20 187 (∼0.58)	66 940 (∼3.87)

The results shown in Table 3.31 demonstrate two important findings. First, with a latency time of 0.170 μs, all models satisfy the required latency constraint ($< 1\mu s$). Secondly, despite the fact the latency is the same, the resource usage metrics show huge differences when comparing the results of quantized CNN with a base CNN mode. As a result of model quantization, apart from the fact that less precision is required to achieve Keras accuracy, the number of DSPs is rapidly reduced, from 5436 to 2. A significant reduction was also achieved with the number of FFs, from 91 779 or 2% to $\sim$ 0%. LUT utilization is from $\sim$11% decreased to $\sim$ 3% for quantized models. Although the results shown do not differ significantly if we compare the quantized models, any reduction in resource usage is significant given the large number of algorithms simultaneously executing on FPGAs. Also, the use of quantized data enables a further reduction in precision, which is tested in the next phase of the experiment.

The *hls4ml* offers profiling tools to help decide about the appropriate model precision. The profiling tool outputs two graphs, "before optimization" and "final/after optimization," one for each of the weights/basis layers group and one for the activation function layers group. The "before optimization" version of the plots describes the distributions of the original Keras/QKeras weights/activations, while the "after optimization" plots show the distribu-

tions of the ModelGraph, which represents the internal execution graph of NN that is being processed by *hls4ml*. Graph type is a box and whisker chart, a convenient way of graphical representation of the distribution of numerical data through their quartiles, and is here applied for each layer of the implemented model. As usual, each box shows the median and quartiles of the distribution. Additionally, the grey shaded boxes show the range, which can be represented with the precision defined by *hls4ml* configuration settings. Figure 3.30a shows "final/after optimization" graphs for the QCNN_U2 model with the precision needed so that the achieved HLS accuracy is comparable to the Keras accuracy. It is visible that grey-shaded boxes are much wider than whiskers, indicating the possibility of decreasing precision for those particular layers. To demonstrate the effect of input data quantization together with the possibility of per-layer optimization, the model QCNN_U2 is implemented in two different ways: one optimizing just the hidden layers' precision and another that also optimizes the input layer precision settings.

Hence, the first step is to optimize the inner layers' precision. The *hls* project is rebuilt with new configuration settings chosen carefully to decrease layer precision maintaining accuracy. Figure 3.30b represents profiling graphs for the updated configuration. It's worth noting that the precision settings for each layer, in this case, fit within the gray-shaded box; therefore, no resources are unnecessarily reserved.



(a) For model with precision optimized on model level.

(b) For model with optimized inner layers precision.

Figure 3.30. Activations part of "final/after optimization" graphs for QCNN_UNIQ2. The box and whisker chart represent the distribution of the values required for the keras model, while the grey shaded boxes show the range which can be represented with the precision defined by hls4ml configuration settings.

In the last step, we also reduce the precision setting for the input layer. The resource consumption results for all three versions of QCNN_U2 model implementations are given in Table 3.32.

The results clearly show that experiments with optimized input and hidden layers overperform standard approaches in terms of resource usage. A QCNN model with quantized in-

Table 3.32. The resource consumption for different levels of optimizations for the QCNN_U2 model.

Settings	FF #	FF %	LUT #	LUT %
default	20187	0.584	66940	3.874
hidden layer precision adjusted	19872	0.575	67179	3.888
input + hidden layer precision adjusted	12963	0.375	62809	3.634

puts has enabled optimization of precision in the input layer that leads to savings of 35.76% in a number of FFs and 6.17% in the number of LUTs compared with base QCNN with precision defined on the model level.

4 CONCLUSIONS AND FUTURE WORK

This thesis presents an NN model for classifying EM showers from the mixed background made of PU and QCD jets at CMS L1T within a given time limit of a few μs. The suggested NNs are intended to run on targeted L1T hardware, FPGAs. FPGAs are able to achieve high speed due to a combination of factors, with parallelism and customization being the most prominent. These factors result in highly optimized and efficient implementations that can outperform general-purpose processors for tasks that require high speed, like data selection in L1T. In recent times, advancements, among which are QKeras and *hls4ml*, have enabled the customization of NN models to run on FPGAs.

Even if it does not represent perfectly the realistic situation, the first set of HGCAL images was created based on simulated data containing ideal signal (EM) images without PU. The initial tests were conducted using a dataset that included images of ideal electromagnetic signals. The neural network classifiers, which differentiate between ideal EM showers and PU, achieved a remarkable 99% accuracy. The next set of generated data contained an EM shower that included PU as a signal and two different kinds of backgrounds: PU and QCD jets, both with the same PU value of 200. Although the generated dataset is closer to a more realistic situation, it was noticed that there is a big difference in average EM and PU cluster *pt*, and somewhat smaller compared to QCD. Hence, one of the baseline methods for comparison is cut-off, an event selection method often used in HEP for particle identification. Additional methods for comparison include Support Vector Machine and Random Forest. Three classification cases are considered depending on background: EM vs. PU, EM vs. QCD, and finally EM vs. mixed background.

Throughout the research, special attention is given to the development of neural network models that could satisfy limited L1T resource and timing conditions. By using the hyper-parameter optimization framework Optuna, two models are developed: convolutional neural network and multi-layer perceptron models with signal/background classification accuracy between 96 and 98% and FNR between 2 and 4%, depending on the background case. The achieved results could eventually be improved by adding physical features to input data, but it would be difficult to incorporate into L1T latency requirements.

The following sections of the paper discuss the problem of EM vs. mixed background in one-vs.-all classification. Comprehensive research is provided to develop models with as little resource consumption as possible while preserving the achieved accuracy and satisfying

the required latency. In-depth model analysis was also conducted to understand the potential limitations of the models. Both NNs struggle to accurately predict outcomes in two scenarios. One is for low-energy EM showers that start showering earlier, and the second is in the case when EM hadronic deposits are higher than usual. The use of low-level features, such as HGCAL images and energy deposits, has its limitations when dealing with these specific scenarios. To address the limitations caused by using low-level features represented in the form of HGCAL images, future efforts could possibly include additional high-level features or explore more advanced model architectures that can capture the nuances of the data more effectively.

The interpretability of the presented NN models is examined using the SHAP framework. According to SHAP results, the features that have the greatest impact on the prediction are mostly between layers 5 to 10, corresponding to layers where the peak energy deposition occurs for both PU and QCD. Eventually, these findings could be used to enhance the NN models with an attention mechanism. It would enable the models to selectively focus on relevant information, potentially leading to improved performance.

The next step in model optimization was model(s) quantization on 2 bits using the QKeras quantization library, which reduces computational and memory costs with minimal accuracy degradation. In order to reduce model requirements additionally, different techniques for data refinement are presented. The results have shown that methods that affect the range of input data, like capping, can significantly lower FPR, which results in a significant reduction in the amount of unnecessarily stored data. Suggested image channel reduction methods have resulted in a slight drop in accuracy, with a lower number of NN parameters. Although quantization is a lossy process that can result in degradation of accuracy, it has been shown that both data quantization approaches, uniform and nonuniform quantization, achieve performances comparable to a base model while requiring significantly less precision for data storage.

Furthermore, this thesis considers a multi-class classification problem with four classes: EM shower, PU, QCD jets, and pions. Two scenarios can improve the low performance of developed NN models based just on raw detector data. The first one involves joining PU and QCD samples in a unique class. The second approach, based on additional features that measure the similarity between the longitudinal profile of each particle using Chebyshev distance, overperforms previous attempts, achieving 96% accuracy.

In the previously presented approaches, the goal was to improve the efficiency and reduce the resource consumption of the developed models by quantizing the model and applying various refinement methods to the data. As the quality of the data is crucial for the accuracy and effectiveness of the machine learning methods, the last conducted tests are aimed at improving the quality of the database itself by presenting an alternative way of generating HGCAL images. The model performances are comparable with those based on databases generated by using standard techniques. Even though the suggested approach has its own

limitations (for some shower shapes), the observation suggests that research into alternative approaches to HGCAL image generation should continue.

As intended, the presented data refinement methods have enabled a further reduction in the model resource consumption, which is confirmed in the final step of the experiment. Using the *hls4ml* library, selected models are converted into the firmware to be implemented on FPGA. The chosen models depending on the applied reduction approach, enable different levels of optimization. All implemented models can be executed within a set latency. Model quantization resulted in just 2 digital signal processors used and flip-flop and lookup table utilization under 1% and 4%, respectively. It has been demonstrated that employing variations of the proposed quantized convolutional neural network can result in additional resource savings when implementing the model in FPGA. Specifically, compared with a base model, the use of model and data quantization can reduce the number of flip-flops by 35.78% and the number of lookup tables by 6.17%.

In order to improve the accuracy of trigger algorithms based on HGCAL data, future efforts will be aimed toward introducing new features, including high-level ones, or exploring advanced model architectures that can better capture the subtleties of the data. Another potential research direction is to investigate an effective method for exploiting the sparsity of HGCAL images.

BIBLIOGRAPHY

[1] G. Apollinari, B. A. I., O. Brüning, P. Fessia, M. Lamont, L. Rossi and L. Tavian, editors, *High-Luminosity Large Hadron Collider (HL-LHC): Technical design report*, CERN Yellow Reports: Monographs, CERN, Geneva, 2020.

[2] CMS Collaboration, The CMS experiment at the CERN LHC, *Journal of Instrumentation*, 3, 08, S08004, 2008.

[3] P. Mouche, Overall view of the LHC. Vue d'ensemble du LHC, https://cds.cern.ch/record/1708847?ln=hr, [Accessed 2023-06-15].

[4] The Nobel Prize in Physics 2013, https://www.nobelprize.org/prizes/physics/2013/summary/, [Accessed June 26th 2024.].

[5] I. Alonso, O. Brüning, P. Fessia, M. Lamont, L. Rossi, L. Tavian, M. Zerlauth and K. Marinov, High-luminosity large hadron collider (HL-LHC): Technical design report, 2023.

[6] W. Herr and B. Muratori, Concept of luminosity, *CERN Accelerator School and DESY Zeuthen: Accelerator Physics*, 361–377, 2003.

[7] LS3 schedule change, https://hilumilhc.web.cern.ch/article/ls3-schedule-change, [Accessed June 26th 2024.].

[8] J. Mnich, The Future of Particle Physics CERN view, https://indico.cern.ch/event/877521/contributions/4954197/attachments/2520448/4333991/JM%20Split%20Oct2022-vs3.pdf, 2022, [Accessed 30-06-2024].

[9] K. Albertsson et al., Machine learning in high energy physics community white paper, *Journal of Physics: Conference Series*, 1085, 2018.

[10] H. Yang, B. Roe and J. Zhu, Studies of stability and robustness for artificial neural networks and boosted decision trees, *Nuclear Instruments and Methods in Physics Research Section A Accelerators Spectrometers Detectors and Associated Equipment*, 574, 2006.

[11] B. Roe, H. Yang, J. Zhu, Y. Liu, I. Stancu and G. McGregor, Boosted decision trees as an alternative to artificial neural networks for particle identification, *Nuclear Instruments and Methods in Physics Research A*, 543, 2004.

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

[13] P. Baldi, P. Sadowski and D. Whiteson, Searching for exotic particles in high-energy physics with deep learning, *Nature communications*, 5, 4308, 2014.

[14] J. Cogan, M. Kagan, E. Strauss and A. Schwartzman, Jet-images: Computer vision inspired techniques for jet tagging, *Journal of High Energy Physics*, 2015, 2014.

[15] C. F. Madrazo, I. H. Cacha, L. L. Iglesias and J. M. de Lucas, Application of a convolutional neural network for image classification to the analysis of collisions in high energy physics, *CoRR*, abs/1708.07034, 2017.

[16] M. Andrews, M. Paulini, S. Gleyzer and B. Poczos, Exploring end-to-end deep learning applications for event classification at CMS, *EPJ Web of Conferences*, 214, 06031, 01 2019.

[17] A. Aurisano, A. Radovic, D. Rocco, A. Himmel, M. Messier, E. Niner, G. Pawloski, F. Psihas, A. Sousa and P. Vahle, A convolutional neural network neutrino event classifier, *Journal of Instrumentation*, 11, 04 2016.

[18] A. D. Florio, F. Pantaleo, A. Carta and on behalf of the CMS collaboration, Convolutional neural network for track seed filtering at the CMS high-level trigger, *Journal of Physics: Conference Series*, 1085, 4, 042040, 2018.

[19] M.-H. Shahid, Anomaly Detection for CMS L1 trigger at HL-LHC, https://cds.cern.ch/record/2790202?ln=hr, 2021.

[20] S. R. Qasim, N. Chernyavskaya, J. Kieseler, K. Long, O. Viazlo, M. Pierini and R. Nawaz, End-to-end multi-particle reconstruction in high occupancy imaging calorimeters with graph neural networks, *The European Physical Journal C*, 82, 8, 2022.

[21] A. Butter, T. Plehn and R. Winterhalder, How to GAN LHC events, *SciPost Physics*, 7, 2019.

[22] R. Sipio, M. Giannelli, S. Haghighat and S. Palazzo, A generative-adversarial network approach for the simulation of QCD dijet events at the LHC, *The Journal of High Energy Physics*, 050, 2019.

[23] E. Shumka, A. Samalan, M. Tytgat, M. el Sawy, G. Alves, F. Marujo, E. Coelho, E. M. Costa, H. Nogima, A. Santoro, S. Fonseca de Souza, D. de Jesus Damião, M. Thiel, K. Amarilo, M. Filho, A. Aleksandrov, R. Hadjiiska, P. Iaydjiev, M. Rodozov and F. Araujo, Machine learning based tool for CMS RPC currents quality monitoring, *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 1054, 168449, 2023.

[24] B. Denby, M. Campbell, F. Bedeschi, N. Chriss, C. Bowers and F. Nesti, Neural networks for triggering, *Nuclear Science, IEEE Transactions on*, 37, 248 – 254, 05 1990.

[25] M. Courbariaux, Y. Bengio and J. David, Binaryconnect: Training deep neural networks with binary weights during propagations, *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, 3123–3131, MIT Press, Cambridge, MA, USA, 2015.

[26] M. Rastegari, V. Ordonez, J. Redmon and A. Farhadi, XNOR-net: ImageNet classification using binary convolutional neural networks, 10 2016.

[27] T. Boser, J. Nielsen, I. J. Johnson and P. Calafiura, CNNs on FPGAs for Track Reconstruction, https://dca.ue.ucsc.edu/system/files/dca/1091/1091.pdf, 2018.

[28] D. Diaz, M. Quinnan, R. Kansal, E. E. Khoda and J. Duarte, hls4ml tutorial, https://indico.cern.ch/event/1176254/contributions/4940415/attachments/2483939/4264585/snowmass%20hls4ml%20tutorial.pdf, [Accessed 2024-01-15].

[29] FastML Team, fastmachinelearning/hls4ml, 2023.

[30] J. Duarte, S. Han, P. Harris, S. Jindariani, E. Kreinar, B. Kreis, J. Ngadiuba, M. Pierini, N. Tran and Z. Wu, Fast inference of deep neural networks in FPGAs for particle physics, *Journal of Instrumentation*, 13, 2018.

[31] C. N. Coelho, A. Kuusela, S. Li, H. Zhuang, T. Aarrestad, V. Loncar, J. Ngadiuba, M. Pierini, A. Pol and S. Summers, Automatic heterogeneous quantization of deep neural networks for low-latency inference on the edge for particle detectors, *Nature Machine Intelligence*, 675–686, 2021.

[32] J. Ngadiuba, V. Loncar, M. Pierini, S. Summers, G. Di Guglielmo, J. Duarte, P. Harris, D. Rankin, S. Jindariani, M. Liu, K. Pedro, N. Tran, E. Kreinar, S. Sagear, Z. Wu and D. Hoang, Compressing deep neural networks on FPGAs to binary and ternary precision with hls4ml, *Machine Learning: Science and Technology*, 2, 1, 015001, 2020.

[33] T. Aarrestad et al., Fast convolutional neural networks on FPGAs with hls4ml, *Mach. Learn. Sci. Tech.*, 2, 4, 045015, 2021.

[34] A. Elabd, V. Razavimaleki, S.-Y. Huang, J. Duarte, M. Atkinson, G. DeZoort, P. Elmer, S. Hauck, J.-X. Hu, S.-C. Hsu, B.-C. Lai, M. Neubauer, I. Ojalvo, S. Thais and M. Trahms, Graph neural networks for charged particle tracking on FPGAs, *Frontiers in Big Data*, 5, 828666, 03 2022.

[35] Y.-j. Jwa, G. Di Guglielmo, L. Arnold, L. Carloni and G. Karagiorgi, Real-time inference with 2D convolutional neural networks on field programmable gate arrays for high-rate particle imaging detectors, *Frontiers in Artificial Intelligence*, 5, 855184, 05 2022.

[36] E. Ferro, Implementation of Deep Neural Networks for the Level 1 Trigger system of the future High-Granularity Calorimeter (HGCAL), https://webthesis.biblio.polito.it/15937/, [Accessed December 04th 2023.].

[37] D. Barney, CMS Detector Slice, https://cds.cern.ch/record/2120661, 2016, [Accessed 2023-08-15].

[38] S. Chatrchyan et al., Observation of a New Boson at a Mass of 125 GeV with the CMS Experiment at the LHC, *Phys. Lett. B*, 716, 30–61, 2012.

[39] T. Sakuma, Cutaway diagrams of CMS detector, https://cds.cern.ch/record/2665537, 2019, [Accessed 2024-01-10].

[40] I. Neutelings, CMS wiki pages, how to draw diagrams in latex with TikZ., [Accessed 2024-01-15].

[41] V. Veszpremi, Operation and performance of the CMS tracker, *Journal of Instrumentation*, 9, 03, C03005, mar 2014.

[42] S. Chatrchyan et al., The CMS Experiment at the CERN LHC, *JINST*, 3, S08004, 2008.

[43] G. Organtini, Avalanche photodiodes for the CMS electromagnetic calorimeter, *1998 IEEE Nuclear Science Symposium Conference Record.*, 1, 189–194 vol.1, 1998.

[44] S. Chatrchyan, E. Anttila, S. Czellar, J. Härkönen, A. Heikkinen, V. Karimäki, R. Kinnunen, J. Klem, M. Kortelainen, T. Lampén, K. Lassila-Perini, S. Lehti, T. Linden, P. Luukka, T. Mäenpää, J. Nysten, E. Tuominen, J. Tuominiemi, D. Ungaro and R. Romaniuk, Performance of CMS hadron calorimeter timing and synchronization using test beam, cosmic ray, and LHC beam data, *Journal of Instrumentation*, 5, T03013, 03 2010.

[45] J. G. Layter, *The CMS muon project: Technical Design Report*, Technical design report. CMS, CERN, Geneva, 1997.

[46] V. Khachatryan, A. Sirunyan, A. Tumasyan, W. Adam, E. Asilar, T. Bergauer, J. Brandstetter, E. Brondolin, M. Dragicevic, J. Erö, M. Flechl, M. Friedl, R. Frühwirth, V. Ghete, C. Hartl, N. Hörmann, J. Hrubec, M. Jeitler, V. Knünz, A. König, M. Krammer, I. Krätschmer, D. Liko, T. Matsushita, I. Mikulec, D. Rabady, B. Rahbaran, H. Rohringer, J. Schieck, R. Schöfbeck, J. Strauss, W. Treberer-Treberspurg, W. Waltenberger, C.-E. Wulz, V. Mossolov, N. Shumeiko, J. S. Gonzalez, S. Alderweireldt, T. Cornelis and E.A., The CMS trigger system, *Journal of Instrumentation*, 12, 01, P01020, 2017.

[47] CMS Colaboration, The phase-2 upgrade of the CMS Level-1 Trigger, Tech. rep., CERN, Geneva, 2020.

[48] E. Govorkova, E. Puljak, T. Aarrestad, M. Pierini, K. A. Woźniak and J. Ngadiuba, LHC physics dataset for unsupervised new physics detection at 40 MHz, *Scientific Data*, 9, 03 2022.

[49] W. Adam et al., The CMS high level trigger, *Eur. Phys. J. C*, 46, 605–667, 2006.

[50] CMS Collaboration, The Phase-2 Upgrade of the CMS Data Acquisition and High Level Trigger, Tech. rep., CERN, Geneva, 2021.

[51] CMS Colaboration, Particle-Flow Event Reconstruction in CMS and Performance for Jets, Taus, and MET, Tech. rep., CERN, Geneva, 2009.

[52] F. Beaudette, The CMS Particle Flow Algorithm, *International Conference on Calorimetry for the High Energy Frontier*, 295–304, 2013.

[53] B. Akgun, The CMS High Granularity Calorimeter for HL-LHC - a tracking detector for calorimetry, Tech. rep., CERN, Geneva, 2019.

[54] CMS Colaboration, The Phase-2 Upgrade of the CMS Endcap Calorimeter, Tech. rep., CERN, Geneva, 2017.

[55] M. O. Wiehe, The CMS High Granularity Calorimeter for the High Luminosity LHC, Tech. rep., CERN, Geneva, 2022.

[56] H. Gerwig, Engineering challenges in mechanics and electronics in the world's first particle-flow calorimeter at a hadron collider: The CMS high-granularity calorimeter, *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 1044, 167493, 2022.

[57] J.-B. Sauvan and on behalf of the CMS Collaboration, Concepts and design of the CMS high granularity calorimeter level-1 trigger, *Journal of Physics: Conference Series*, 928, 1, 012026, 2017.

[58] S. Pandey, Performance of High Granularity Calorimeter prototypes for the CMS HL-LHC upgrade in beam test experiments at CERN, https://cds.cern.ch/record/2810369, 2022, [Accessed on 2024-02-20].

[59] T. Quast, *Beam Test Calorimeter Prototypes for the CMS Calorimeter Endcap Upgrade: Qualification, Performance Validation and Fast Generative Modelling*, Springer Cham, 2021.

[60] M. Bonanomi, Beam-tests of CMS high granularity calorimeter prototypes at CERN, *Journal of Instrumentation*, 15, 04, C04001, 2020.

[61] T. Quast, Construction and beam-tests of silicon-tungsten prototype modules for the CMS high granularity calorimeter for HL-LHC, *Journal of Instrumentation*, 13, 02, C02044, feb 2018.

[62] C. Bishop, *Pattern Recognition and Machine Learning*, 16, 140–155, Springer New York, NY, 2006.

[63] L. Cao, Data science: A comprehensive overview, *ACM Computing Surveys*, 50, 3, 2017.

[64] J. Qiu, Q. Wu, G. Ding, Y. Xu and S. Feng, A survey of machine learning for big data processing, *EURASIP Journal on Advances in Signal Processing*, 2016, 05 2016.

[65] M. Najafabadi, F. Villanustre, T. Khoshgoftaar, N. Seliya, R. Wald and E. Muharemagic, Deep learning applications and challenges in big data analytics, *Journal of Big Data*, 2, 12 2015.

[66] S. Bridges, M. Figueroa, C. Diorio and D. Hsu, Field-programmable learning arrays, S. Becker, S. Thrun and K. Obermayer, editors, *Advances in Neural Information Processing Systems*, 15, MIT Press, 2002.

[67] J. Ruiz-Rosero, G. Ramirez-Gonzalez and R. Khanna, Field programmable gate array applications—a scientometric review, *Computation*, 7, 4, 2019.

[68] C. Sun, T. Nakajima, Y. Mitsumori, Y. Horii and M. Tomoto, Fast muon tracking with machine learning implemented in FPGA, *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 1045, 167546, 01 2023.

[69] R. Ospanov, C. Feng, W. Dong, W. Feng, K. Zhang and S. Yang, Development of a resource-efficient FPGA-based neural network regression model for the ATLAS muon trigger upgrades, *The European Physical Journal C*, 82, 06 2022.

[70] E. Khoda, D. Rankin, R. Lima, P. Harris, S. Hauck, S.-C. Hsu, M. Kagan, V. Loncar, C. Paikara, R. Rao, S. Summers, C. Vernieri and A. Wang, Ultra-low latency recurrent neural network inference on FPGAs for physics applications with hls4ml, *Machine Learning: Science and Technology*, 4, 03 2023.

[71] A. L. Samuel, Some studies in machine learning using the game of checkers, *IBM Journal of Research and Development*, 3, 3, 210–229, 1959.

[72] D. Solyali, A comparative analysis of machine learning approaches for short-/long-term electricity load forecasting in cyprus, *Sustainability*, 12, 04 2020.

[73] G. Langs, S. Röhrich, J. Hofmanninger, F. Prayer, J. Pan, C. Herold and H. Prosch, Machine learning: from radiomics to discovery and routine, *Der Radiologe*, 58, 06 2018.

[74] A. Esteva, A. Robicquet, B. Ramsundar, V. Kuleshov, M. DePristo, K. Chou, C. Cui, G. Corrado, S. Thrun and J. Dean, A guide to deep learning in healthcare, *Nature Medicine*, 25, 01 2019.

[75] T. Ching, D. Himmelstein, B. Beaulieu-Jones, A. Kalinin, T. Do, G. Way, E. Ferrero, P. Agapow, M. Zietz, M. Hoffman, W. Xie, G. Rosen, B. Lengerich, J. Israeli, J. Lanchantin, S. Woloszynek, A. Carpenter, A. Shrikumar, J. Xu and C. Greene, Opportunities and obstacles for deep learning in biology and medicine, *Journal of The Royal Society Interface*, 15, 20170387, 04 2018.

[76] J. Goodell, S. Kumar, W. M. Lim and D. Pattnaik, Artificial intelligence and machine learning in finance: Identifying foundations, themes, and research clusters from bibliometric analysis, *Journal of Behavioral and Experimental Finance*, 32, 08 2021.

[77] S. Athey, *The Impact of Machine Learning on Economics*, 507–552, National Bureau of Economic Research, Inc., 2019.

[78] S. Kaffash, A. Nguyen and J. Zhu, Big data algorithms and applications in intelligent transportation system: A review and bibliometric analysis, *International Journal of Production Economics*, 231, 107868, 07 2020.

[79] X. Ying, An overview of overfitting and its solutions, *Journal of Physics: Conference Series*, 1168, 2, 022022, feb 2019.

[80] P. Cunningham and S. J. Delany, Underestimation bias and underfitting in machine learning, https://arrow.tudublin.ie/cgi/viewcontent.cgi?article=1407&context=scschcomcon, [Accessed 30-03-2024].

[81] A. Piotrowski and J. Napiórkowski, A comparison of methods to avoid overfitting in neural networks training in the case of catchment runoff modeling, *Journal of Hydrology*, 476, 97–111, 01 2013.

[82] B. Efron and R. Tibshirani, Improvements on cross-validation: The 632+ bootstrap method, *JASA. Journal of the American Statistical Association*, 92, 06 1997.

[83] P. Burman, A comparative study of ordinary cross-validation, v-fold cross-validation and the repeated learning-testing methods, *Biometrika*, 76, 503–514, 09 1989.

[84] P. Zhang, Model selection via multifold cross validation, *The Annals of Statistics*, 21, 03 1993.

[85] L. Prechelt, Automatic early stopping using cross validation: Quantifying the criteria, *Neural Networks*, 11, 761–767, 06 1998.

[86] A. Collaboration, The ATLAS Trigger System for LHC Run 3 and Trigger performance in 2022, 2024.

[87] M. Sahin, D. Krücker and I.-A. Melzer-Pellmann, Performance and optimization of support vector machines in high-energy physics classification problems, *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 838, 01 2016.

[88] A. Vaiciulis, Support vector machines in analysis of top quark production, *Nuclear Instruments and Methods in Physics Research Section A Accelerators Spectrometers Detectors and Associated Equipment*, 502, 05 2002.

[89] T. Trzcinski, L. Graczykowski and M. Glinka, *Using Random Forest Classifier for Particle Identification in the ALICE Experiment*, 3–17, Springer, Cham, 2020.

[90] M. Azhari, A. Abarda, B. Ettaki, J. Zerouaoui and M. Dakkon, Higgs Boson Discovery using Machine Learning Methods with Pyspark, *Procedia Computer Science*, 170, 1141–1146, 01 2020.

[91] M. Patel and H. Skottowe, A cut-based selection for $Bd \to K^{*0}\mu+\mu-$ at LHCb, Tech. rep., CERN, Geneva, 2010.

[92] D. Alvestad, N. Fomin, J. Kersten, S. Maeland and I. Strumke, Beyond cuts in small signal scenarios: Enhanced sneutrino detectability using machine learning, *The European Physical Journal C*, 83, 05 2023.

[93] C. Bini, Data analysis in particle physics, http://www.roma1.infn.it/~bini/StatEPP_new.pdf, [Accessed December 04th 2013.].

[94] C. Cortes and V. Vapnik, Support vector network, *Machine Learning*, 20, 273–297, 09 1995.

[95] E. García-Gonzalo, Z. Fernández-Muñiz, P. J. Garcia Nieto, A. Sánchez and M. Menéndez, Hard-rock stability analysis for span design in entry-type excavations with learning classifiers, *Materials*, 9, 531, 06 2016.

[96] C.-W. Hsu, C.-C. Chang and C.-J. Lin, A practical guide to support vector classication, https://github.com/tpn/pdfs/blob/master/A2008.

[97] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay, Scikit-learn: Machine learning in Python, *Journal of Machine Learning Research*, 12, 2825–2830, 2011.

[98] P. Vannerem, K. R. Muller, B. Scholkopf, A. Smola and S. Soldner-Rembold, Classifying LEP data with support vector algorithms, *6th International Workshop on New Computing Techniques in Physics Research: Software Engineering, Artificial Intelligence Neural Nets, Genetic Algorithms, Symbolic Algebra, Automatic Calculation*, 1999.

[99] F. Sforza, V. Lippi and G. Chiarelli, Rejection of multi-jet background in p anti-p — > e nu + j anti-j channel through a SVM classifier, *Journal of Physics: Conference Series*, 331, 2011.

[100] D. Whiteson and N. Naumann, Support vector regression as a signal discriminator in high energy physics, *Neurocomputing*, 55, 1, 251–264, 2003.

[101] A. Vossen, Support vector machines in high energy physics, 2008, https://cds.cern.ch/record/1100522/files/p23.pdf.

[102] L. Breiman, Random forests, *Machine Learning*, 45, 5–32, 2001.

[103] M. Y. Khan, A. Qayoom, M. Nizami, M. S. Siddiqui, S. Wasi and K.-U.-R. R. Syed, Automated prediction of good dictionary examples (GDEX): A comprehensive experiment with distant supervision, machine learning, and word embedding-based deep learning techniques, *Complexity*, 2021.

[104] A. Cutler, D. Cutler and J. Stevens, *Random Forests*, 45, 157–176, Springer, New York, NY, 01 2011.

[105] M. Kursa and W. Rudnicki, Feature selection with boruta package, *Journal of Statistical Software*, 36, 1–13, 09 2010.

[106] S. Kuzu, Random forest based multiclass classification approach for highly skewed particle data, *Journal of Scientific Computing*, 95, 02 2023.

[107] M. Azhari, A. Alaoui, Z. Acharoui, B. Ettaki and J. Zerouaoui, Adaptation of the random forest method: solving the problem of pulsar search, *SCA '19: Proceedings of the 4th International Conference on Smart City Applications*, 1–6, 10 2019.

[108] T. Abraham, (Physio)logical circuits: The intellectual origins of the McCulloch-Pitts neural networks, *Journal of the history of the behavioral sciences*, 38, 3–25, 02 2002.

[109] F. Rosenblatt, The perceptron: a probabilistic model for information storage and organization in the brain., *Psychological review*, 65, 6, 386, 1958.

[110] F. Rosenblatt, *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*, Spartan Books, 01 1962.

[111] M. Minsky and S. Papert, *Perceptrons - An Introduction to Computational Geometry*, MIT Press, 1987.

[112] S. Khamaiseh, D. Bagagem, A. Al-Alaj, M. Mancino and H. Alomari, Adversarial deep learning: A survey on adversarial attacks and defense mechanisms on image classification, *IEEE Access*, PP, 1–1, 2022.

[113] Z. Yanling, D. Bimin and W. Zhanrong, Analysis and study of perceptron to solve xor problem, *The 2nd International Workshop on Autonomous Decentralized System*, 168 – 173, 2002.

[114] K. Hornik, M. Stinchcombe and H. White, Multilayer feedforward networks are universal approximators, *Neural Networks*, 2, 5, 359–366, 1989.

[115] H. Afan, A. Ibrahem Ahmed Osman, Y. Essam, A.-M. Najah, Y. Huang, O. Kisi, M. Sherif, A. Sefelnasr, K. Chau and A. El-Shafie, Modeling the fluctuations of groundwater level by employing ensemble deep learning techniques, *Engineering Applications of Computational Fluid Mechanics*, 15, 1420–1439, 2021.

[116] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard and L. D. Jackel, Backpropagation applied to handwritten zip code recognition, *Neural Computation*, 1, 541–551, 1989.

[117] A. Krizhevsky, I. Sutskever and G. Hinton, Imagenet classification with deep convolutional neural networks, *Neural Information Processing Systems*, 25, 01 2012.

[118] H. Gu, Y. Wang, S. Hong and G. Gui, Blind channel identification aided generalized automatic modulation recognition based on deep learning, *IEEE Access*, PP, 1–1, 08 2019.

[119] C. Guo, G. Pleiss, Y. Sun and K. Weinberger, On calibration of modern neural networks, *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, 1321–1330, JMLR.org, 2017.

[120] V. Sheng and C. Ling, Thresholding for making classifiers cost sensitive, *AAAI Conference on Artificial Intelligence*, 1, 01 2006.

[121] C. Sammut and G. Webb, *Encyclopedia of Machine Learning*, Springer Publishing Company, Incorporated, 1st edn., 2011.

[122] W. J. Youden, Index for rating diagnostic tests, *Cancer*, 3, 1, 32–35, 1950.

[123] H. Abramowicz, A. Caldwell and R. Sinkus, Neural network based electron identification in the ZEUS calorimeter, *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 365, 2, 508–517, 1995.

[124] Electron Identification with a Convolutional Neural Network in the ATLAS Experiment, https://cds.cern.ch/record/2850666/export/hx?ln=hr, 2023.

[125] A. Butter and T. Plehn, *Generative Networks for LHC Events*, 191–240, World Scientific Publishing Company, 2022.

[126] S. Otten, S. Caron, W. de Swart, M. Beekveld, L. Hendriks, C. Leeuwen, D. Podareanu, R. Austri and R. Verheyen, Event generation and statistical sampling for physics with deep generative models and a density information buffer, *Nature Communications*, 12, 2985, 05 2021.

[127] B. Dillon, L. Favaro, T. Plehn, P. Sorrenson and M. Krämer, A normalized autoencoder for LHC triggers, *SciPost Physics Core*, 6, 11 2023.

[128] B. Nachman and D. Shih, Anomaly detection with density estimation, *Physical Review D*, 101, 04 2020.

[129] K. Krzyzanska and B. Nachman, Simulation-based anomaly detection for multileptons at the LHC, *Journal of High Energy Physics*, 2023, 01 2023.

[130] K. Fraser, S. Homiller, R. Mishra, B. Ostdiek and M. Schwartz, Challenges for unsupervised anomaly detection in particle physics, *Journal of High Energy Physics*, 2022, 03 2022.

[131] V. Ngairangbam, M. Spannowsky and M. Takeuchi, Anomaly detection in high-energy physics using a quantum autoencoder, *Physical Review D*, 105, 05 2022.

[132] J. Collins, Y. Huang, S. Knapen, B. Nachman and D. Whiteson, Machine-learning compression for particle physics discoveries, https://inspirehep.net/literature/2168899, 2022.

[133] G. Di Guglielmo, F. Fahim, C. Herwig, M. Blanco Valentin, J. Duarte, C. Gingu, P. Harris, J. Hirschauer, M. Kwok, V. Loncar, Y. Luo, L. Miranda, J. Ngadiuba, D. Noonan, S. Memik, M. Pierini, S. Summers and N. Tran, A reconfigurable neural network ASIC for detector front-end data compression at the HL-LHC, *IEEE Transactions on Nuclear Science*, PP, 1–1, 06 2021.

[134] R. Shenoy, J. Duarte, C. Herwig, J. Hirschauer, D. Noonan, M. Pierini, N. Tran and C. Suarez, Differentiable earth mover's distance for data compression at the high-luminosity LHC, *Machine Learning: Science and Technology*, 4, 12 2023.

[135] Y. Huang, Y. Ren, S. Yoo and J. Huang, Efficient data compression for 3D sparse tpc via bicephalous convolutional autoencoder, *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 1094–1099, 2021.

[136] Wulff, Eric, Deep Autoencoders for Compression in High Energy Physics, 2020, Student Paper.

[137] Wallin, Erik, Tests of Autoencoder Compression of Trigger Jets in the ATLAS Experiment, 2020, Student Paper.

[138] N. Nottbeck, D. C. Schmitt and P. D. V. Büscher, Implementation of high-performance, sub-microsecond deep neural networks on FPGAs for trigger applications, *Journal of Instrumentation*, 14, P09014–P09014, 09 2019.

[139] G. Aad, T. Calvet, N. Chiedde, R. Faure, E. Fortin, L. Laatu, E. Monnier and N. Sur, Firmware implementation of a recurrent neural network for the computation of the energy deposited in the liquid argon calorimeter of the ATLAS experiment, *Journal of Instrumentation*, 18, P05017, 05 2023.

[140] G. Aad, A.-S. Berthold, T. Calvet, N. Chiedde, E. Fortin, N. Fritzsche, R. Hentges, L. Laatu, E. Monnier, A. Straessner and J. Voigt, Artificial neural networks on FPGAs for real-time energy reconstruction of the ATLAS LAr calorimeters, *Computing and Software for Big Science*, 5, 12 2021.

[141] *Accelerated Charged Particle Tracking with Graph Neural Networks on FPGAs*, 2020.

[142] H. M. zu Theenhausen, B. von Krosigk and J. Wilson, Neural-network-based level-1 trigger upgrade for the SuperCDMS experiment at SNOLAB, *Journal of Instrumentation*, 18, 06, P06012, 2023.

[143] A. Belousov, I. Kisel, R. Lakos and A. Mithran, Neural-network-based quark & gluon plasma trigger for the CBM experiment at FAIR, *Algorithms*, 16, 7, 2023.

[144] C. Collaboration, Pileup mitigation at CMS in 13 TeV data, *Journal of Instrumentation*, 15, 09, P09018, sep 2020.

[145] A. Kalweit, Data analysis in particle physics, https://indico.cern.ch/event/355973/contributions/1766267/attachments/1124931/1605658/DataAnalysisTeacher.pdf.

[146] CMS Collaboration, CMSSW: CMS Software, https://github.com/cms-sw/cmssw, 2024, [Accessed: 2023-08-03].

[147] T. Sjöstrand, S. Ask, J. R. Christiansen, R. Corke, N. Desai, P. Ilten, S. Mrenna, S. Prestel, C. O. Rasmussen and P. Z. Skands, An introduction to PYTHIA 8.2, *Computer Physics Communications*, 191, 159–177, 2015.

[148] Geant4 Collaboration, Geant4: Geant4 is a toolkit to create simulations of the passage of particles or radiation through matter. (software), 2006, [Accesed: 2023-07-08].

[149] L. Almeida, M. Backovic, M. Cliche, S. Lee and M. Perelstein, Playing tag with ANN: Boosted top identification with pattern recognition, *Journal of High Energy Physics*, 2015, 01 2015.

[150] P. Baldi, K. Bauer, C. Eng, P. Sadowski and D. Whiteson, Jet substructure classification in high-energy physics with deep neural networks, *Physical Review D*, 93, 03 2016.

[151] P. Komiske, E. Metodiev and M. Schwartz, Deep learning in color: towards automated quark/gluon jet discrimination, *Journal of High Energy Physics*, 2017, 12 2016.

[152] L. de Oliveira, M. Kagan, L. Mackey, B. Nachman and A. Schwartzman, Jet-images – deep learning edition, *Journal of High Energy Physics*, 2016, 11 2015.

[153] A. Hammad, S. Khalil and S. Moretti, Search for mono-higgs signals in $b\bar{b}$ final states using deep neural networks, *Phys. Rev. D*, 107, 075027, Apr 2023.

[154] T. Flacke, J. Kim, M. Kunkel, P. Ko, J. Pi, W. Porod and L. Schwarze, Uncovering doubly charged scalars with dominant three-body decays using machine learning, *Journal of High Energy Physics*, 2023, 11 2023.

[155] M. Prvan, Algorithms for the Level-1 trigger with the HGCAL calorimeter for the CMS HL-LHC upgrade, https://llr.in2p3.fr/IMG/pdf/thesis_postfinal_prvan.pdf, 2020.

[156] S. Kotsiantis, D. Kanellopoulos and P. Pintelas, Data preprocessing for supervised learning, *International Journal of Computer Science*, 1, 111–117, 01 2006.

[157] C. Fan, M. Chen, X. Wang, J. Wang and B. Huang, A review on data preprocessing techniques toward efficient and reliable knowledge discovery from building operational data, *Frontiers in Energy Research*, 9, 03 2021.

[158] Y. Liu, L. Lita, R. Niculescu, K. Bai, P. Mitra and C. Giles, Real-time data preprocessing technique for efficient feature extraction in large scale datasets, *International Conference on Information and Knowledge Management, Proceedings*, 981–990, 10 2008.

[159] L. de Amorim, G. Cavalcanti and R. Cruz, The choice of scaling technique matters for classification performance, *Applied Soft Computing*, 133, 109924, 2023.

[160] G. Kumar and P. K. Bhatia, A detailed review of feature extraction in image processing systems, *2014 Fourth International Conference on Advanced Computing & Communication Technologies*, 5–12, 2014.

[161] M. Andrews, J. Alison, S. An, B. Burkle, S. Gleyzer, M. Narain, M. Paulini, B. Poczos and E. Usai, End-to-end jet classification of quarks and gluons with the CMS open data, *Nucl. Instrum. Methods Phys. Res. Sect. A: Accel. Spectrometers Detect. Assoc. Equip.*, 977, 164304, 2020.

[162] K. Pearson, Liii. on lines and planes of closest fit to systems of points in space, *Philosophical Magazine Series 1*, 2, 559–572, 1901.

[163] H. Hotelling, Analysis of a complex of statistical variables into principal components., *Journal of Educational Psychology*, 24, 498–520, 1933.

[164] F. Pantaleo and M. Rovere, The Iterative Clustering framework for the CMS HGCAL Reconstruction, Tech. rep., CERN, Geneva, 2023.

[165] G. Grasseau, A. Kumar, A. Sartirana, A. Lobanov and F. Beaudette, A deep neural network method for analyzing the CMS High Granularity Calorimeter (HGCAL) events, *EPJ Web Conf.*, 245, 02003, 2020.

[166] M. A. Mansoori and M. R. Casu, Efficient FPGA implementation of PCA algorithm for large data using high level synthesis, *2019 15th Conference on Ph.D Research in Microelectronics and Electronics (PRIME)*, 65–68, 2019.

[167] D. Fernandez, C. Gonzalez, D. Mozos and S. Lopez, FPGA implementation of the principal component analysis algorithm for dimensionality reduction of hyperspectral images, *Journal of Real-Time Image Processing*, 16, 1–12, 10 2019.

[168] G. V. Cybenko, Approximation by superpositions of a sigmoidal function, *Mathematics of Control, Signals and Systems*, 2, 303–314, 1989.

[169] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell and S. Xie, A ConvNet for the 2020s, *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 11966–11976, 2022.

[170] K. He, X. Zhang, S. Ren and J. Sun, Deep residual learning for image recognition, *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778, 2016.

[171] M. Tan and Q. V. Le, Efficientnet: Rethinking model scaling for convolutional neural networks, *ArXiv*, abs/1905.11946, 2019.

[172] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, You only look once: Unified, real-time object detection, *IEEE Conference on Computer Vision and Pattern Recognition*, 779–788, 06 2016.

[173] L. Bernstein, A. Sludds, R. Hamerly, V. Sze, J. Emer and D. Englund, Freely scalable and reconfigurable optical hardware for deep learning, *Scientific Reports*, 11, 02 2021.

[174] A. B. Mišura, J. Musić, J. Ožgović and D. Lelas, Performance Comparison of Generic and Quantized Fully Connected and Convolutional Neural Networks for Real- Time Signal/Background Classification, *International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, 2022.

[175] T. Akiba, S. Sano, T. Yanase, T. Ohta and M. Koyama, Optuna: A next-generation hyperparameter optimization framework, *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, 2623–2631, Association for Computing Machinery, New York, NY, USA, 2019.

[176] M. Vilares Ferro, Y. Doval Mosquera, F. J. Ribadas Pena and V. M. Darriba Bilbao, Early stopping by correlating online indicators in neural networks, *Neural Networks*, 159, 109–124, 2023.

[177] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. Mahoney and K. Keutzer, *A Survey of Quantization Methods for Efficient Neural Network Inference*, 291–326, Chapman and Hall/CRC, 01 2022.

[178] G. Shomron, F. Gabbay, S. Kurzum and U. Weiser, Post-training sparsity-aware quantization, 2021.

[179] Google, QKeras: a quantization deep learning library for tensorflow keras (software), https://github.com/google/qkeras, 2019, [Accessed: 2023-08-03].

[180] L. Geiger and P. Team, Larq: An open-source library for training binarized neural networks, *Journal of Open Source Software*, 5, 45, 1746, 2020.

[181] Y. Bengio, N. Léonard and A. Courville, Estimating or propagating gradients through stochastic neurons for conditional computation, *CoRR*, abs/1308.3432, 2013.

[182] S. Gupta, A. Agrawal, K. Gopalakrishnan and P. Narayanan, Deep learning with limited numerical precision, *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, 1737–1746, JMLR.org, 2015.

[183] T. Bai, J. Luo, J. Zhao, B. Wen and Q. Wang, Recent advances in adversarial training for adversarial robustness, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI-21)*, 4312–4321, 08 2021.

[184] M. Prvan, A. B. Mišura, V. Pekić and J. Musić, The transfer learning-based approach for electromagnetic signal classification using simulated HGCAL data, *2023 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, 1–6, 2023.

[185] S. M. Lundberg, G. Erion, H. Chen, A. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal and S.-I. Lee, From local explanations to global understanding with explainable ai for trees, *Nat. Mach. Intell.*, 2, 1, 2522–5839, 2020.

[186] S. M. Lundberg and S.-I. Lee, A unified approach to interpreting model predictions, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, 4765–4774, Curran Associates, Inc., 2017.

[187] S. M. Lundberg and S.-I. Lee, A unified approach to interpreting model predictions, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, 4765–4774, Curran Associates, Inc., 2017.

[188] A. Vossen, Basics of feature selection and statistical learning for high energy physics, 2008, https://cds.cern.ch/record/1094673/files/p1.pdf.

[189] G. H. Nguyen, A. Bouzerdoum and S. L. Phung, Learning pattern classification tasks with imbalanced data sets, *Pattern Recognition*, 2009.

[190] A. Ali, S. M. Shamsuddin and A. Ralescu, Classification with class imbalance problem: A review, *International Journal of Advances in Soft Computing and its Applications*, 7, 176–204, 01 2015.

[191] S. M. A. Elrahman and A. Abraham, A review of class imbalance problem, *Journal of Network and Innovative Computing*, 2014.

[192] N. Chawla, N. Japkowicz and A. Kołcz, Editorial: Special issue on learning from imbalanced data sets, *SIGKDD Explorations*, 6, 1–6, 06 2004.

[193] A. Fernández, S. Río, N. Chawla and F. Herrera, An insight into imbalanced big data classification: outcomes and challenges, *Complex & Intelligent Systems*, 3, 03 2017.

[194] W. G. Cochran, The $\chi 2$ test of goodness of fit, *The Annals of mathematical statistics*, 315–345, 1952.

[195] G. Forman et al., An extensive empirical study of feature selection metrics for text classification., *Journal of Machine Learning Research*, 3, Mar, 1289–1305, 2003.

[196] A. Bellet, A. Habrard and M. Sebban, Metric learning, *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 9, 1–151, 01 2015.

[197] B. Amidan, T. Ferryman and S. Cooley, Data outlier detection using the Chebyshev theorem, *IEEE Aerospace Conference*, 3814 – 3819, 2005.

[198] S. A. Afghani and W. Y. M. Putra, Clustering with Euclidean Distance, Manhattan - Distance, Mahalanobis - Euclidean Distance, and Chebyshev Distance with Their Accuracy, *Indonesian Journal of Statistics and Its Applications*, 2021.

[199] A. Singh, A. Yadav and A. Rana, K-means with three different distance metrics, *International Journal of Computer Applications*, 67, 13–17, 2013.

Curriculum Vitae

Arijana Burazin Mišura

Arijana Burazin Mišura has extensive experience in teaching and IT. Since 2009, she has worked as an Assistant, Lecturer, and Senior Lecturer at the University of Split, at the University Department of Professional Studies. Her responsibilities include teaching, guiding students, and participating in academic activities to improve education quality and learning environments. Before her teaching career, Arijana worked as a Web Developer at Protyron Web Solutions from 2001 to 2008. During this time, she enhanced her web development skills and gained valuable experience in the tech industry, which she now applies in her teaching role. Arijana's academic qualifications include a degree in Mathematics, specializing in Mathematical Statistics and Informatics, obtained from the Faculty of Science (PMF) in Zagreb in 1997. Her technical skills include proficiency in operating systems like Windows and Linux, and programming languages such as Python, Java, and JSP. She also has experience using MATLAB for computational tasks, showing her adaptability in both education and technology. Arijana's blend of practical IT experience and strong academic background makes her a valuable asset in education, where she continues to inspire and educate the next generation of professionals.

Arijana Burazin Mišura ima bogato iskustvo u nastavi i informatici. Od 2009. je radila je kao asistent, predavač i viši predavač na Sveučilištu u Splitu, na Sveučilišnom odjelu za stručne studije. Njezine obveze uključuju podučavanje, vođenje studenata, te sudjelovanje u akademskim aktivnostima za poboljšanje kvalitete obrazovanja i učenja okruženja. Prije svoje profesorske karijere, Arijana je radila kao web programer u tvrtki Protyron Web Solutions od 2001. do 2008. Tijekom tog vremena unaprijedila je svoje programerske vještine i stekla dragocjeno iskustvo u tehnološkoj industriji, koje sada primjenjuje u svom podučavanju. Arijanine akademske kvalifikacije uključuju diplomu iz matematike, specijalizaciju iz Matematičke statistike i informatike, dobivenu na Prirodoslovno-matematičkom fakultetu (PMF) u Zagrebu 1997. Njezine tehničke vještine uključuju poznavanje operativnih sustava poput Windowsa i Linux te programskih jezika kao što su Python, Java i JSP. Ona također ima iskustva u korištenju MATLAB-a u nastavi, što pokazuje njezinu prilagodljivost u području obrazovanja i tehnologije. Arijanin spoj praktičnog iskustva u informatici i jakog akademskog obrazovanja čini je vrijednom u obrazovanju, gdje nastavlja inspirirati i poučavati sljedeće generacije profesionalaca.