Summer 8-2022

MACHINE LEARNING AND SOFTWARE SOLUTIONS FOR DATA QUALITY ASSESSMENT IN CERN'S ATLAS EXPERIMENT

Cary Randazzo

MACHINE LEARNING AND SOFTWARE SOLUTIONS FOR DATA QUALITY ASSESSMENT IN CERN'S ATLAS EXPERIMENT

by

Cary Randazzo B.S. Physics

A Thesis Presented in Partial Fulfillment
of the Requirements of the Degree
Master of Science

COLLEGE OF ENGINEERING AND SCIENCE
LOUISIANA TECH UNIVERSITY

August 2022

LOUISIANA TECH UNIVERSITY

GRADUATE SCHOOL

April 1, 2022
Date of thesis defense

We hereby recommend that the thesis prepared by

Cary Randazzo, B.S. Physics

entitled **MACHINE LEARNING AND SOFTWARE SOLUTIONS FOR DATA QUALITY ASSESSMENT IN CERN'S ATLAS EXPERIMENT**

be accepted in partial fulfillment of the requirements for the degree of

Master of Science in Applied Physics

Dr. Lee Sawyer
Supervisor of Thesis Research

Dr. Neven Simicevic
Head of Physics

Thesis Committee Members:
Lee Sawyer
Thomas Bishop
Rakitha Beminiwattha

Approved:

Hisham Hegab
Dean of Engineering & Science

Approved:

Ramu Ramachandran
Dean of the Graduate School

GS Form 13
(01/20)

ABSTRACT

The Large Hadron Collider (LHC) is home to multiple particle physics experiments designed to verify the standard model and push our understanding of the universe to its limits. The ATLAS detector is one of the large general-purpose experiments that make use of the LHC and generates a significant amount of data as part of its regular operations. Prior to physics analysis, this data is cleaned through a data assessment process which involves significant operator resources. With the evolution of the field of machine learning and anomaly detection, there is great opportunity to upgrade the ATLAS Data Quality Monitoring Framework to include automated, machine learning based solutions to reduce operator requirements and improve data quality for physics analysis. This thesis provides an infrastructure, theoretical foundation and a unique machine learning approach to automate this process. It accomplishes this by combining 2 heavily documented algorithms (Autoencoders and DBScan) and organizing the dataset around geometric descriptor features. The results of this work are released as code and software solutions for the benefit of current and future data quality assessment, research, and collaborations in the ATLAS experiment.

APPROVAL FOR SCHOLARLY DISSEMINATION

The author grants to the Prescott Memorial Library of Louisiana Tech University the right to reproduce, by appropriate methods, upon request, any or all portions of this Thesis. It is understood that "proper request" consists of the agreement, on the part of the requesting party, that said reproduction is for his personal use and that subsequent reproduction will not occur without written approval of the author of this Thesis. Further, any portions of the Thesis used in books, papers, and other works must be appropriately referenced to this Thesis.

Finally, the author of this Thesis reserves the right to publish freely, in the literature, at any time, any or all portions of this Thesis.

Author _____

Date _____

DEDICATION

To those that would make their own luck, gain wisdom, and take action in the face of adversity, I dedicate this thesis to you. The moment you decide to do something of value and *act* on it, you win.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

I would like to first thank my advisor and chair of my thesis committee Dr. Sawyer. Without the opportunity he gave me to work on this project, the detailed explanations, and frequent visits to his office, this thesis would not be possible. I would also like to thank Dr. Bishop and Dr. Beminiwattha for being part of the committee for this thesis. A special thanks to John Spurgeon for assisting in the deployment of software and models for this work via Docker. And thank you to the ATLAS collaboration for bringing people of diverse backgrounds together, pursuing knowledge that can be of benefit to all, and providing the tools necessary to build on that foundation.

CHAPTER 1

INTRODUCTION

1.1 Introduction to High Energy Physics and the LHC

High energy physics is a field that studies the fundamental components of the universe. As the universe is currently understood, the standard model predicts the existence of elementary particles that make up all matter. The objective of high energy physics is to explore and test the standard model, the particles that make it up, and the particles' interactions with one another. One of the largest collaborative operations in the world that studies high energy physics is the *Conseil Européen pour la Recherche Nucléaire* (CERN, European Council for Nuclear Research). With many uncertainties to explore in high energy physics, physicists at CERN are currently searching for answers to many questions. In order to explore these questions, 574 feet beneath the surface of the France-Switzerland border was created the current largest particle accelerator in the world, the Large Hadron Collider (LHC).

The LHC is the home of various high energy physics experiments including the CMS, ALICE, LHCb, and ATLAS experiments, and required roughly 17 miles of tunnel for its construction. Each of the detectors are similar in nature but have a few advantages and disadvantages with respect to one another in terms of design. For example, the CMS detector features a 4T magnetic field in its solenoid while the ATLAS detector has only a 2T solenoid field allowing it an advantage in terms of momentum resolution from the

tracker. However, this strong magnetic field imposes limitations on the other detector components. Thus, the ATLAS detector is less restrictive in the component design leading to a less dense and larger volume detector whose components have advantages in its hadronic calorimeter detector system compared to CMS. [1]

1.2 Introduction to ATLAS and the Data Processing Chain

The solution developed in this work will be in reference to and applied with respect to the ATLAS (A Toroidal LHC ApparatuS) experiment. The ATLAS experiment is one of the four longest running experiments run in the LHC. Its purpose is to study the Standard Model as well as push the frontiers of knowledge beyond the model.

The LHC produces proton particle beams to analyze proton-proton collisions. The lifecycle process of this data can be thought of as "The ATLAS Data Processing Chain" [2]. In the beginning of this chain, particles pass through various parts of the detector and data is collected based on triggers. Trigger data is calibrated, aligned, then reconstructed.

As part of the data reconstruction, various groups are responsible for specific areas of data preparation. The data quality group is responsible for "how good" the data is, so that it may be safely used for physics analysis.

The "express_express" or Express Stream contains triggers to monitor efficiencies, backgrounds, and detector noise and is used for quick Data Quality and prompt calibration. It is promptly reconstructed at the ATLAS Tier 0 facility and occurs simultaneously to data taking allowing it to be used for rapid validation and Data Quality assessment tasks [2].

Following the Express stream reconstruction job and Data Quality Monitoring Framework checks, monitoring histograms are produced and used for offline data quality

assessment. Currently, data quality assessment for anomalies in these histograms are handled by shifters with the aid of these monitoring algorithms [3].

In addition to the Express stream, there also is a "physics_Main" stream. According to [4], "In some cases, the number of events collected in the express stream is not sufficient to perform a full assessment of the DQ. In these cases, the express stream is used as a preliminary check, and the final assessment is done when looking at the same histograms but produced by processing all events collected in the physics stream" This stream should include a cleaner, more holistic view of experimental data by ATLAS and its sub-detectors. Therefore, this stream will contain the data that is focused on training and testing the machine model [4].

With the vast amount of data available to ATLAS and the field of machine learning based anomaly detection rapidly expanding, there is great opportunity for its data quality professionals to maximize the accuracy, efficiency, and human resources by harnessing this technology. The results of this work bring the tools and models necessary for these improvements to data quality professionals of ATLAS.

1.3 Introduction to Validation

Occasionally, the software used by ATLAS may require changes. These changes can be from a variety of persons for a variety of reasons. For example, software changes may occur if certain kinds of data are needed by scientists, improvements can be made to the results of the sub-detectors, or new technology needs to be added to the system. For various reasons, software changes can cause undesirable behavior in the detector's experimental data. The purpose of validation in ATLAS is to identify this undesirable behavior if it exists and correct it. Naturally, several technologies have been developed to

assist in this process. While preparing the infrastructure for this research, a tool was also developed that could assist individuals in such validation - The ATLAS Validation Dashboard. Validation is not the focus of this work, but this dashboard establishes the foundation for deployment of the forthcoming machine learning model. Thus, there will be only a brief mention of it in the final sections.

1.4 Research Objectives

Three opportunities for using machine learning tools in this domain include reduction of human operator (known as shifter) input, reduction of time and complexity of the procedures that would require this input and generating higher quality reference histograms to maximize the quality of data for physics experiments further down "The ATLAS Data Processing Chain". The overall goals of this kind of project are as follows:

- Reduce shifter input necessary to assess data quality using monitoring histograms via machine learning and deployment solutions
- Reduce time and complexity for assessing data quality using monitoring histograms via machine learning and deployment solutions
- Generate reference histograms that more accurately represent the collision data when no anomalies are present via machine learning solutions

The specific goals and contributions of this work will provide the infrastructure, theoretical foundation, and first-generation software that will culminate in the realization of these data quality assessment objectives for the ATLAS experiment.

1.5 Typical Approaches

Previous research focused on solutions to the previously identified objectives with the following machine learning model architectures and software tools:

- Convolutional Neural Networks (CNNs): Initial interest and previous work done in particle physics data quality assessment models included Convolutional Neural Networks.[5] For the scope, input data strongly resembles image data, thus this approach will be tested for potential use in a final model.
- Denoising Autoencoders (DAEs): Current literature strongly supports use of Autoencoders and autoencoder based unsupervised machine learning methods for anomaly detection solutions.[6] The data contained in the histograms is largely absent ground truth anomaly labels, thus this approach will be tested for potential use in a final model.
- Variational Autoencoders (VAEs): In addition to the strong support of AEs and unsupervised learning methods, Variational Autoencoders are known for their ability to reconstruct distributions that trained data came from by extracting the mean and standard deviations from that data.[6] This approach allows data quality experts to generate reference histograms from the trained dataset. Since the first-generation solution will not touch on the reference histogram generation yet, this information is provided as reference for extending this work.

The final approach will start out with the above techniques in mind but will also explore a unique final approach based on experiments and modifications done with various approaches to this task. In this way, this work will take advantage of any internal patterns in the data using the research objectives as a guide. The user end deployment of

the tools will be made possible with a dashboard using Dash by Plotly and Docker (see section 3.10 and 3.11).

1.6 What to Expect

This section describes a summary of what to expect in the upcoming chapters of this work. In the first chapter, details are provided on the overall domain of the work, how the task for this work ties to that domain, the overall and specific objectives of this work, and an explanation is given on the approach with respect to common approaches to this task. The second chapter explains that domain in greater detail, provides background details of core technology used in this work, and sets the scope and limitations of the work to fulfill the objectives. The third chapter lists the essential information leading up to the experiments, initial explorations and experiments, core technology that has been and will be developed during this work, construction of the dataset for the experiments, the experiments themselves, important information relevant to final automation of the machine learning system, and the intended deployment methods. The fourth chapter presents the results of the previously described systems and experiments that demonstrate the abilities of the developed system and begins the discussion of those results. The fifth chapter further develops the discussion and meaning of those results including assumptions, limitations, and possibilities for future work. The final chapter concludes with a summary of the objectives, the results that were achieved, and how it compares to the current system.

CHAPTER 2

BACKGROUND

2.1 Introduction

2.1.1 CERN and the LHC

The *Conseil Européen pour la Recherche Nucléaire* (European Council for Nuclear Research), or CERN is an organization that was established in 1954 for the purposes of scientific discovery and collaboration. It was also decided that their work would have no investment in military interests and that the work produced by the organization would be publicly published. Throughout CERN's lifetime, improvements and upgrades have been made to verify the limits of the Standard Model as it is called in particle physics, as well as discover how the universe might work beyond that model [7].

In 2008, these efforts led to the completion of the Large Hadron Collider (LHC), a nearly 27-kilometer-long particle accelerator that passes through both France and Switzerland. It is currently the largest and highest energy particle collider in the world making the collaboration Louisiana Tech University works with, the ATLAS experiment, one of the largest scientific collaborations in the world. The LHC's main function is to accelerate and collide particle beams of protons, lead-proton and lead-lead heavy ions. The information in this work has focused on data specific to proton collisions only. CERN's accelerator complex connects several experiments and devices together (see Fig 1). The LHC is home to several major experiments and some of the largest collaborations

in history such as the CMS, ALICE, LHCb, and ATLAS experiments. The detectors used in these experiments vary from general purpose studies to specific interests such as studying flavor physics like that of CP violations in B-hadrons.

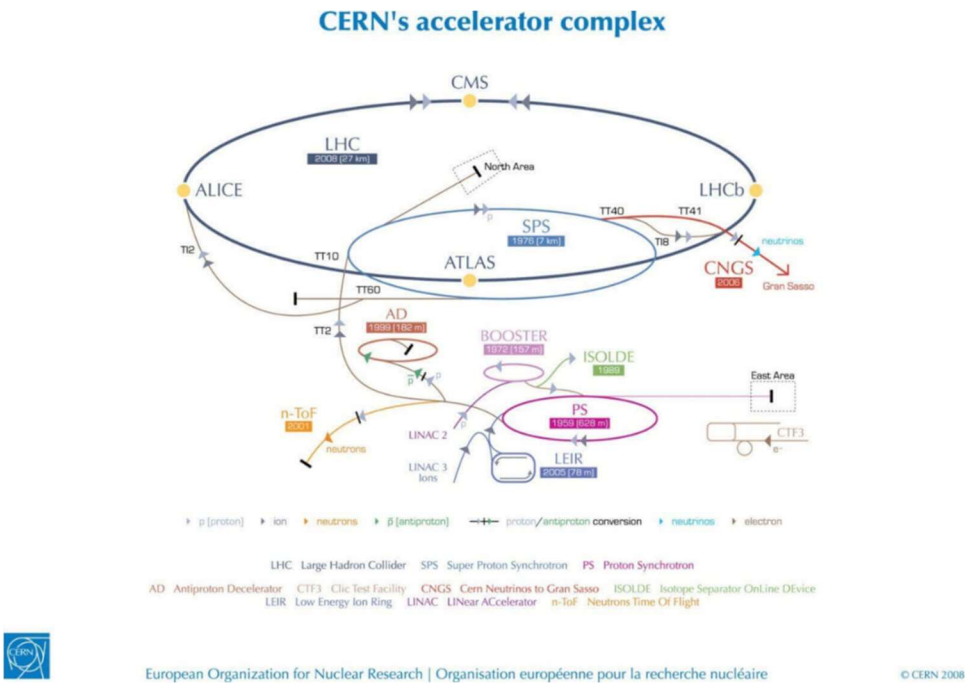

Figure 1: The CERN Accelerator Complex [8].

2.1.2 The ATLAS Detector

The ATLAS detector is the device that makes the ATLAS experiment possible. Containing some one-hundred-million electric channels and its various sub-detectors including the muon detector, inner detector, and calorimeters, the ATLAS detector is capable of identifying various particles and particle energies. Unlike other detectors of its kind, ATLAS has a magnetic field of two Tesla in its central solenoid allowing more flexibility in design considerations, such as a larger overall volume and advantages in certain sub-detectors such as the Hadronic Calorimeter system [9].

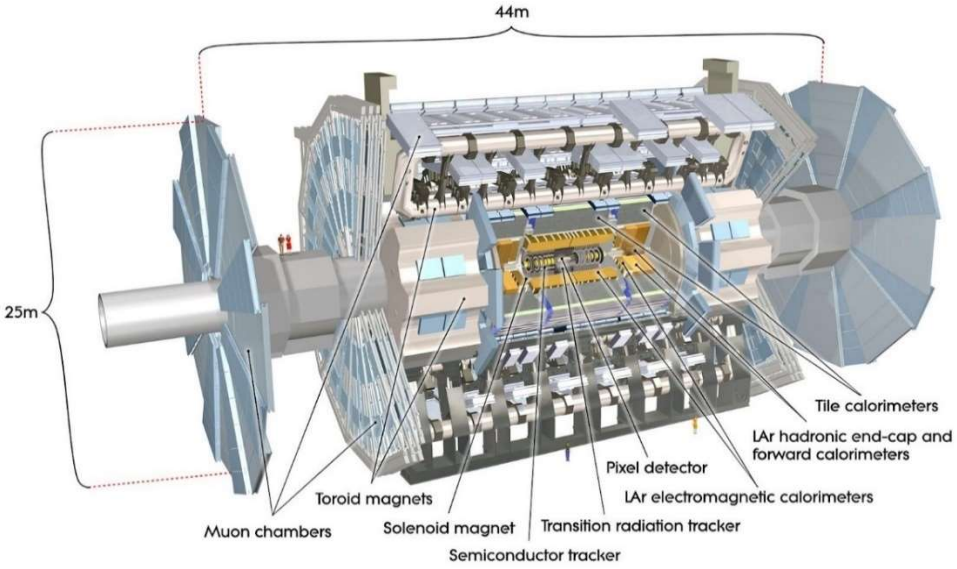

Figure 2: The ATLAS detector and its sub detector systems [10].

2.1.3 The ATLAS Calorimeter Systems

The area of the detector that houses the calorimeter sub-detectors are where the datasets constructed for the experiment will originate from. The calorimeter is made up of two systems, the Liquid-Argon (LAr) calorimeter and the hadronic calorimeter. Within these calorimeters includes 8 sections: The LAr Hadronic End-Cap (HEC), the Min Bias Trigger Scintillators (MBTS), the Tile Barrel (TB), the Tile-Extended Barrel (TEB), the LAr Forward Calorimeter (FCAL), the Cryostat, the LAr EM barrel (EMB), and the LAr Electromagnetic End-Cap (EMEC). As the overall detector structure is symmetric about the collision point, there tends to be part of each sub detector on each side such as HEC1 and HEC2 (see **Figure 3**). Hadronic calorimeters such as the HEC, TB, and TEB calorimeters are designed to measure energies of particles that interact with the strong nuclear force. The HEC is also a sampling calorimeter where the Copper LAr structure alternates materials that absorb particle energy and active media that measures the energy

signal while the Tile calorimeters use an Iron scintillator structure [11]. The MBTS is responsible for providing key physics measurements via event triggers such as proton-proton cross section, charge multiplicity, and others. It is a scintillator sub detector made of polystyrene disks that can detect particle energies as they pass by scintillation of the material [12]. The cryostat is implemented to maintain the liquid state of Argon at a temperature of 185℃ [13]. The FCAL is made of copper and tungsten and LAr fills the gaps between the materials. The EMEC and EMB calorimeters are accordion shaped lead plates and cover the pseudorapidity of $1.375 < |\eta| < 2.5$ and $2.5 < |\eta| < 3.2$ respectively.

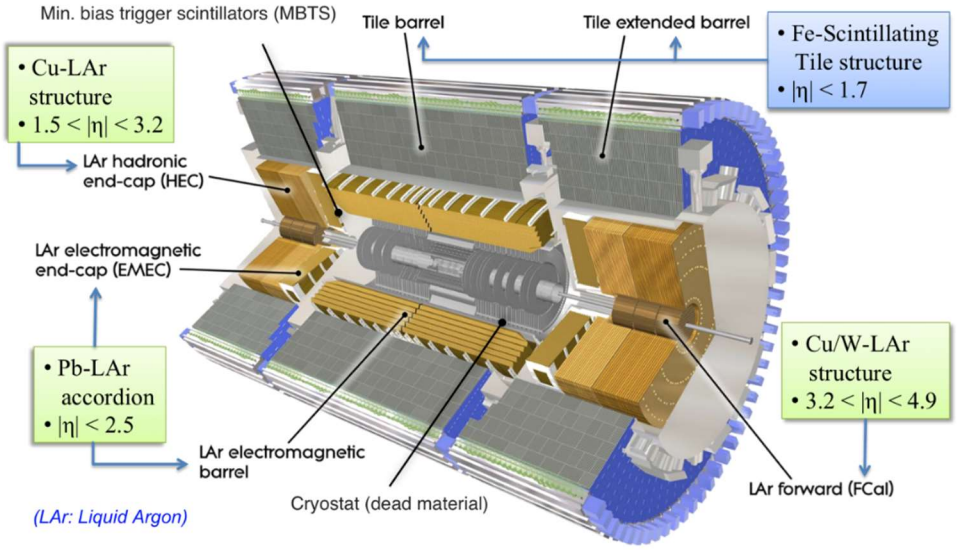

Figure 3: The Calorimeter based systems of the ATLAS detector [14].

The main connection between these sub-detector parts, and this work is that the pseudorapidity values for each calorimeter sub-detector section coincide with the pseudorapidity values in monitoring histograms for those locations in the calorimeter.

2.2 Data Quality in ATLAS

The purpose of gathering data in the ATLAS experiment is partly to confirm and discover physics beyond the Standard Model. In order to do this, the data provided by the ATLAS detector is subject to physics analysis. For the physics analysis to be of value, the data from the detector must be as free from errors as possible. Realistically, the detector is imperfect, and errors occur in the form of excessive noise, electronic malfunctions, and more. Identifying and correcting these errors is the primary reason for the existence of Data Quality Assessment.

Due to the size of the detector and the large amount of data being passed from data to storage, the responsibilities of Data Quality Assessment are divided up among several groups - one for each sub system in the detector. These groups include the LAr, MS, Pixel, SCT, Tile, and TRT groups. Also, there are several groups tasked with handling the combined data performance of several sub systems. These groups include b-tagging, CaloCombined, ID global, and MuonGlobal groups.

As can be seen in **Figure 4**, the Data Quality Workflow (DQW) for RUN II is shown. The DQW shows the lifecycle of data as it makes its way through several review systems and is processed to different collections known as streams. The various streams allow data quality checks and calibration of systems at various times in the processing timeline. As part of the data quality checks, a system called the Data Quality Monitoring Framework (DQMF) has been implemented to assist Shifters and/or Data Quality Experts in assessing the quality of data.

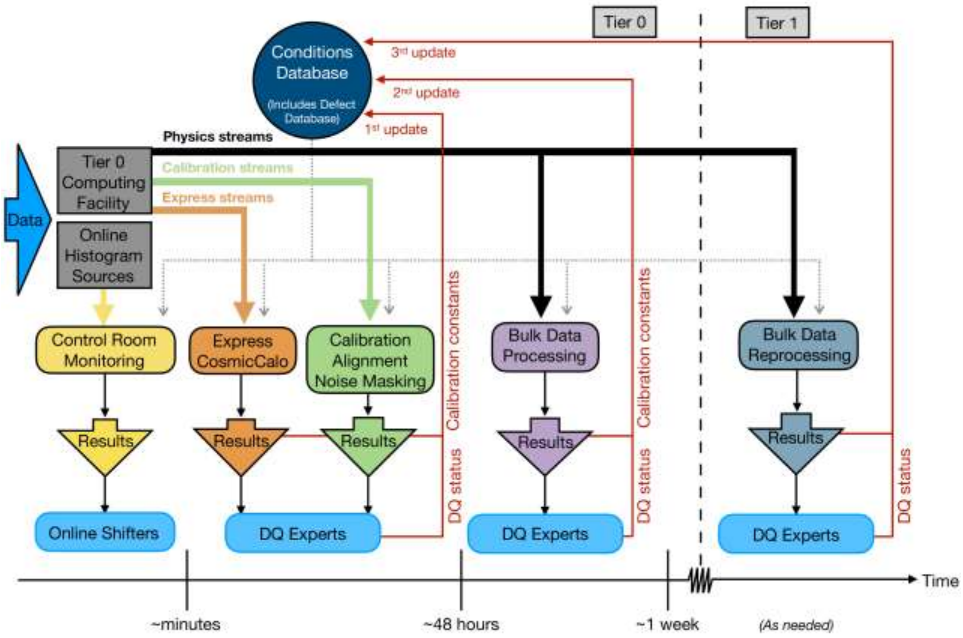

Figure 4: The Data Quality Workflow (DQW) for ATLAS [15].

The DQMF is a computerized system that assists both online and offline Shifters with various data quality checks. This system exists as a series of programs that include DQParameters, DQAlgorithms, DQResults, and a web display. The DQParameters contain the necessary input information to the web display to provide shifters with information relevant to the DQ task at hand. The information that the parameters include are histogram information, algorithms, and programs specific parameters required by the algorithms. The DQAlgorithms or algorithms are currently a collection of statistical software run over histograms to assist in determining the quality of data or quality of that histogram. The DQResult is the result of that algorithm's classification as either good or bad. It is not necessarily the DQParameters or web display that are relevant to this work's objectives. However, the DQAlgorithms and DQResults are important as they are currently the primary means of machine assisted data quality assessment in the ATLAS DQMF.

The DQAlgorithms include statistical methods of identifying anomalous data such as “BinsDiffFromStripAvg” and “BinsDiffFromStripMedian”. While these algorithms are reasonably effective, it is a modest assumption to say that with the vast amount of data ATLAS has generated and the advances in the field, a machine learning approach could be superior. In fact, similar approaches are already being implemented in other LHC experiments. The other distinct advantage within this system that machine learning offers is the ability to reduce the manpower (Shifter/DQExpert) input necessary to achieve data quality assessment of incoming data. The results of this work will provide a machine learning solution that will be the first step in achieving these objectives and the objectives outlined in the previous chapter. **Figure 5** shows the current flow of data from the perspective of the DQMF.

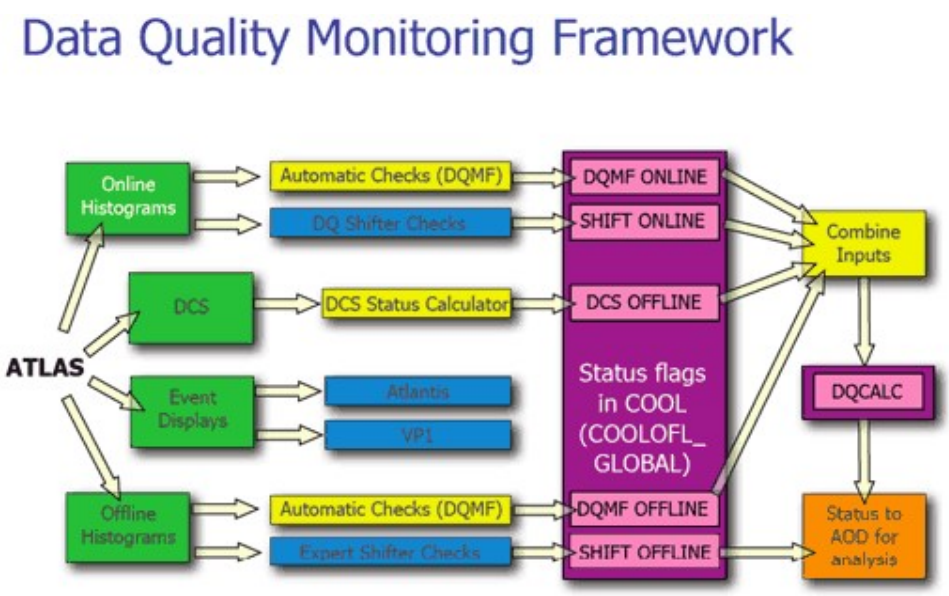

Figure 5: The Data Quality Monitoring Framework (DQMF) of ATLAS [16].

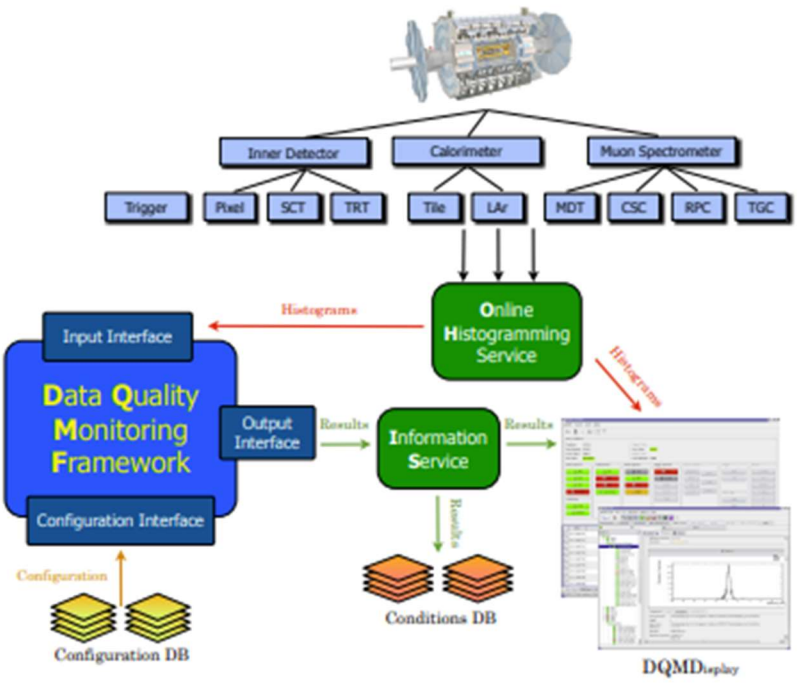

Figure 6: Another view of the DQW as it is connected to the DQMF and Data Quality Monitoring Display (DQMD) [17].

Online Shifters are workers who rotate in and out on a 24-hour shift and are tasked with monitoring detector components and reconstruction algorithms in real time. They seek to identify anomalous information while being aided by various systems including the DQMF and document that information and the data that was taken when an error occurred. Offline Shifters have similar tasks although they are less in number, and their work is less time sensitive. Considering this, a new machine learning data quality system that would improve on the existing DQMF would be of most benefit to the online shifter group but is also beneficial to offline shifters.

The data quality monitoring histograms are the main subject of dataset that will be constructed. As mentioned previously, the histograms are generated on the web display by the DQMF by way of DQParameters, DQAlgorithms, and DQResults. Furthermore,

the histograms are of several types including TH1 type, TH2 type, and TProfile type histograms provided by the ROOT framework. Each sub detector gathers data that will be subject to physics analysis, so its data quality must be assessed. Thus, the monitoring histograms include data from all sub detectors. This work will focus on the TH2 type of monitoring histogram, but the results could be extended to other types of monitoring histograms in this area similar to approaches by other LHC experiments. This work will also focus on monitoring histograms from the calorimeter sub detectors that are part of the CaloCombined and LAr groups.

Half of all offline monitoring histograms that will be present in the constructed dataset comes from the CaloCombined combined performance group, while the other half comes from the LAr group. Details on monitoring histograms will be explained further in this section and the details about the dataset will follow in Chapter 3.

2.3 Anomaly Detection

What is an anomaly? Anomalies or outliers can be defined as data points that lie outside an expected range or distribution. They are a type of rare information or event that does not seem to fit with the system [18]. Of course, for an observer to judge if it is outside an expected range or distribution, there must be up front assumptions made about what the entire population looks like. Very rarely do scientists have all information regarding a population or its distribution and this makes identifying outliers at the highest level most difficult. On the other hand, it is often much simpler to classify a data point as anomalous on strict and pre-defined rules.

2.3.1 Statistical Anomaly Detection

Statistical anomaly detection seeks to identify outliers by rules and mathematical principles. One such principle known popularly as the Empirical Rule [19] assumes that all data points that fall at the extreme ends of a distribution (beyond the 99.7 percentile range) are anomalous. Assuming that outliers are rare events occurring in low frequency, data occurring less than 0.3% of the time makes relative sense. Other statistical approaches to anomaly detection are available, but having described one such method, the approach in current use by the ATLAS DQMF system will now be examined.

The current DQAlgorithm in use for the monitoring histograms that will be in the constructed dataset is called "BinsDiffFromStripMedian". The DQResult of "good" (green), "review" (yellow), or "bad" (red) is given based on this algorithm and roughly on how far any individual data point is from the median of the strip in η to which it belongs. Formally, the algorithm exists in the CERN Gitlab [20] with the same name. The thresholds for a histogram to be deemed green, yellow, and red are compared to the "outstandingRatio" (Equation 1) and classified accordingly. The outstandingRatio or BinsDiffFromStripMedian formula is defined in the algorithm as follows:

$$o = \frac{b_i - s_{med}}{\sqrt{|s_{med}|}} \tag{1}$$

where b_i is the i-th occupancy value of the strip and s_{med} is calculated as the average of the first three quantile values of the strip:

$$s_{med} = \frac{\frac{n}{4}+\frac{n}{2}+\frac{3n}{4}}{3} \quad (2)$$

Assume a strip of size 4 and with basic increasing values ([1,2,3,4]) then the median according to this formula differs from the median definition according to statistics:

$$s_{medTrue} = \frac{n_{odd}}{2} \text{ or } \frac{\frac{n_{even}}{2}+\frac{n_{even}+1}{2}}{2} \quad (3)$$

where n is the sample size.

The former formula would yield a median of 2 and the latter formula would yield a median of 2.5. It has been assumed this method was chosen in the algorithm for the necessity of having an integer value.

What is the meaning of outstandingRatio o? Occupancy(bin) values cannot be negative. Therefore, s_{med} the sum of all positive numbers must be positive. Looking first at the numerator, $(b_i - s_{med})$, the Median Absolute Deviation (MAD) which is similar in form to the numerator is given by:

$$MAD = med(|b_i - med(\vec{b})| \quad (4)$$

removing the median of this absolute deviation:

$$AD = |b_i - med(\vec{b})| \quad (5)$$

And finally removing the absolute value of the deviation yields:

$$D_{med} = b_i - med(\vec{b}) \quad (6)$$

In the algorithm, there are additional limits in place that allow the formula to not need the absolute value of this difference. What is left in Equation 6 is the amount of residual occupancy that deviates from the median as in the numerator.

Since the numerator is a deviation, the Standard Error (SE) for that deviation can be defined, typically given in terms of standard deviation about the mean, by dividing by the square root of the positive sample size s_{med}:

$$o = \frac{D_{med}}{\sqrt{|s_{med}|}} \sim SE \tag{7}$$

for individual deviations.

This SE gives a measure for how accurate the value of any sample from that population is likely to be compared to the true population median [21]. **T**he addition of absolute value to the sample size is specific to the kind of values these strips can take on and the additional limits and controls included in the algorithm. Thus, the larger this value is, the more likely it is to be an outlier, and this corresponds to the ascending threshold values moving from green to yellow, and finally red.

How effective are these methods? The 2 and 3 sigma rules are reasonably effective and is widely known, but the main issue with this method is that the standard deviation itself is calculated including outliers that can affect its value. Many outliers or large values of outliers could lead to issues with this method's results. The MAD method is generally more effective at detecting outliers but can lead to a high false positive rate in the outlier classification. The algorithm further underestimates the median by taking only the integer part and further increases the probability of a false positive when subtracting out a value that is lower than the true mean [22].

2.3.2 Machine Learning Based Anomaly Detection

As the field of machine learning continues to attract attention, it is no surprise that the topic of anomaly detection is currently seeing a surge in machine learning based techniques. From the information given in Typical Approaches(1.5), the discussion of

anomaly detection in machine learning in this context begins with understanding the general machine learning approaches and will proceed until details have been developed about convolutional neural networks, various kinds of autoencoders, and methods relevant to this work. The general approaches can be seen in **Figure 7**.

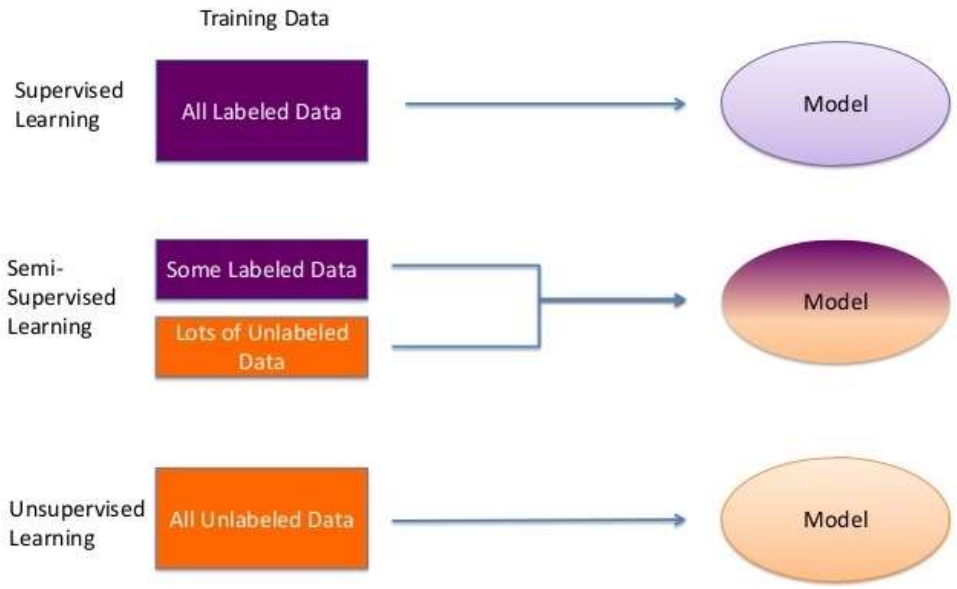

Figure 7: An overview of 3 general types of machine learning approaches [23].

The first major approach to a machine learning project is the Supervised Machine Learning (SML) method. SML derives its name from the idea that the machine learning algorithm learns how to predict or classify information from known ground truth target data. Having a clear target for the algorithm to learn from, an advantage of performance characteristics is present since a benchmark for comparison is available. The ground truth data is often a time-consuming hand labeled feature making its strongest asset often its greatest challenge. The primary SML algorithm of interest for this work that will be discussed in a later section is the Convolutional Neural Network.

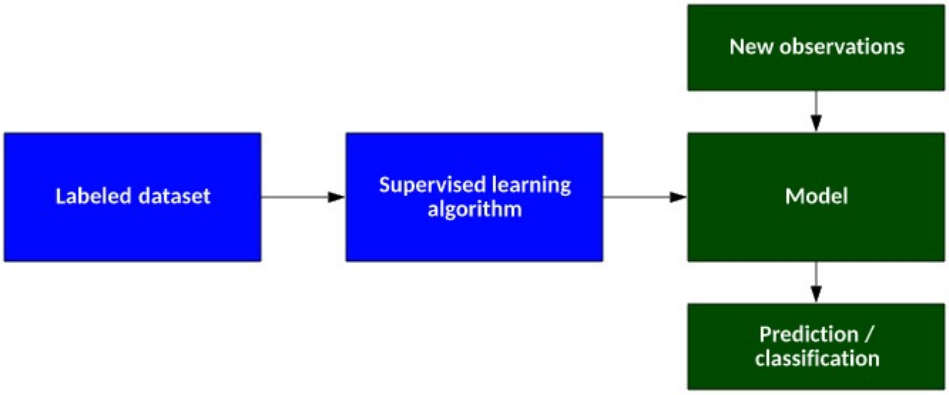

Figure 8: A more specific description of the supervised machine learning approach [24].

Another popular machine learning approach is the Semi Supervised Machine Learning (SSML) approach. The algorithm for SSML works by using both hand labeled and unlabeled target data and using both Supervised and Unsupervised approaches - often further generating its own ground truth labels in process. This approach is more of a technique rather than indicative of a specific model architecture. In real world applications, approaches that use both supervised and unsupervised methods often happen and a careful study of this subject can be beneficial.

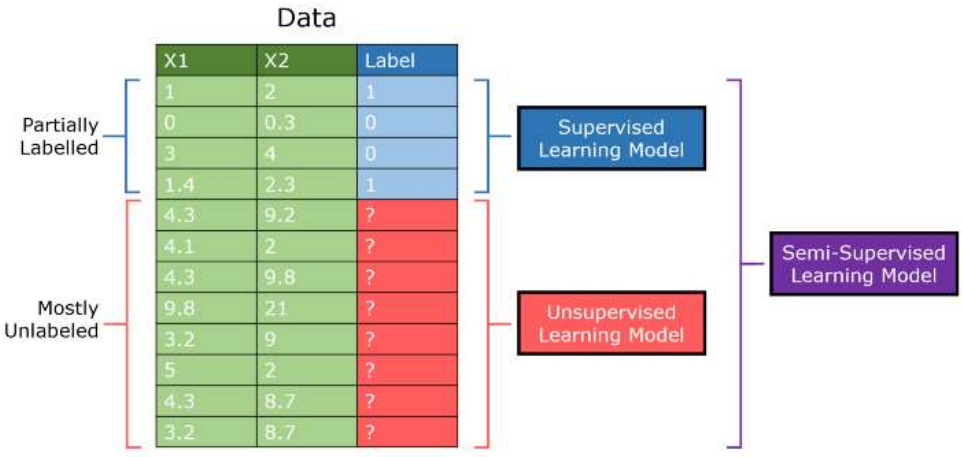

Figure 9: A more specific look at the semi-supervised approach [25].

Unsupervised Machine Learning (UML) algorithms are the last approach that will be discussed. This approach is the exact opposite of SML in that it uses and requires no ground truth label for the algorithm. One huge advantage of these methods is that most data is unlabeled by nature and these algorithms have the potential to highlight patterns that go unnoticed. One of the major drawbacks of UML algorithms is that they are notoriously hard to evaluate as evaluation requires human assumptions on how well an algorithm performs (i.e., target values to compare the predictions to). Sometimes, Autoencoder Neural Networks are classified as SSML algorithms, but they are a special case of UML known as Self Supervised Machine Learning as it maps its own input as ground truth labels.

Within anomaly detection in machine learning, there are also common approaches. These approaches include probabilistic, ensemble based, neural network based, linear based, and combination-based methods. The majority of these methods are unsupervised, but there are also some supervised methods. More supervised methods are available in the classification algorithms due to the ground truth classes of outlier and non-outlier when available. One such algorithm, the Local Outlier Factor (LOF) initially seemed like a good candidate, but many basic machine learning algorithms were limited in the complexity of the features they could learn or were not designed to perform well with a significant amount of data. Due to the overwhelming amount of data this particle physics application provides, neural network and deep learning methods were identified to be more suitable [26].

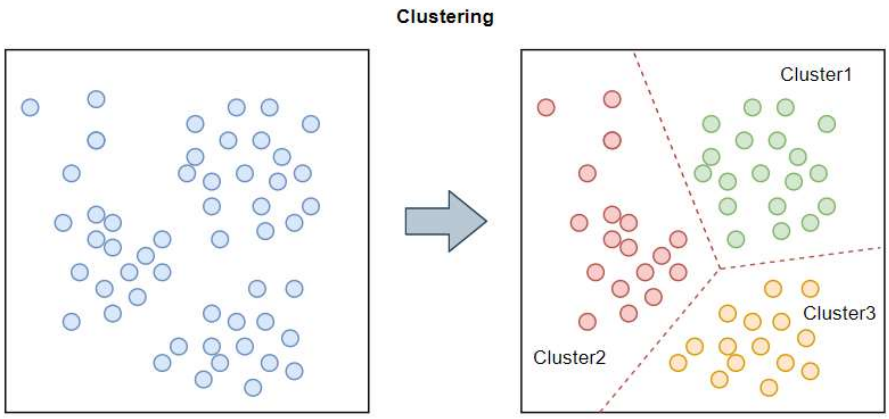

Figure 10: A more specific look at an unsupervised learning approach. This method depicts a clustering algorithm in unsupervised learning [27].

2.3.3 DBScan

DBScan is a density based unsupervised machine learning algorithm developed by Ester et.al. [28]. Its main function is to cluster data points that fall within a pre-set minimum number of points of a neighborhood whose range is defined by some pre-set distance. In this work, the minimum number of points is set to its lowest value to allow the clusters to spread more easily. The DBScan algorithm is strong against noise and varying size clusters but does not handle clusters of varying density well. This can be an advantage in cases such as in this work where it is expected normal points semi tightly clustered together and outliers in small external outlying clusters will be found. It also is known to be sensitive to parameter adjustments, so finding the correct setting can be challenging as well as important. Of the various clusters that DBScan identifies, the class labeled -1 is reserved for noise or outlier labels identified by the algorithm and is the cluster of most interest for this work. In general, the algorithm can be summarized by the following **Figure 11**:

Algorithm 1: DBSCAN algorithm

```
DBSCAN (W, MinPts,Eps)
Input: a data set W, MinPts
Output: arbitrary shape clusters
for each data point p ∈ W do
    if p is not mark as 'seen' then
        Mark p as 'seen'
        Find neighborhood of data point p, NeighborPts
        if NeighborPts < minimum points then
            Mark data point as a noise
            else
                clusterid=clusterid+1
            end
        end
        for all q ∈ NeighborPts do
            Mark data points q as seen
            Find neighborhood of data point q, NeighborPts
            if NeighborPts > minimum points then
                Give data point q a clusterid
            end
        end
    end
end
```

Figure 11: This graphic depicts the general steps involved in setting clusters according to the DBScan algorithm [29].

2.3.4 Neural Networks, Deep Learning, and Anomaly Detection

Neural Networks are a special kind of machine learning algorithm designed to learn new information from input information and use that new information to make predictions. The origin of the neural network can be traced back to its fundamental unit, the perceptron where inputs are calculated together in a specific way to form an output prediction. There is a long complex history involving the design and development of neural networks and their many different flavors over the years. Here, details will be listed that are most relevant to this work.

From the Perceptron to the Deep Neural Network

If the most basic form of neural network is the feed forward neural network, then the fundamental building block of that neural network is the perceptron. One of the most important areas of this study is the “Interpretability and Explainability” of deep neural

networks. It is therefore essential to understand all details of how a perceptron functions in a neural network. From here, the details of the perceptron will be described including the mathematics that go into building the neural networks used in this study.

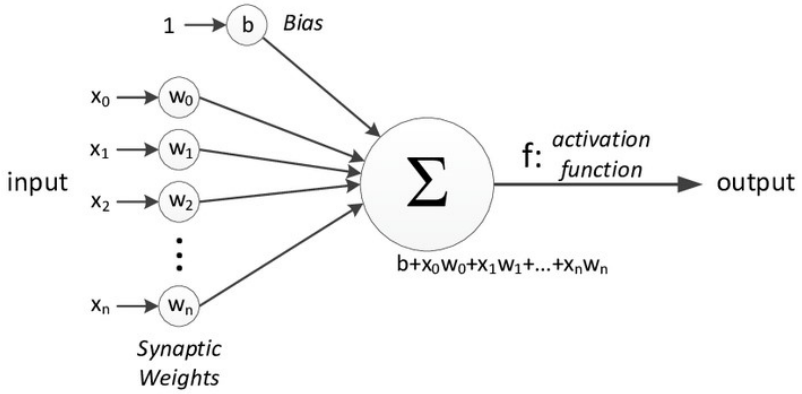

Figure 12: A diagram of the basic perceptron neural network model [30].

The perceptron only uses forward propagation as the original outline of the perceptron was done by [31] whereas the original outline of backpropagation was not introduced until [32].

According to [31], the calculated value at any node or neuron is the weighted sum of the inputs plus the bias. This value is calculated before applying the activation function such that:

$$a = \sum_0^n x_i w_i + b \tag{8}$$

Following calculation of the node (a), an activation function is applied over the resulting value. Here, the depicted activation function (∑) is the sigmoid function where

$$\sum(x) = \frac{1}{1+e^{-x}} \tag{9}$$

Here, the output y is given when the activation function is applied to this single node. Applying this function over (a) results in the following:

$$y = \sum(a) = \frac{1}{1+e^{-a}}$$

$$= \frac{1}{1+e^{-(\sum_0^n x_i w_{i0} + b_0)}} \tag{10}$$

Perceptrons were initially used as binary classifiers. This means they are naturally suited for binary classification problems such as anomaly detection. Unfortunately, they are an oversimplification of what machine learning based anomaly detection is currently capable of.

Having discussed the fundamental unit of a neural network, a description of the first deep learning neural network relevant to this study will be given. The Convolutional Neural Network is both a special kind of Feed Forward Neural Network and a Deep Learning Neural Network.

A simple Feed Forward Neural Network (FFNN) is built from a collection of inputs mapped to all nodes in a single hidden layer who are also mapped to all outputs. Each node in the hidden layers functions like the perceptron described previously, and each node in the output layer also functions like the perceptron [33]. In practice, this is a simplification of what is actually at work in a framework like Keras where various other kinds of nodes and layers along with activation functions and weight initialization functions set in the background.

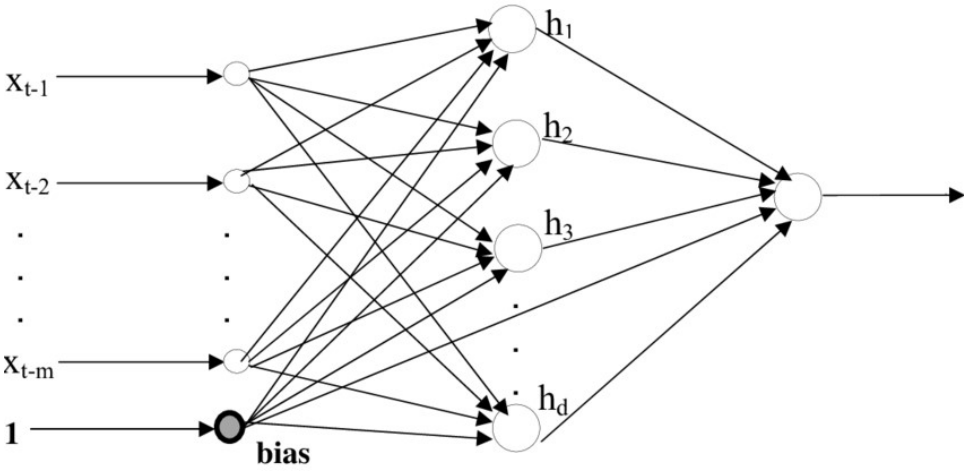

Figure 13: A FFNN with m inputs, a bias of 1, and a single output [34].

There appears to be no strict definition of how Deep Learning Neural Networks (DNNs) are structured, but most sources suggest that a sufficient number of network layers are required for it to be classified as a DNN. The definition of sufficient is where definitions diverge. According to [35], "DNNs are the underlying architectures, which, in contrast to neural networks (NNs), consist of multiple hidden layers."

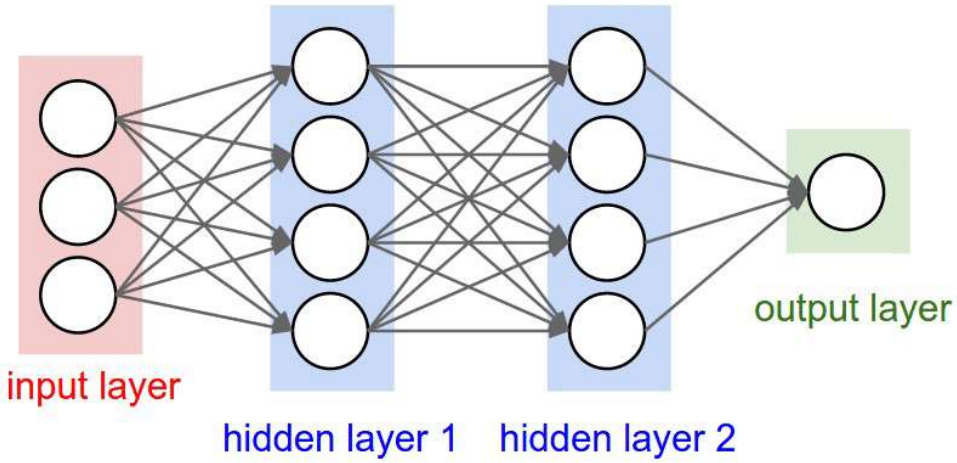

Figure 14: A basic DNN that is also a 2-layer FFNN. This follows the definition that DNNs are NNs with multiple layers [36].

The Convolutional Neural Network

The Convolutional Neural Network (CNN) is a special modification of a FFNN in that a technique derived from work on images (convolution) is performed section by

section in a multidimensional data space in order to generate features that are representative to that section of data. The data space is typically a 2d image (TH2 histogram for this work) and the matrix it is multiplied with to complete the convolution is known as the kernel. The kernel is a weight matrix that is updated during the backpropagation phase of the algorithm. This weight matrix is commonly initialized with random values from a “glorot uniform” distribution. A CNN architecture contains more than just the convolutional layer (see **Figure 16**). The distributions of available kernel initializers using the Keras package are given in the following **Figure 15**.

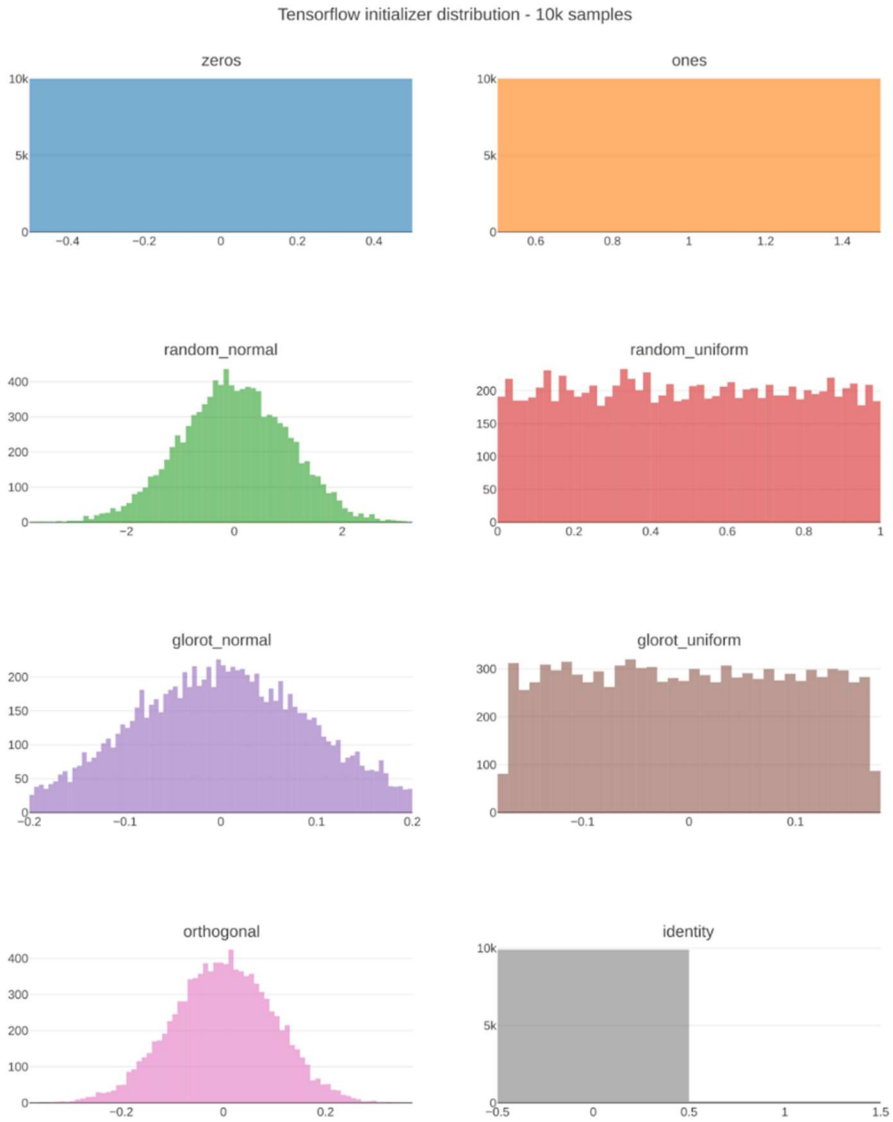

Figure 15: Plots of weight initialization distributions available in the Keras package for python [37].

The advantage of using a CNN architecture derives from the possibility of identifying anomalous data in a monitoring histogram whose anomalies are identifiable due to occupancy differences in localized areas. If the ground truth values are available, the CNN is a strong and commonly used approach in recent anomaly detection works with similar objectives.

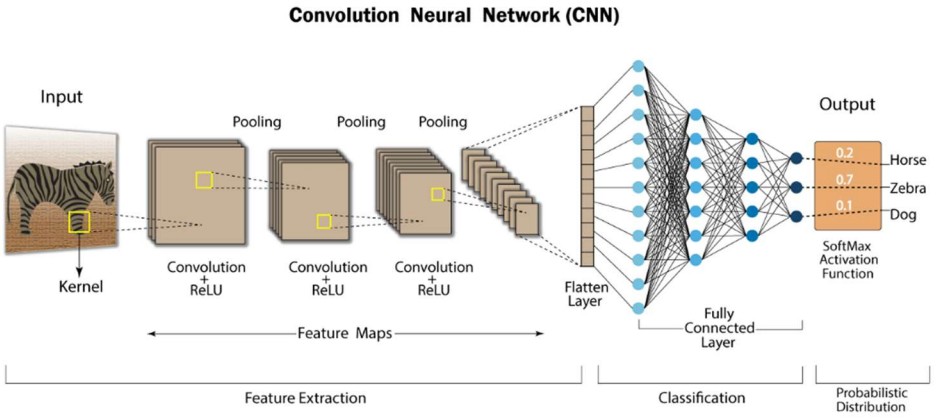

Figure 16: An example of an image classification model whose main architecture is that of a Convolutional Neural Network (CNN) [38].

The Autoencoder

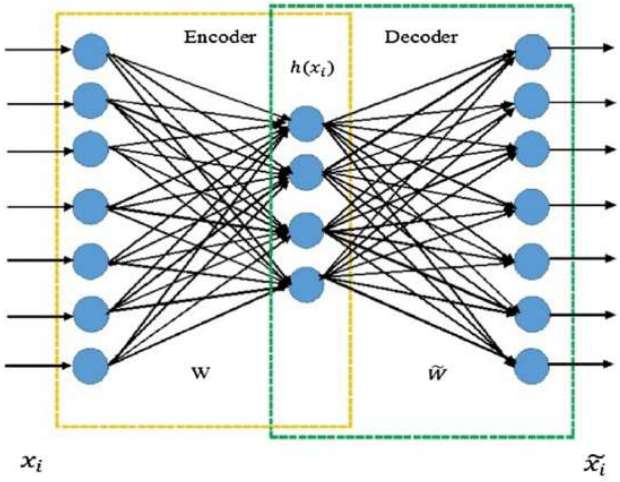

Figure 17: Simple autoencoder with one hidden layer where the values x_i are the inputs, $h(x_i)$ are the latent space values, $\tilde{x}_i$ are the reconstructed input values, w is the encoder weights, and $\widetilde{w}$ is the decoder weights [39].

The Autoencoder (AE) is one of the simplest neural networks. Again, they technically are classified as self-supervised in that they utilize the inputs as both input and target for the model, but they are often interpreted as unsupervised since they do not require labeled data. In this self-supervised approach, an autoencoder will seek to train on

input samples so that it can accurately reconstruct a dataset and register high reconstruction error for anomalous data points. Despite returning a reconstruction error, establishing a dataset whose classes are linearly separable requires different techniques that will be discussed in detail later.

The general structure of an autoencoder is that of an encoding layer, latent space layer, and decoding layer in addition to input and output layers. The encoding layer is where the input features are reconstructed to a smaller feature space similar to PCA, but often in a nonlinear fashion unless specified by the architecture. The latent space is the zone where the compression occurs - information found to be less important is left out of the limited space that more important information will occupy. The decoding layer simply reverses the feature reduction process, but this time on the compressed information resulting in a limited reconstruction of the original input. Some autoencoder structures have the encoding layer and input layer the same while simultaneously having the output layer and decoding layer the same (see **Figure 17**). These types of AEs do not fall under the definition of a DNN, but AEs who have all these layers separate or are more complex are a type of DNN called Deep Autoencoders. All these models will simply be referred to as AEs unless it is of key importance to make the distinction going forward.

Complete Autoencoder

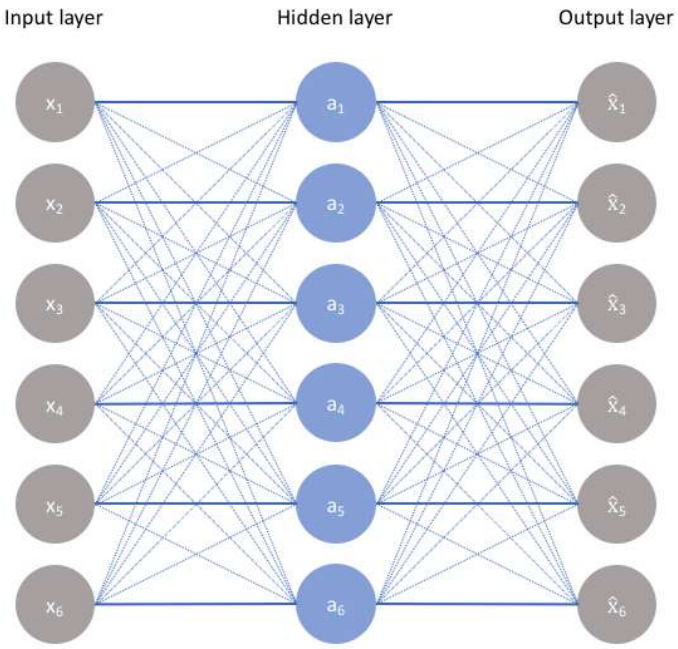

Figure 18: A "complete" autoencoder [40].

This is a type of autoencoder whose encoder has the same number of dimensions as the input layer. The loss is very low for both training and validation sets. It is an example of a simple autoencoder that is overfit. It functions by simply passing along the input information directly through the network, with no bottleneck, and maps the input directly to the output. This network is not designed for anomaly detection, but it is a useful calibration measure before starting experiments.

Denoising Autoencoder

This type of autoencoder more aggressively ignores noise in the data. This would be useful in the case that the original data is suspected to be corrupted.

"Another regularization method, similar to contractive autoencoder, is to add noise to the inputs, but train the network to recover the original input." The architecture

varies from a basic autoencoder setup to sparse and/or stacked autoencoder setups. The preparation of a Denoising Autoencoder requires that inputs are trained on noisy data and targets be noiseless data with the assumption that the algorithm will learn to predict noiseless data from noisy data [41].

Sparse Autoencoder

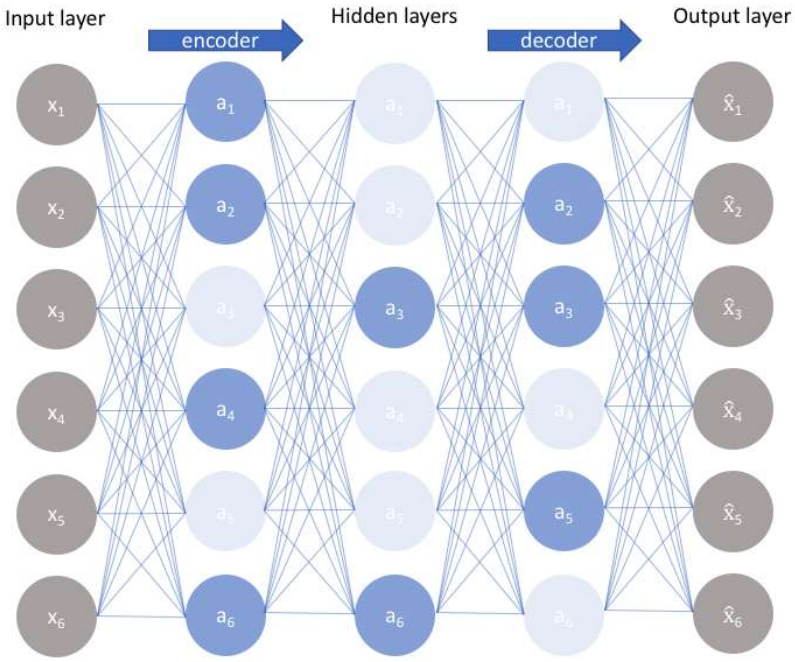

Figure 19: A sparse autoencoder [42].

Autoencoders of the most typical form are called Dense Autoencoders. Sparse autoencoders output a sparse final matrix such that the information captured is better distributed across the features that the autoencoder learns. The result of which should be a final matrix that has more zeros throughout and the information captured will be better distributed across the learned features.

In addition to the standard components of an autoencoder, the sparse autoencoder requires what is known as a sparsity penalty.

Sparse autoencoders are generally overcomplete. They have hidden layers with more units than the number of input features with only a small fraction of the hidden units being allowed to be active at the same time.

A sparsity penalty forces the auto encoder to take the sparsity of the final matrix into consideration.

Stacked Autoencoders

A stacked autoencoder is a neural network consisting of several layers of sparse autoencoders where the output of each hidden layer is connected to the input of the successive hidden layer.

Convolutional Autoencoder

A Convolutional autoencoder makes use of the advantage of convolution applied to standard autoencoder architectures.

As previously mentioned, convolution is a technique where not all elements in a layer are fully connected to all elements in another layer. Instead, it is assumed that features can be extracted in blocks such that individual weights are connected to subsets of neurons rather than every neuron in a convolutional layer. This technique also has the advantage of less fully connected neurons, meaning less neural network parameters being passed forwards and backwards during training, and this means it will significantly reduce training time for applicable systems. These too are reasons that the technique of convolution is commonly used in image datasets.

The convolutional autoencoder combines the two ideas where localized feature information is picked up in the data space and simultaneously mapped through some variation of nonlinear feature reduction space. Of course, this technique will still require

ground truth labels, but is a candidate for the most effective algorithm in the application of this work [43].

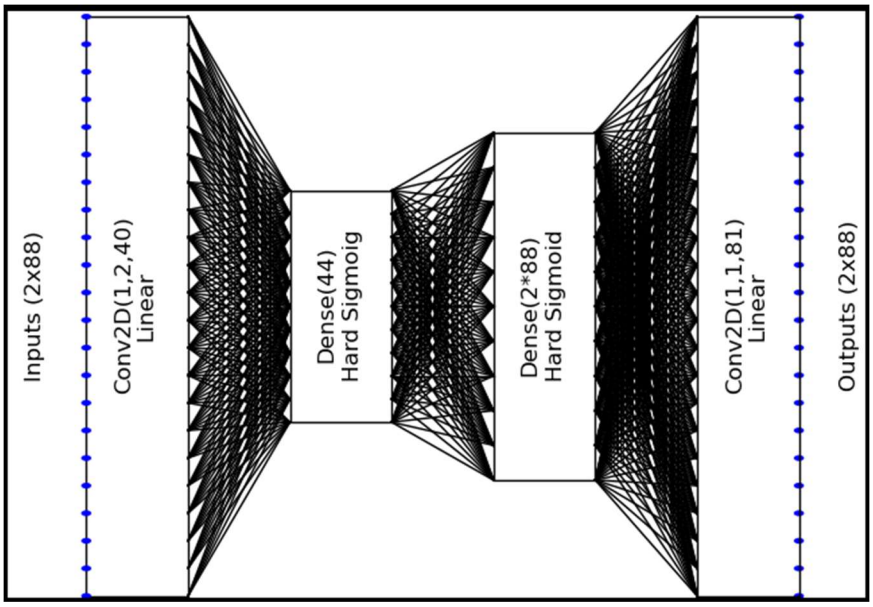

Figure 20: An example Convolutional Autoencoder architecture featuring 2d convolutional layers, and dense latent spaces [45].

Variational Autoencoder

This type of autoencoder has an encoder that outputs two vectors instead of one. The outputs are vectors of the means and the vectors of the standard deviations. The i-th value of these vectors corresponds to the mean and standard deviation of the i-th random variable.

This type of autoencoder generates a sample of the population space. It is able to sample across a continuous space based on what is learned from the input data.

It is not limited to examples it has trained on but can generalize and output new examples even if it has never seen similar ones. It generates synthetic data that appears to belong to the same distribution as the trained data. This led to generative adversarial networks and synthetic data such as images, speech, music, art, etc.

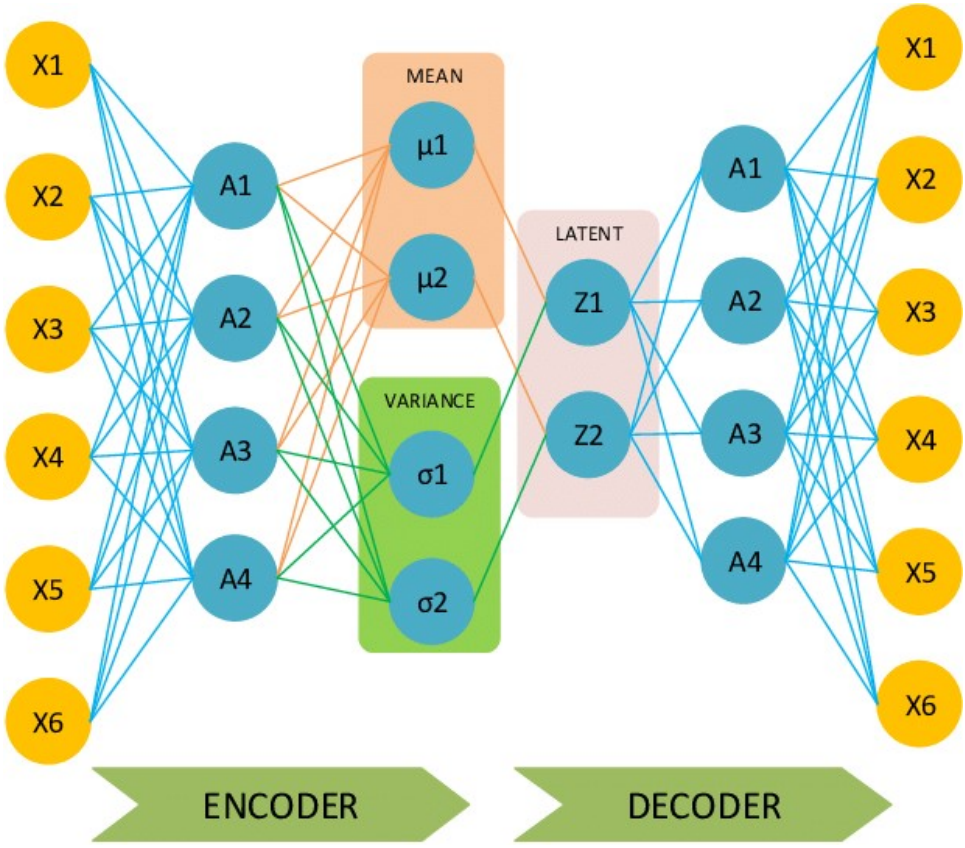

Figure 21: Variational Autoencoder Architecture. The latent space includes features that represent the mean of the learned distribution and variance of the learned distribution. The learned features would then be able to generate samples from this learned distribution.

2.4 Scope and Limitations

In the process of building the machine learning model for these objectives, focus will be placed on interpretation and evaluation to successfully achieve the desired results. At the time of designing this work, the following limitations were placed upon this project due to resources available or requirements of the final product:

- Luminosity independent system – Resources unknown regarding information of the histograms were of interest with respect to luminosity (time series information)

- Histogram independent system – Rather than looking at all data for a sub detector side by side and including these features to determine defects across sub detectors, the task was to find histogram level defects first, whose results would later be compared between histograms (sub-detector level features such as TileCal energy vs HCAL energy).
- Label independent system – It was reported that expert labeled defects were typically recorded in the defect database, but labels lacked the detail required for a histogram level detection system.
- Common defects known, but uncommon defects possible – Known defects make up about 2% of the results according to estimates.

These limitations imply time dependent machine learning algorithms are out as well as the ability to make time dependent predictions (no RNNs, etc. and no time series features). It also implies that as little as a single histogram's worth of data and as much as the entire historical set of data is available. Therefore, it is not possible to mimic expert behavior without a record of expert behavior (Supervised algorithms such as CNNs via labels). Were such labels available, this information suggests that supervised/human labels could perform well in the automation task due to known defect types.

As a result, this approach will focus on coordinate-by-coordinate inputs as is common with neural networks in similar works or an alternative configuration based on geometric features via coordinates. Furthermore, unsupervised methods will be relied on such as autoencoders that are commonly used in anomaly detection literature, but other options will be explored.

In consideration of the lack of historical data available at the start of a LHC Run (changes to the detector cause inconsistent patterns between historical data and gathered data after shutdowns), as few as a single histogram worth of data (6435 coordinates) or about 384 histograms of a single type available per year according to estimates (2,471,040 data points in the dataset per year, but only about 384 data points per input in the dataset per year, shape of 384 samples by 6435 features, could cause overfitting requiring a different configuration), the choice has been made to avoid DNN approaches such as CNNs requiring significant labeled data which is convenient since this information is unavailable.

Currently, anomaly detection literature suggests and previous researchers in this domain have generated anomalies as a method to both improve results and provide a method to evaluate results. This also implies that expert labeled anomaly sets would be skewed against the number of anomalies for prediction results.

To that effect, the objectives will be considered fulfilled for a final model that satisfies a number of the following criteria:

1. As a first phase workable solution, this particular model will be restricted to 2 dimensional histograms of the class TH2 according to the ROOT framework.
2. Furthermore, this model will seek to identify all generated anomalies and/or *at least* present results potentially superior to the in use DQalgorithm per monitoring histogram.
3. The scope of the model will restrict its training and prediction for monitoring histograms primarily to the physics_Main stream. The physics_Main stream will be used to construct and evaluate datasets as is and the version of the

physics_Main datasets that contain generated anomalies will be used for pressure testing the detection system. The physics_Main stream is assumed to be more representative of clean histogram data, but without comparing to the good run lists some additional anomalies can be expected to be present during evaluation for runs not on the certified good run list.

4. The scope of the model will restrict the monitoring histograms to the TH2 histograms in the CaloCombined folders contained in run files and found in the DQ web display.
5. The model is capable of successfully identifying known anomalous data points in data quality monitoring histograms. These anomalous data points can be of the form of individual data points as well as clusters of anomalous data points
    a. Identifies hotspots (occupancy levels in monitoring histograms that are greater than the expected inlier distribution)
    b. Identifies cold spots (occupancy levels in monitoring histograms that are less than the expected inlier distribution)
    c. Identifies hot strips and cold strips (same as hotspots and cold spots except occurring along a single strip in $\boldsymbol{\eta}$ for several values of ϕ)
    d. Identifies hot layers and cold layers (same as hot strips and cold strips except now including several values of $\boldsymbol{\eta}$ and several values of ϕ such that a layer would typically have one value greater than the other).

CHAPTER 3

METHODOLOGY

3.1 Introduction

In this chapter, the process of setting up the experiments in a section-by-section fashion will be described. It will begin with the initial system setup and software requirements, followed by initial investigations, plans to prototype following those investigations, construction details of the datasets, exploratory analysis of the constructed datasets, basic calibration, some information on reproducibility, the experiments, the outlier decision function and outlier probability, performance evaluation and interpretation, the dashboard, and a final section on deployment.

The primary focus of the chapter is providing details of the approach that will achieve the listed objectives for this work but will also focus on details to simplify future work in this domain.

3.2 Hardware, Software, and Packages

The following is a list of hardware, software, and packages that were used in construction of this work. Anyone working on a similar project to this may require "ATLAS Collaborator" status and "GRID credentials" in order to access various essential systems such as "lxplus" where the .root files are moved and downloaded from, "Rucio" [46] where file requests are made, "CERNbox" [47] where files are stored, and "SWAN"

[48] where analytical work for machine learning can be done efficiently. The local system specifications used by the researcher and packages necessary for this work are listed sequentially with descriptions below. The analysis and experiments have been run mainly on SWAN while the local system is used for final software construction.

3.2.1 Local System Specs

- Linux c-B450M-DS3H 5.13.0-30-generic #33~20.04.1-Ubuntu SMP x86_64 x86_64 x86_64 GNU/Linux
- AMD Ryzen 5 1600 Six-core 3.2GHz Processor: 2 threads per core, 6 cores per socket
- RAM: 32GB, DIMM DDR4 3.2GHz
- GPU: Radeon RX 5700 XT
- Docker is an industry standard, portable, lightweight platform for sharing and deploying secure applications [49].

3.2.2 Miniconda3 and Packages Used

Anaconda is a data science targeted application meant to simplify python and R package management and deployment. Miniconda is the lightweight version of that application with no pre-installed packages [50]. The current system uses Miniconda version 4.11.0, and it has the following packages installed with corresponding version numbers in "<>" brackets:

- Python <3.7.6> is a well-known high-level scripting language that is industry standard in the field of data science, but with thriving communities in other scientific areas and non-scientific areas [51].

- ROOT <6.18.00> is an analysis framework developed by CERN to provide a coding infrastructure that will assist in analyses following data collection by the LHC [52].
- Pandas <1.3.4> is a commonly used data management and manipulation package for python. It excels in cleaning, transformation, and other essential practices relevant to scientists and analysts working with data [53].
- Matplotlib <3.5.0> is a commonly used visualization package available for python with great depth in its capabilities [54].
- Seaborn <0.11.2> is another commonly used visualization tool built over Matplotlib. It provides a few unique tools and aesthetic advantages to Matplotlib [55].
- SQLAlchemy <1.4.27> is an open-source SQL package built for Python [56].
- Scikit-learn <1.0.2> is a commonly used machine learning library for Python. While it is useful in many cases, occasionally more configuration is required as in the case of neural networks [57].
- Keras with TensorFlow as a backend <2.4.3> - Keras is a package built on top of TensorFlow, built on top of NumPy, built on top of python. It is specifically geared toward deep learning neural networks as opposed to TensorFlow which is a more general system for neural network development [58].
- Dash by Plotly <1.19.0> is a dashboard development system developed with the use of Flask, Plotly, and Python [59], with the following required component packages installed:

- Dash-bootstrap-components <0.11.1>
  - Dash-html-components <1.0.1>
  - Dash-core-components <1.3.1>

3.3 Initial Steps and Investigations

The initial steps taken and prototypes generated included setting up software, gaining access to ATLAS's online databases and systems, and unpacking that data in the intended python or pyROOT cloud system (SWAN notebook) to view, manipulate, and work with data. Following successful completion of those steps, the next step was to convert the now accessible and viewable histograms to pandas data frame structures. More details on this can be found in Section 3.4 for constructing the dataset.

Having successfully completed the software necessary for converting histograms to data frames, rapid prototyping of what a machine learning solution might look like and do in relation to this project and research goals had begun. Using a small collection of data from several histograms, preliminary results were investigated using pyOD [60] using different algorithms. As specific kinds of patterns were observed in the data, the algorithms that were chosen according to these patterns performed best. Proximity based machine learning approaches were able to classify several data points that were visually identified to be faulty. This suggested an exploitable pattern related to how outlying a data point is for data points in this dataset. From these preliminary results, targets were set to scale up the size and scope of the project to meet its goals with a full solution. The experiments and details that follow are designed from these initial investigations.

3.4 Construction of the Datasets

Note: The full construction of the datasets is also given in step by step detail via Jupyter Notebook files that will be uploaded to GitHub and included as a resource link in the near future.

In order to construct the dataset, the first step was to identify the data needed to train a model that would satisfy the objectives. Also, it was important to identify what resources were available to access it. Ultimately, the objectives required using the same data that is available to the shifters at the time of determining if anomalies are present in the monitoring histograms - the geographical information. The geographical information that is directly viewable from the monitoring histogram itself is the heatmap of occupancies of the TH2 in η and ϕ coordinate space.

The ATLAS detector is roughly cylindrical in shape around the beam lines. The η coordinate in the monitoring histograms, also known as the pseudorapidity in particle physics, is the angle a particle is measured with respect to the beam direction. $\eta = 0$ indicates particle trajectory perpendicular to the beam axis. The ϕ coordinate, otherwise known as the azimuthal angle, refers to the location about the circular cross section of the detector [61]. Together, these two pieces of information, along with the meta information of the histogram itself, make it possible to locate on a monitoring histogram where in the detector the particle location or energy registered during beam crossing, depending on what part of the detector you are viewing data from. The monitoring histograms that were identified for use in this work focused primarily on calorimeter information, therefore the information in the datasets refers to particle energies measured in that respective region of the detector.

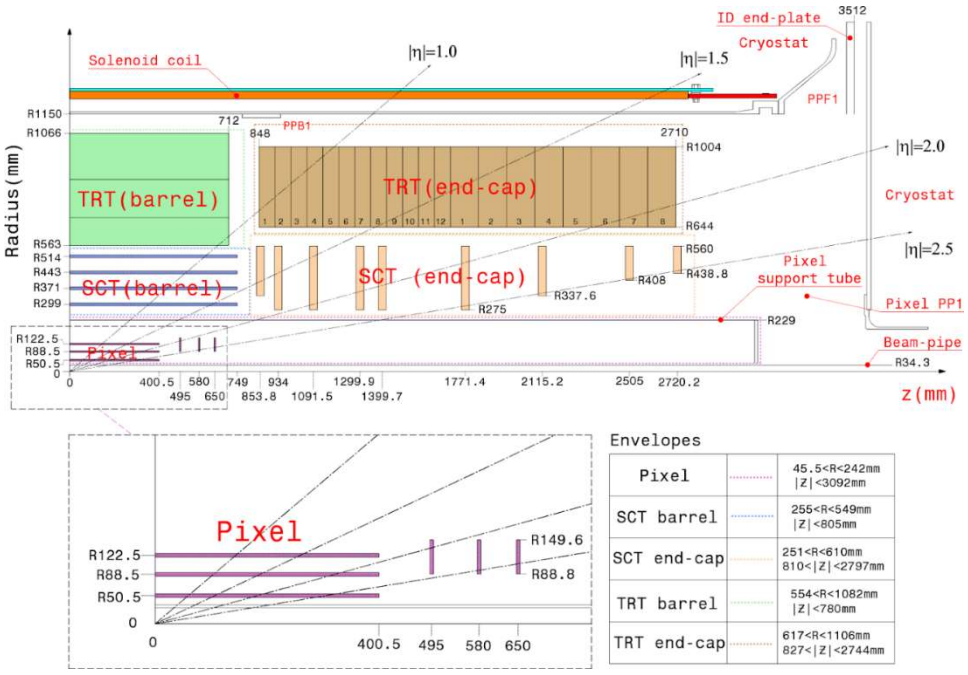

Figure 22: Cross section of the inner detector through the beam axis [61].

Earlier, it was mentioned that 3 features are part of the main datasets: η, ϕ, and occupancy. The next feature of interest is the quality label. There are also 3 additional features that give scaled and/or normalized versions of the occupancy values in a specific way that will be explained further shortly. The quality label holds a value of 0 for already present data points and 1 for generated outlier data points. The remaining features are meta information used to understand and explain where the data comes from and will be used in explaining and interpreting the model results. Those meta features include the path of each datapoint from the run file to the histogram, the ftag id, the histogram id, and histogram type.

3.4.1 Feature List and Explanation

The experiments will report on various combinations of the following features with the interest in the objectives outlined previously in mind.

- paths - For each data point, this feature lists the path in the run file to the monitoring histogram specific to that data point. As part of the pathing information, it also shows the run number it originates from. The format of any such path string appears as: <run_XXXXXXXX>/sub_directory1/sub_directory2/…etc…/<monitoring_histogram>. All such monitoring histograms come from 2 dimensional TH2 (see Root Class Reference [62]) class histogram files.
- x - Feature x is the pseudorapidity η, given by the monitoring histogram.
- y - Feature y is the azimuthal angle ϕ, given by the monitoring histogram.
- occ - is the occupancy for any given (x, y) coordinate in the monitoring histogram.
- occ_0to1 - is the occupancy as above, but scaled and not normalized using the minmax formula given below:

$$\vec{o}_{scaled} = \frac{\vec{o} - \vec{o}_{min}}{\vec{o}_{max} - \vec{o}_{min}} \tag{11}$$

where $\vec{o}$ is the vector occupancy values at the corresponding (x, y) coordinate, $\vec{o}_{max}$ is the maximum occupancy value for this monitoring histogram, $\vec{o}_{min}$ is the minimum occupancy value for this monitoring histogram, and $\vec{o}_{scaled}$ is the occupancy vector containing all scaled values. All occupancies scaled in this manner will have values between 0 and 1. If the minimum and maximum values are equivalent, they are set to their original value $\vec{o}_{scaled} = 0$. Defined in this way, it emphasizes using vector operations to complete the scaling task for large vectors of data.

- occ_robust - is a normalization method commonly used that is robust against the presence of scalers. The robust scaler formula is given below:

$$\vec{o}_{\boldsymbol{robust}} = \vec{o} - \frac{\boldsymbol{med}(\vec{o})}{\boldsymbol{r_{iqr}}} \tag{12}$$

where r_{iqr} is the interquartile range such that

$$r_{iqr} = q_3(\vec{o}) - q_1(\vec{o}) \tag{13}$$

and $med(\vec{o})$ is the median of the occupancy vector $\vec{o}$.

- occ_zscore - is a commonly used normalization method in the machine learning realm. The occ_zscore formula given below:

$$\vec{o}_{zscore} = \frac{o - \mu(\vec{o})}{\sigma(\vec{o})} \tag{14}$$

where o is the occupancy vector at the given (x, y) coordinate, $\mu(\vec{o})$ is the average occupancy for the monitoring histogram, $\sigma(\vec{o})$ is the standard deviation for the monitoring histogram, and $\vec{o}_{zscore}$ is the vector of normalized occupancies. All occupancies normalized in this manner have no specific range they are normalized to. It is useful for determining how far away values are from the standard deviation.

- ftag_id -is the meta information feature that identifies what specific ftag this collection of run numbers and monitoring histograms come from with respect to the order in the database file. The database file contains a total of 87 tables, 1 table for each ftag. Therefore, there are a total of 87 such ftag_ids that ftag_id can list in this meta feature. (The ftags in the database file are ordered such that the first table is table 0 and the last table is table 86 for the 87 tables). Some datasets

contain less than this number as it is the unique number of histograms in that dataset that is reported, not the ftag_id.

- hist_id - is the meta information feature that identifies which of the monitoring histograms who all come from a single ftag id, the data point refers to. (This together with the path feature and ftag id allows one to identify all such data points from a single monitoring histogram).
- hist_type - is the meta information feature that identifies which of the 18 types of monitoring histograms this histogram is (refer to the upcoming paragraphs for more information on these 18 histogram types). This feature also allows one to experiment using input data from individual histograms.
- quality - is the label feature that identifies if the datapoint came from the monitoring histogram in the specified run file or if that datapoint was generated as an anomaly as part of the anomaly generation process. This feature will be used derived from anomalies that have been manually generated (details ahead) for evaluation of unsupervised learning tasks as well as training and testing of supervised learning tasks.

In order to properly interpret and explain the model results, it is necessary to plot the monitoring histograms prior to and following predictions. Therefore, the meta features were added to facilitate this extra level of detail.

3.4.2 Specific Details on Constructing the Final Datasets

An overview of the dataset construction process following identification of research objectives can be best described by the flowchart below:

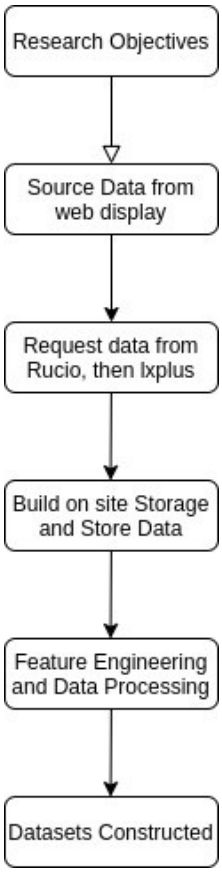

Figure 23: Diagram of the dataset construction process.

Research Objectives

In order to construct the dataset, first the research objectives were clearly defined. These have been discussed in detail in Chapter 1 with associated scopes and limitations in Chapter 2.

Source data from Web Display

From the online web display, the most recent runs are displayed that have physics_Main streams, copy that textual meta information from the display, and get the information needed to make the run requests from Rucio by pushing that information through a python processing script.

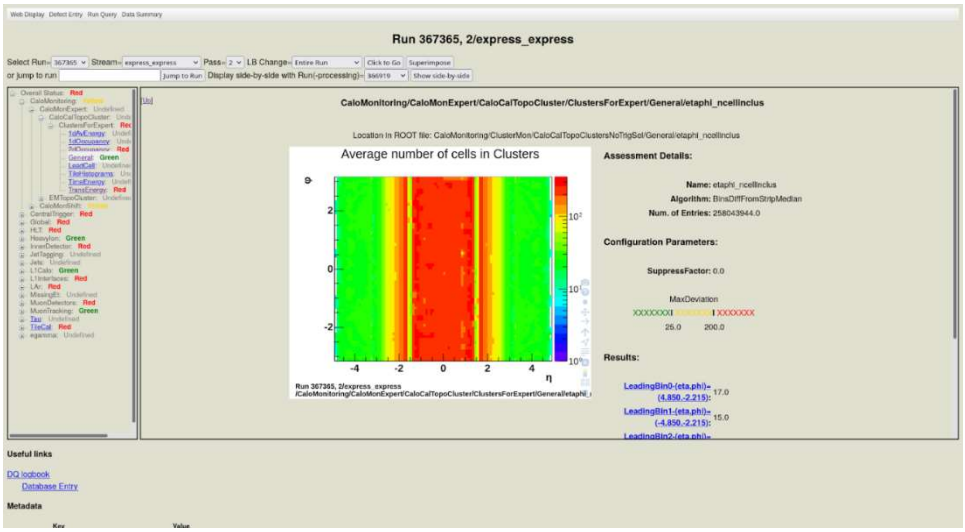

Figure 24: The Data Quality Monitoring Display (DQMD) or web display for short is where the data and its meta information are sourced [63].

```
run_366142/ f1027_h331 data18_hi
CaloMonitoring/ClusterMon/CaloCalTopoClustersNoTrigSel/2d_Rates/m_clus_etaphi_Et_thresh2
CaloMonitoring/ClusterMon/CaloCalTopoClustersNoTrigSel/2d_Rates/m_clus_etaphi_Et_thresh3
CaloMonitoring/ClusterMon/CaloCalTopoClustersNoTrigSel/General/etaphi_ncellinclus
CaloMonitoring/ClusterMon/CaloCalTopoClustersNoTrigSel/TransEnergy/etaphi_thresh_avgEt_0
CaloMonitoring/ClusterMon/LArClusterEMNoTrigSel/2d_Rates/m_clus_etaphi_Et_thresh1
CaloMonitoring/ClusterMon/LArClusterEMNoTrigSel/2d_Rates/m_clus_etaphi_Et_thresh2
CaloMonitoring/ClusterMon/LArClusterEMNoTrigSel/2d_Rates/m_clus_etaphi_Et_thresh3
CaloMonitoring/ClusterMon/LArClusterEMNoTrigSel/General/etaphi_ncellinclus

run_366092/ f1027_h331
CaloMonitoring/ClusterMon/CaloCalTopoClustersNoTrigSel/2d_Rates/m_clus_etaphi_Et_thresh2
CaloMonitoring/ClusterMon/CaloCalTopoClustersNoTrigSel/2d_Rates/m_clus_etaphi_Et_thresh3
CaloMonitoring/ClusterMon/CaloCalTopoClustersNoTrigSel/General/etaphi_ncellinclus
CaloMonitoring/ClusterMon/LArClusterEMNoTrigSel/2d_Rates/m_clus_etaphi_Et_thresh1
CaloMonitoring/ClusterMon/LArClusterEMNoTrigSel/2d_Rates/m_clus_etaphi_Et_thresh2
CaloMonitoring/ClusterMon/LArClusterEMNoTrigSel/2d_Rates/m_clus_etaphi_Et_thresh3
CaloMonitoring/ClusterMon/LArClusterEMNoTrigSel/General/etaphi_ncellinclus

run_366029/ f1027_h331
CaloMonitoring/ClusterMon/CaloCalTopoClustersNoTrigSel/2d_Rates/m_clus_etaphi_Et_thresh2
CaloMonitoring/ClusterMon/CaloCalTopoClustersNoTrigSel/2d_Rates/m_clus_etaphi_Et_thresh3
CaloMonitoring/ClusterMon/CaloCalTopoClustersNoTrigSel/General/etaphi_ncellinclus
CaloMonitoring/ClusterMon/LArClusterEMNoTrigSel/2d_Rates/m_clus_etaphi_Et_thresh0
CaloMonitoring/ClusterMon/LArClusterEMNoTrigSel/2d_Rates/m_clus_etaphi_Et_thresh1
CaloMonitoring/ClusterMon/LArClusterEMNoTrigSel/2d_Rates/m_clus_etaphi_Et_thresh2
CaloMonitoring/ClusterMon/LArClusterEMNoTrigSel/2d_Rates/m_clus_etaphi_Et_thresh3
CaloMonitoring/ClusterMon/LArClusterEMNoTrigSel/General/etaphi_ncellinclus

run_366011/ f1027_h331
CaloMonitoring/ClusterMon/CaloCalTopoClustersNoTrigSel/2d_Rates/m_clus_etaphi_Et_thresh2
CaloMonitoring/ClusterMon/CaloCalTopoClustersNoTrigSel/2d_Rates/m_clus_etaphi_Et_thresh3
CaloMonitoring/ClusterMon/CaloCalTopoClustersNoTrigSel/General/etaphi_ncellinclus
CaloMonitoring/ClusterMon/CaloCalTopoClustersNoTrigSel/TransEnergy/etaphi_thresh_avgEt_0
CaloMonitoring/ClusterMon/LArClusterEMNoTrigSel/2d_Rates/m_clus_etaphi_Et_thresh0
CaloMonitoring/ClusterMon/LArClusterEMNoTrigSel/2d_Rates/m_clus_etaphi_Et_thresh1
CaloMonitoring/ClusterMon/LArClusterEMNoTrigSel/2d_Rates/m_clus_etaphi_Et_thresh2
CaloMonitoring/ClusterMon/LArClusterEMNoTrigSel/2d_Rates/m_clus_etaphi_Et_thresh3
CaloMonitoring/ClusterMon/LArClusterEMNoTrigSel/General/etaphi_ncellinclus

run_365932/ f1027_h331
CaloMonitoring/ClusterMon/CaloCalTopoClustersNoTrigSel/2d_Rates/m_clus_etaphi_Et_thresh2
```

Figure 25: A visualization of the output of the processing scripts as data is gathered.

Request Data from Rucio, then LXPLUS

The data is stored in not readily available format; therefore, requests must be made prior to being able to locally download it. Having processed the web display

information, the script outputs properly formatted requests that can simply be pasted into rucio. Rucio processes those requests, and when all of them get the OK status, they can be downloaded from lxplus. To download them from lxplus, first login to lxplus with GRID credentials (information on how to do this can be found within the ATLAS twiki online reference material), run the “setup ATLAS” command, the “voms init” type command that is suggested by the prompt, then run the “lsetup rucio” command. Having done this, the script previously outputted download commands as well as formatted requests. Finally, paste those download commands into the command window, and the commands will run and download the associated run file, line by line (depending on the number of downloads it can take some time).

```
data18_13TeV:data18_13TeV.00348251.physics_Main.merge.HIST.f920_h295
data18_13TeV:data18_13TeV.00364292.physics_Main.merge.HIST.f1002_h327
data18_13TeV:data18_13TeV.00364214.physics_Main.merge.HIST.f1002_h327
data18_13TeV:data18_13TeV.00364160.physics_Main.merge.HIST.f1002_h327
data18_13TeV:data18_13TeV.00364098.physics_Main.merge.HIST.f1002_h327
data18_13TeV:data18_13TeV.00364076.physics_Main.merge.HIST.f1002_h327
data18_13TeV:data18_13TeV.00364030.physics_Main.merge.HIST.f1002_h327
data18_13TeV:data18_13TeV.00363979.physics_Main.merge.HIST.f1002_h327
data18_13TeV:data18_13TeV.00363947.physics_Main.merge.HIST.f1002_h327
data18_13TeV:data18_13TeV.00363910.physics_Main.merge.HIST.f1002_h327
data18_13TeV:data18_13TeV.00363830.physics_Main.merge.HIST.f1002_h327
data18_13TeV:data18_13TeV.00363738.physics_Main.merge.HIST.f1002_h327
data18_13TeV:data18_13TeV.00363710.physics_Main.merge.HIST.f1001_h327
data18_13TeV:data18_13TeV.00363664.physics_Main.merge.HIST.f1001_h327
data18_13TeV:data18_13TeV.00348251.express_express.merge.HIST.f920_h295
data18_13TeV:data18_13TeV.00363664.express_express.merge.HIST.f1001_h327
data18_13TeV:data18_13TeV.00363710.express_express.merge.HIST.f1001_h327
data18_13TeV:data18_13TeV.00363738.express_express.merge.HIST.f1002_h327
data18_13TeV:data18_13TeV.00363830.express_express.merge.HIST.f1002_h327
data18_13TeV:data18_13TeV.00363910.express_express.merge.HIST.f1002_h327
data18_13TeV:data18_13TeV.00363947.express_express.merge.HIST.f1002_h327
data18_13TeV:data18_13TeV.00363979.express_express.merge.HIST.f1002_h327
data18_13TeV:data18_13TeV.00364030.express_express.merge.HIST.f1002_h327
data18_13TeV:data18_13TeV.00364076.express_express.merge.HIST.f1002_h327
data18_13TeV:data18_13TeV.00364098.express_express.merge.HIST.f1002_h327
data18_13TeV:data18_13TeV.00364160.express_express.merge.HIST.f1002_h327
data18_13TeV:data18_13TeV.00364214.express_express.merge.HIST.f1002_h327
data18_13TeV:data18_13TeV.00364292.express_express.merge.HIST.f1002_h327
```

Figure 26: Further processing from scripts. This format of the data is for the requests that are input to Rucio.

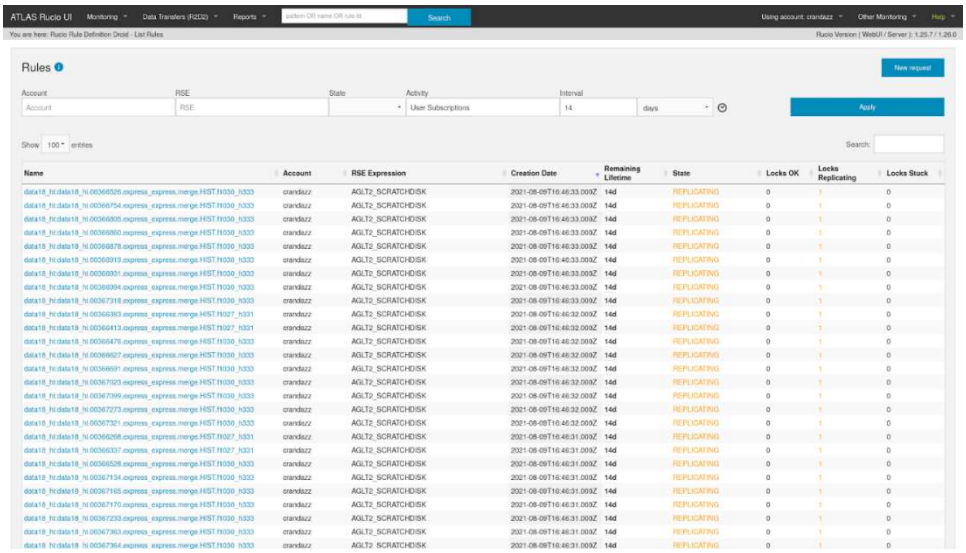

Figure 27: A visualization of all the requests having been made on Rucio prior to downloading the files.

```
rucio download data18_13TeV:data18_13TeV.00348251.physics_Main.merge.HIST.f920_h295
rucio download data18_13TeV:data18_13TeV.00364292.physics_Main.merge.HIST.f1002_h327
rucio download data18_13TeV:data18_13TeV.00364214.physics_Main.merge.HIST.f1002_h327
rucio download data18_13TeV:data18_13TeV.00364160.physics_Main.merge.HIST.f1002_h327
rucio download data18_13TeV:data18_13TeV.00364098.physics_Main.merge.HIST.f1002_h327
rucio download data18_13TeV:data18_13TeV.00364076.physics_Main.merge.HIST.f1002_h327
rucio download data18_13TeV:data18_13TeV.00364030.physics_Main.merge.HIST.f1002_h327
rucio download data18_13TeV:data18_13TeV.00363979.physics_Main.merge.HIST.f1002_h327
rucio download data18_13TeV:data18_13TeV.00363947.physics_Main.merge.HIST.f1002_h327
rucio download data18_13TeV:data18_13TeV.00363910.physics_Main.merge.HIST.f1002_h327
rucio download data18_13TeV:data18_13TeV.00363830.physics_Main.merge.HIST.f1002_h327
rucio download data18_13TeV:data18_13TeV.00363738.physics_Main.merge.HIST.f1002_h327
rucio download data18_13TeV:data18_13TeV.00363710.physics_Main.merge.HIST.f1001_h327
rucio download data18_13TeV:data18_13TeV.00363664.physics_Main.merge.HIST.f1001_h327
rucio download data18_13TeV:data18_13TeV.00348251.express_express.merge.HIST.f920_h295
rucio download data18_13TeV:data18_13TeV.00363664.express_express.merge.HIST.f1001_h327
rucio download data18_13TeV:data18_13TeV.00363710.express_express.merge.HIST.f1001_h327
rucio download data18_13TeV:data18_13TeV.00363738.express_express.merge.HIST.f1002_h327
rucio download data18_13TeV:data18_13TeV.00363830.express_express.merge.HIST.f1002_h327
rucio download data18_13TeV:data18_13TeV.00363910.express_express.merge.HIST.f1002_h327
rucio download data18_13TeV:data18_13TeV.00363947.express_express.merge.HIST.f1002_h327
rucio download data18_13TeV:data18_13TeV.00363979.express_express.merge.HIST.f1002_h327
rucio download data18_13TeV:data18_13TeV.00364030.express_express.merge.HIST.f1002_h327
rucio download data18_13TeV:data18_13TeV.00364076.express_express.merge.HIST.f1002_h327
rucio download data18_13TeV:data18_13TeV.00364098.express_express.merge.HIST.f1002_h327
rucio download data18_13TeV:data18_13TeV.00364160.express_express.merge.HIST.f1002_h327
rucio download data18_13TeV:data18_13TeV.00364214.express_express.merge.HIST.f1002_h327
rucio download data18_13TeV:data18_13TeV.00364292.express_express.merge.HIST.f1002_h327
```

Figure 28: Further processing from the scripts, the same as **Figure 26**, but with the download commands for lxplus generated.

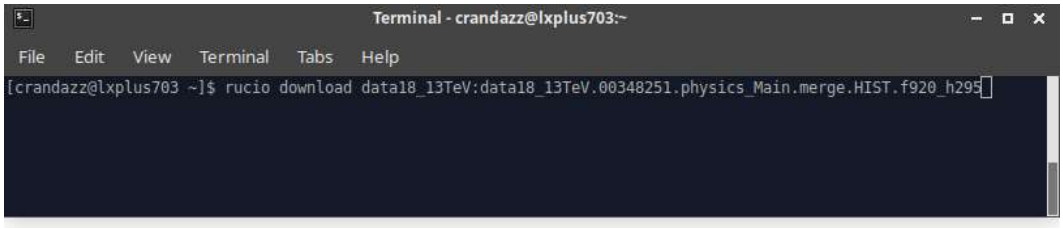

Figure 29: An example of one of the command lines from the previous **Figure 28** going into lxplus. These are pasted all at once and the command prompt knows to download them one at a time.

Build on Site Storage and Store Data

There are a number of ways to handle this, but for speed and scalability, python's implementation of SQL called SQLAlchemy was chosen to build the storage for this system. The current system stores all the data in a local file, but this can be migrated to a server database with the built infrastructure relatively easy. From there, SQL based data requests would improve the time required to run future experiments. Having built this infrastructure, the data gets put into a workable format and stored in this type of database file. Some testing with the database has shown some volatility during storage.

Figure 30: A visualization of the data that has been downloaded straight to the user area in CERNbox or SWAN.

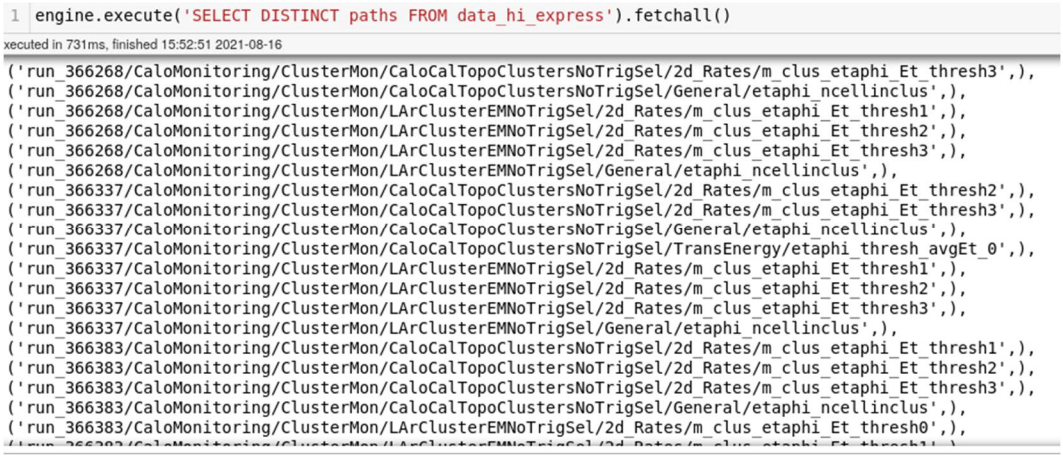

Figure 31: A visualization of the processed data that was downloaded and has been moved to the SQLAlchemy database.

Feature Engineering and Data Processing

Prior to the final cleaning and processing of the data, the features of interest must be engineered from the stored data. The final decision on what features to use was a

combination of time and space limitations with regards to the machine learning, suggestions from previous authors' work such as the feasibility to use geometric image features rather than a parameter per pixel approach, various conversations with my advisor who is a subject matter expert with regard to this matter, and exploration of the initial data during and following storage. Having divined these features of interest, the process of any necessary cleaning and formatting of the data is handled, followed by generation of those features of interest. Further details on this step will be included below.

Following the description of relevant features, next the specifics of data that was requested and included in this work will be discussed. Since the tools generated from this work are designed for the upcoming 2022 Run, the input data will be the most recent previous Run data as far back as 2018. Additionally, the data18_13TeV run files were requested to be focused on for this work so these were selected to construct the main dataset.

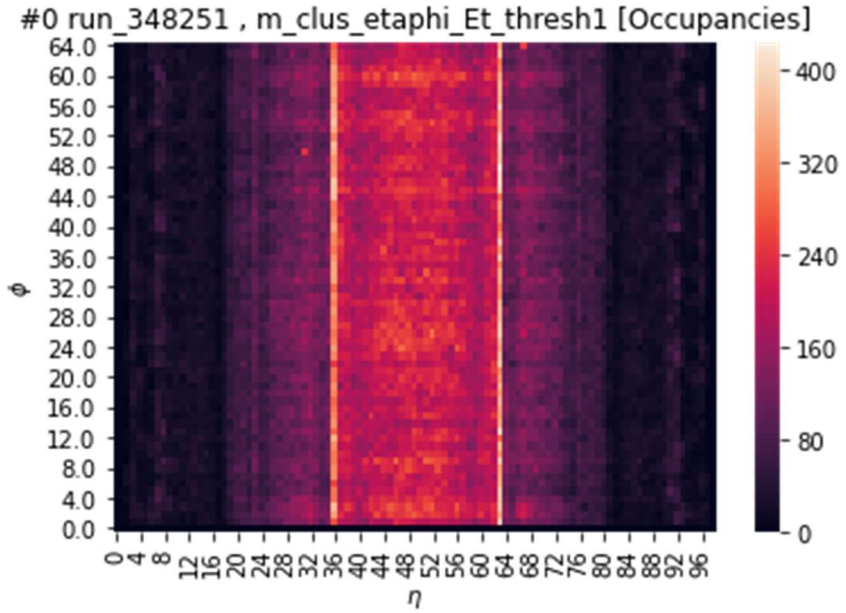

Figure 32: Example monitoring histogram heatmap from the dataset.

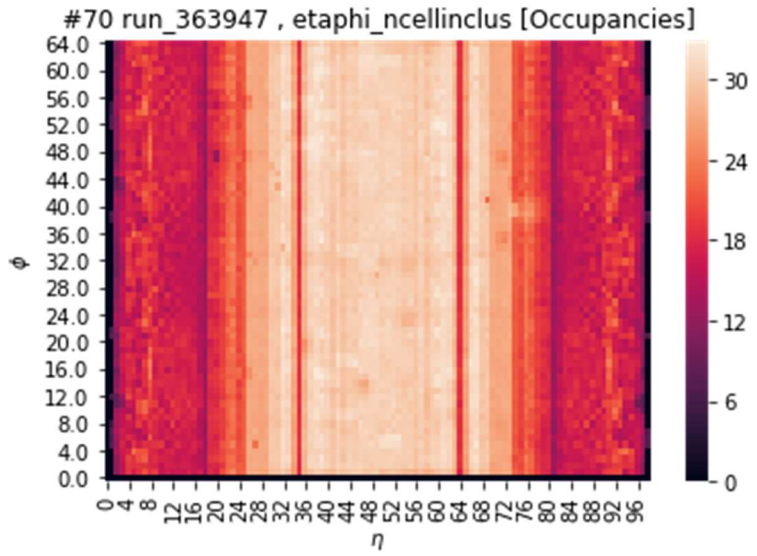

Figure 33: Another example of a monitoring histogram from the dataset.

The downloaded files were stored in a SQLAlchemy database file in tabular format such that each table in the database contained a collection of run files specific to a single ftag. This ftag was used as the table name for each such collection. To date, the database contains 6408 data18_13TeV physics_Main stream monitoring histograms worth of data. Initially, the CaloCombined, Egamma, Jets, MissingEt, and Tau folders were coded into the processing system, but the data stored in the database now only reflect the folders whose location in the run folder are of the 18 identified monitoring histogram types of interest from the DQ web display (see the following section that details the 18 monitoring histogram types).

In order to simplify the process of storing the raw TH2 data (as well as simplifying future data processing and machine learning training) in the *SQLAlchemy* database, a script was developed that converts an entire chosen monitoring histogram in a run file to a *pandas* data frame. During this time, the geographic image data as well as the pathing information for the histogram are extracted to the data frame. Using this script

and pandas's DataFrames (called data frame in this work), all relevant monitoring histograms in all relevant run files were inserted into the database file with the extracted information.

Up to this point, all steps prior to the final data processing step in the flowchart are demonstrated in the Jupyter Notebook whose filename is "Step1-DataGatherAndInitialPrep–2-15-2.ipynb". The final step on the flowchart is composed of a collection of 4 internal steps. Those 4 internal steps are the following (starting with step 2 to follow the naming convention).

- Step2("Step2DropDuplicates-outputMainDfs–2-21-22.ipynb"): In this step is where all the data cleaning would take place. For the purposes of this project, only removing duplicate entries were necessary due to loose use of the database insertion methods.
- Step3("AnomalyGenCode–2-22-2.ipynb"): Anomaly generation have been and will be a key technique in upcoming experiments first because of the testing done with the supervised CNN for labels, and later for evaluation and validation of the unsupervised approach as will be explained in the evaluation section.
- Step4("Step4NormalizeOccByHist–2-9-22.ipynb": From the previous step, the main dataset will branch off into an anomalous and non-anomalous dataset mostly due to preference of format. Therefore, a sister file named similarly will normalize the non-anomalous dataset in this step. The normalizations are responsible for generating the normalized features that were previously described.
- Step5("FullHistSplit–2-23-22.ipynb"): Having Cleaned, generated anomalous data points for, and added normalized features to the main datasets, all that is left

is to split the datasets into their respective training and testing sets for the machine learning and experimentation. To do this, a custom train test split method was used rather than the industry standard scikit-learn method as a data frame full of various histogram datapoints would leave one with histograms that holes of missing data. Instead, the custom split function looks at the dataset on a histogram level, randomly selects those for either the train or test set based on the desired percentage of histograms to hold out, and moves all the data points as they are, individually selected with all their respective data points intact. From here, the flowchart and data construction are complete.

	paths	x	y	occ	ftag_id	hist_type	hist_id	quality
0	run_363664/CaloMonitoring/ClusterMon/CaloCalTo...	0	0	3872943.0	0	0	0	1
1	run_363664/CaloMonitoring/ClusterMon/CaloCalTo...	0	1	4551316.0	0	0	0	1
2	run_363664/CaloMonitoring/ClusterMon/CaloCalTo...	0	2	0.0	0	0	0	0
3	run_363664/CaloMonitoring/ClusterMon/CaloCalTo...	0	3	3610718.0	0	0	0	1
4	run_363664/CaloMonitoring/ClusterMon/CaloCalTo...	0	4	0.0	0	0	0	0
...	...	...	...	...	...	...	...	...
41235475	run_360402/CaloMonitoring/ClusterMon/LArCluste...	98	60	0.0	84	17	233	0
41235476	run_360402/CaloMonitoring/ClusterMon/LArCluste...	98	61	0.0	84	17	233	0
41235477	run_360402/CaloMonitoring/ClusterMon/LArCluste...	98	62	0.0	84	17	233	0
41235478	run_360402/CaloMonitoring/ClusterMon/LArCluste...	98	63	78.0	84	17	233	1
41235479	run_360402/CaloMonitoring/ClusterMon/LArCluste...	98	64	0.0	84	17	233	0

41235480 rows × 8 columns

Figure 34: The full unsplit dataset prior to feature engineering and after anomaly generation. 9,146,559 of the 41,235,480 are generated anomalies (~22%).

A variation of the final datasets shows what the shape would look like if it were formatted it in the manner that is typical of image based neural network inputs (pixel by pixel). This would lead to a dimension of 6408 input neurons with 4165 samples. This highlights the concerns of overfitting of the model on the histogram level. If all inputs

and data points were independent of each other, this would be less of an issue.

Regardless, either approach has been shown to yield results [64].

	hist_metadata	hist_type	thresh_val	occ_at_0_0	occ_at_0_1	occ_at_0_2	occ_at_0_3	occ_at_0_4	occ_at_0_5	occ_at_0_6	...	occ_at_98_55	occ_at_98_56	occ_at_98_57	occ_at_98_58	occ_at_98_59	occ_at
0	('run_363664', 'data18_13TeV', 'f1001_h327', '...	CaloMonitoring/ClusterMon/CaloCalTopoClustersN...	undecided	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	6.125000	0.0	0.0	0.0	0.0	
1	('run_363664', 'data18_13TeV', 'f1001_h327', '...	CaloMonitoring/ClusterMon/CaloCalTopoClustersN...	undecided	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.652109	0.0	0.0	0.0	0.0	
2	('run_363664', 'data18_13TeV', 'f1001_h327', '...	CaloMonitoring/ClusterMon/CaloCalTopoClustersN...	undecided	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.000000	0.0	0.0	0.0	0.0	
3	('run_363664', 'data18_13TeV', 'f1001_h327', '...	CaloMonitoring/ClusterMon/CaloCalTopoClustersN...	undecided	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.000000	0.0	0.0	0.0	0.0	
4	('run_363664', 'data18_13TeV', 'f1001_h327', '...	CaloMonitoring/ClusterMon/CaloCalTopoClustersN...	undecided	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.000000	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
4160	('run_367365', 'data18_hi', 'f1030_h333', 'exp...	CaloMonitoring/ClusterMon/CaloCalTopoClustersN...	undecided	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.000000	0.0	0.0	0.0	0.0	
4161	('run_367365', 'data18_hi', 'f1030_h333', 'exp...	CaloMonitoring/ClusterMon/CaloCalTopoClustersN...	undecided	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.000000	0.0	0.0	0.0	0.0	
4162	('run_367365', 'data18_hi', 'f1030_h333', 'exp...	CaloMonitoring/ClusterMon/CaloCalTopoClustersN...	undecided	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.000000	0.0	0.0	0.0	0.0	
4163	('run_367365', 'data18_hi', 'f1030_h333', 'exp...	CaloMonitoring/ClusterMon/CaloCalTopoClustersN...	undecided	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.000000	0.0	0.0	0.0	0.0	
4164	('run_367365', 'data18_hi', 'f1030_h333', 'exp...	CaloMonitoring/ClusterMon/CaloCalTopoClustersN...	undecided	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.000000	0.0	0.0	0.0	0.0	

4165 rows × 6438 columns

Figure 35: The horizontal dataset contains 4165 histograms, 83 ftags, all compiled together in 4165 datapoints, one per histogram, and 6438 features. Most features are for each coordinate in the 65x99 histogram plane, and a few metadata features as well. This dataset was constructed for example to highlight its limitations.

3.5 Exploration of Reconstruction Error in Anomaly Detection

3.5.1 Does Reconstruction Error Correlate with Anomalous Data?

This is an important question to ask. First, the only way to know this for certain is to use a dataset with known labels. Next, after training the model, the reconstruction error by datapoint similar to **Figure 36** can be studied.

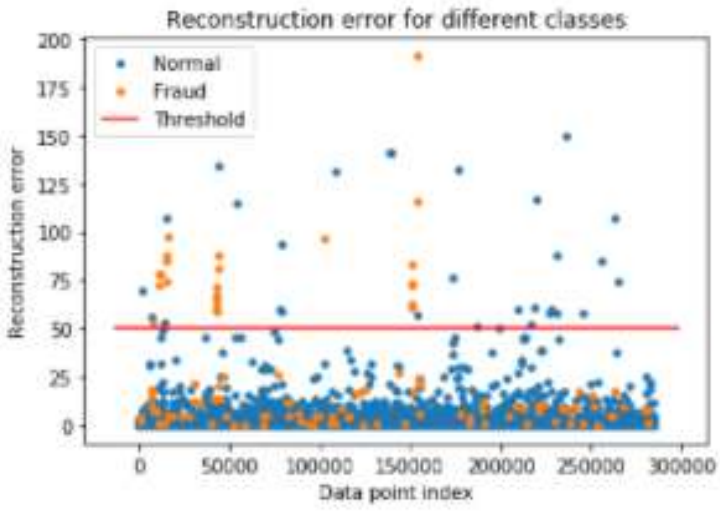

Figure 36: The reconstruction error per data point of some arbitrary anomaly detection algorithm. Anomalies ground truth labels are given by color, and the threshold in this case is a single linear classification boundary in this space. It appears that the reconstruction error, in this case, has more examples that cannot fit inside a linear classification threshold (more fraud data mixed between normal data than fraud data classified by a threshold).

The issue here is that an attempt to impose a linear classification rule is being made (threshold in this case) to what appears to be non-linearly separable data. It is nonlinearly separable by class specifically because, with this example's setup, there is not a linear correlation between the reconstruction error and the normal or fraudulent labels. Further verification of this by plotting the class labels vs reconstruction error can be made to see if the classes for lower values of reconstruction error tend towards class 0 or not and if the classes for higher values of reconstruction error tend towards 1 or not. An even better representation that would allow one to verify a correlation here would plot the class prediction probability against the reconstruction error if it is available.

If such a correlation between class and reconstruction error is found, when the above plot was generated, the majority of anomalous datapoints above such a threshold value would be seen and the data could be linearly separable in this context.

The question is revisited, does reconstruction error correlate with anomalous data? The short answer is it should and the long answer, as above, is it depends. In both cases, this is extremely important to construct and verify for an anomaly detection model to achieve its intended purpose. The next obvious question then is how can the reconstruction error be made correlate with anomalous data points?

3.5.2 How to Make Reconstruction Error Correlate with Anomalous Data?

Typically, it is assumed that reconstruction error correlates with anomalous datapoints, but recent work is showing this may no longer be the case [66, 67, 68].

1. Train on mostly non-anomalous data points. If an autoencoder is trained to learn how to reconstruct input that it sees, then if it is trained on anomalous datapoints, it will learn to reconstruct anomalous and non-anomalous data points. If it is trained on only normal data points, then given an input with anomalous data, it will reconstruct what the non-anomalous set would look like and therefore cause the reconstruction error to correlate high with anomalous data points. The issue with this is that it becomes a supervised task (and to some degree all unsupervised classification of anomalous data becomes supervised in the literal sense such as when imposing a threshold parameter to the model).
2. Use a different algorithm.
3. Use a different architecture for current algorithm.
4. Add relevant features to the dataset.
5. Use a different set of hyperparameters.

Thus, without going the fully labeled dataset route, a series of experiments can be conducted, preprocessing techniques can be applied, and transformations can be made to this approach in order to achieve a reconstruction error that correlates well with anomalous data. Therefore, in order to discover the optimal approach, a series of experiments will be conducted to see the effects the system's experiments will have on the established controls. In an effort to separate anomalous data from non-anomalous data using reconstruction error, a few main assumptions related to the issue is reviewed:

1. The often-made assumption is that anomalous data points will have a high reconstruction error [66, year 2019]. That is, reconstruction error should correlate with outlier probability.
2. It has been found that this is not always the case. To mitigate this issue, many approaches now assume that a semi-supervised approach where training on non-anomalous data only yields high reconstruction error for anomalous data points [66, year 2020].
3. Even then, the assumption in 2 may be poor because it has been shown that when trained on non-anomalous data, the autoencoder is still capable of reconstructing anomalous data. A proposed solution to this is generating anomalous data from non-anomalous data, thereby giving a more concrete boundary for the autoencoder to construct a distribution whose reconstruction error can more reliably predict anomalous data points [67, year 2021].

Exploration of this idea of how to make reconstruction error correlate with outlier probability in the coming experiments will be explored. It is from the previous reasoning, as well as for evaluation purposes, the data generation method was chosen as the

preferred method. From that starting point, how to make this correlation for the final solution will be discovered, or an equally useful solution to achieve the work's objectives will be found.

3.6 Exploratory Analysis of the Training Set and Testing Set

The unsplit dataset contains over 41,000,000 data points consisting of 6408 histograms whose axes are a full length of 99 in x and 65 in y. The train and test sets have file names "x_train_df2.csv" and "x_test_df2.csv" respectively. The split datasets can be viewed in the **Figure 37** and **Figure 38**:

	paths	x	y	occ	ftag_id	hist_type	hist_id	quality	occ_0to1	occ_zscore	occ_robust
0	run_363664/CaloMonitoring/ClusterMon/CaloCalTo...	0	0	0.0	0	2	2	0	0.000000	-0.163700	-0.449285
1	run_363664/CaloMonitoring/ClusterMon/CaloCalTo...	0	1	0.0	0	2	2	0	0.000000	-0.163700	-0.449285
2	run_363664/CaloMonitoring/ClusterMon/CaloCalTo...	0	2	2454.0	0	2	2	1	0.032756	-0.161290	0.021687
3	run_363664/CaloMonitoring/ClusterMon/CaloCalTo...	0	3	0.0	0	2	2	0	0.000000	-0.163700	-0.449285
4	run_363664/CaloMonitoring/ClusterMon/CaloCalTo...	0	4	0.0	0	2	2	0	0.000000	-0.163700	-0.449285
...	...	...	...	...	...	...	...	...	...	...	...
27593275	run_360402/CaloMonitoring/ClusterMon/LArCluste...	98	60	0.0	84	17	233	0	0.000000	-0.090706	-0.170740
27593276	run_360402/CaloMonitoring/ClusterMon/LArCluste...	98	61	0.0	84	17	233	0	0.000000	-0.090706	-0.170740
27593277	run_360402/CaloMonitoring/ClusterMon/LArCluste...	98	62	0.0	84	17	233	0	0.000000	-0.090706	-0.170740
27593278	run_360402/CaloMonitoring/ClusterMon/LArCluste...	98	63	78.0	84	17	233	1	0.006434	-0.090692	0.036157
27593279	run_360402/CaloMonitoring/ClusterMon/LArCluste...	98	64	0.0	84	17	233	0	0.000000	-0.090706	-0.170740

27593280 rows × 11 columns

Figure 37: Anomalous Training Dataset (pMtrain_a, file name "x_train_df2.csv") contains 4288 histograms, 83 unique ftags, all compiled together in 27,593,280 samples and 11 features (x as η, y as ϕ, and occ as occupancy) as well as 2 metadata features.

	paths	x	y	occ	ftag_id	hist_type	hist_id	quality	occ_0to1	occ_zscore	occ_robust
0	run_363664/CaloMonitoring/ClusterMon/CaloCalTo...	0	0	3872943.0	0	0	0	1	0.084320	3.641077	1.564232
1	run_363664/CaloMonitoring/ClusterMon/CaloCalTo...	0	1	4551316.0	0	0	0	1	0.099089	4.307510	1.903885
2	run_363664/CaloMonitoring/ClusterMon/CaloCalTo...	0	2	0.0	0	0	0	0	0.000000	-0.163700	-0.374900
3	run_363664/CaloMonitoring/ClusterMon/CaloCalTo...	0	3	3610718.0	0	0	0	1	0.078611	3.383467	1.432940
4	run_363664/CaloMonitoring/ClusterMon/CaloCalTo...	0	4	0.0	0	0	0	0	0.000000	-0.163700	-0.374900
...	...	...	...	...	...	...	...	...	...	...	...
13642195	run_360402/CaloMonitoring/ClusterMon/LArCluste...	98	60	0.0	84	15	231	0	0.000000	-0.090706	-0.920560
13642196	run_360402/CaloMonitoring/ClusterMon/LArCluste...	98	61	0.0	84	15	231	0	0.000000	-0.090706	-0.920560
13642197	run_360402/CaloMonitoring/ClusterMon/LArCluste...	98	62	0.0	84	15	231	0	0.000000	-0.090706	-0.920560
13642198	run_360402/CaloMonitoring/ClusterMon/LArCluste...	98	63	0.0	84	15	231	0	0.000000	-0.090706	-0.920560
13642199	run_360402/CaloMonitoring/ClusterMon/LArCluste...	98	64	0.0	84	15	231	0	0.000000	-0.090706	-0.920560

13642200 rows × 11 columns

Figure 38: Anomalous Test Dataset (pMtest_a, file name "x_test_df2.csv") contains 2120 histograms, 83 unique ftags, all compiled together in 13,642,200 samples and 11 features (x as η, y as ϕ, and occ as occupancy) as well as 2 metadata features.

From **Figure 37** and **Figure 38**, it can be seen that the datasets have been split such that ⅔ of the data will be for training while ⅓ of the data will be for testing. There is no perfect standard for this split, but others often split datasets between 80/20 and 70/30. The advantage of having a larger training set is a potentially better trained model, but the advantage of a larger testing set is better evaluation on a model's generalizability (Note: the datasets are split histogram-wise rather than data pointwise to allow model interpretation and explainability). Both datasets exist in .csv file format for easy importing and exporting throughout the data processing and machine learning parts of this work.

These datasets contain 18 unique types of monitoring histograms. The histogram types can be identified by their pathing information as given by the run file structure. The 18 types of histograms used are as follows:

1. CaloMonitoring/ClusterMon/CaloCalTopoClustersNoTrigSel/2d_Rates/m_clus_etaphi_Et_thresh0,

2. CaloMonitoring/ClusterMon/CaloCalTopoClustersNoTrigSel/2d_Rates/m_clus_etaphi_Et_thresh1,
3. CaloMonitoring/ClusterMon/CaloCalTopoClustersNoTrigSel/2d_Rates/m_clus_etaphi_Et_thresh2,
4. CaloMonitoring/ClusterMon/CaloCalTopoClustersNoTrigSel/2d_Rates/m_clus_etaphi_Et_thresh3,
5. CaloMonitoring/ClusterMon/CaloCalTopoClustersNoTrigSel/General/etaphi_ncellinclus,
6. CaloMonitoring/ClusterMon/CaloCalTopoClustersNoTrigSel/TransEnergy/etaphi_thresh_avgEt_0,
7. CaloMonitoring/ClusterMon/CaloCalTopoClustersNoTrigSel/TransEnergy/etaphi_thresh_avgEt_1,
8. CaloMonitoring/ClusterMon/CaloCalTopoClustersNoTrigSel/TransEnergy/etaphi_thresh_avgEt_2,
9. CaloMonitoring/ClusterMon/CaloCalTopoClustersNoTrigSel/TransEnergy/etaphi_thresh_avgEt_3,
10. CaloMonitoring/ClusterMon/LArClusterEMNoTrigSel/2d_Rates/m_clus_etaphi_Et_thresh0,
11. CaloMonitoring/ClusterMon/LArClusterEMNoTrigSel/2d_Rates/m_clus_etaphi_Et_thresh1,
12. CaloMonitoring/ClusterMon/LArClusterEMNoTrigSel/2d_Rates/m_clus_etaphi_Et_thresh2,

13. CaloMonitoring/ClusterMon/LArClusterEMNoTrigSel/2d_Rates/m_clus_etaphi_Et_thresh3,
14. CaloMonitoring/ClusterMon/LArClusterEMNoTrigSel/General/etaphi_ncellinclus,
15. CaloMonitoring/ClusterMon/LArClusterEMNoTrigSel/TransEnergy/etaphi_thresh_avgEt_0,
16. CaloMonitoring/ClusterMon/LArClusterEMNoTrigSel/TransEnergy/etaphi_thresh_avgEt_1,
17. CaloMonitoring/ClusterMon/LArClusterEMNoTrigSel/TransEnergy/etaphi_thresh_avgEt_2,
18. CaloMonitoring/ClusterMon/LArClusterEMNoTrigSel/TransEnergy/etaphi_thresh_avgEt_3

(NOTE: The hist_type value in the data frame starts with hist_type = 0 for the 1st type of histogram above and hist_type = 17 for the 18th histogram type above).

Following the exploration of the dataset and reviewing the literature in the domain of anomaly detection and anomaly detection in high energy physics, an unsupervised autoencoder based approach seems most appropriate as primarily unlabeled, clean data is available. In addition to this, rather than relying entirely on a neural network to achieve the anomaly detection objective, there also appears to be an opportunity to exploit the type of pattern that presents itself in the datasets when anomalies are present. Thus, a DBScan clustering algorithm on the back end of the autoencoder model will be included. In section 3.10, further detail on the reasoning behind this choice is explained. The other works in these domains have thoroughly explored autoencoder approaches, but none have

currently been found to have also tried explicitly exploiting this pattern in the data in combination with autoencoders. In an effort to not simplify the results, a standard autoencoder architecture whose parameters will be described further in section 3.7.1 will be used. This should simplify the comparison of this approach to other typical methods on the front end [68].

As a first step, calibration of the systems for training the model will be made by constructing a complete autoencoder for some experiments on normalization of the dataset. It has been observed in other work that the choice in normalization scheme depends on unique dataset [85].

3.7 Experiments

3.7.1 Parameter Explorations

For all experiments excluding the complete autoencoder, the best setup will be determined and compared to other results. Variations in architecture and layers will be made, activation functions, input features and targets, as well as validation features and targets. For conciseness, the details of every setup will not be included. Some of the following parameters will be explored during the experimental phase of the work:

- physics_Main stream monitoring histograms only (identified by "pM" in the filenames)
- pMtrain_nonAnom –vs– pMtest_nonAnom
- pMtrain_Anom –vs– pMtrain_nonAnom
- Hot spots –vs– cold spots
- 4208 histograms training set
- 2120 histograms in the test set

- AE Model is fit to “pM” dataset that does not include generated anomalous data points, but testing and validation utilizes the anomalous version of the dataset (i.e. “pMtrain_a”)
- Hyperparameter tuning
- Architecture and Layers
    - 3 input layer nodes: (η, ϕ, occupancy)
    - Various encoder, latent, and decoder layer nodes will be tested to determine the best reconstruction error performance
    - 3 output layer nodes: (η, ϕ, occupancy) - as the eta and phi coordinates will be reconstructed, but are constant, the eta and phi coordinates are mapped to the reconstructed occupancy values only
- Optimizer - Adam (adaptive momentum) - the Adam optimizer is one of the best general-purpose optimization algorithm for deep learning applications currently available
- Loss function - MSE (mean squared error) - Rather than other approaches that use binary_crossentropy with sigmoid activation outputs in this space, the MSE has been chosen as the reconstruction error in the same manner as other researchers
- Activation function - testing will include the standard Relu as well as some linear activation functions. Reconstruction will occur over the entire histogram and final classification will occur later, the output activation will not be mapped to a sigmoid or binary function.

In summary, the experiments will include these parameter combinations, the previously mentioned explorations, the autoencoder approach, and an exploration on the outlier decision function and final classification methods in the coming sections.

3.7.2 The Autoencoder Neural Network (AE) Approach

In neural network and deep neural network approaches using images and image like inputs, it is common to use each pixel of the image as an input to the neural network. However, while there is plenty of data available in high energy physics, much of which is currently unlabeled. Therefore, an image the size of the monitoring histograms (99x65) would require 6,435 input parameters with only 4200 data points for the model if the process was continued in this way which could lead to overfitting. Fortunately, initial assessments with the complete autoencoder demonstrated that no unusual behavior was present in the vertical dataset that was constructed utilizing coordinates as features. This will allow an advantage in terms of neural network parameters as well as resilience against overfitting the model. This configuration can restrict the autoencoder's ability to learn complex information, but as will be shown, the autoencoder will easily reconstruct the histogram with an expected level of loss and provides a reconstruction error result that is essential for the anomaly detection process.

3.7.3 A Note on Reproducibility

Deep Learning Neural Networks and Neural Networks in general require extra consideration to reproduce results because they are stochastic by design. They are designed with many pieces that are randomized in order to generate the best results [69]. However, this does not mean a well performing model can be made from the random information that is embedded into the training.

One way to reproduce the results is relatively straight forward. Once the model is trained, all the random information that was used to initialize and process the model during training is statically set in the final outputs, the model and its weights can be saved, and this information can be loaded to predict with the same evaluation results as it was when it was originally saved. More complex methods to reproduce the results can be found in the literature [70, 71, 72, 73].

For reproducing the model, a detailed set of instructions is given on dataset construction as well as the parameter configuration and architecture configuration for the model. There will also be included the saved and trained file for the model itself that can be rapidly loaded and tested, and the final form of the code used in producing the results will be uploaded to a GitHub repository.

3.8 Outlier Decision Function, Probability Function, and Model Evaluation

The "outlier decision function" is an important step necessary to move from manual classification of outliers to automatic classification of outliers. In [74], the authors suggest that when moving from manual classification to automatic classification, a threshold parameter must be set. It is not hard to jump from this requirement to applying statistical techniques such as the 3-sigma rule, the mean absolute deviation, or the interquartile range. However, the authors also offer other work that has used machine learning methods to calibrate for this requirement. It is also believed that this threshold requirement is of key importance to run an automatic anomaly detection system. The developed approach will exploit patterns in the data that suggest outlier behavior to calibrate this threshold. In other work, authors have utilized the unsupervised machine learning technique of clustering to exploit distance-based patterns in data to automatically

classify certain data as outliers [64]. The unsupervised clustering algorithm DBSCAN (see 2.3.3) will be used as the decision function to take advantage of the kinds of patterns this detector data exhibits on the occupancy matrices in order to make the final classification decision. Again, outliers can occur in hot or cold spots, strips, or layers. Experts identify these outliers based on the context of nearby data points in the same strips, nearby strips, or the entire map. Nearby, in terms of machine learning, suggests a distanced based or nearest neighbor approach, and experiments in this work will prove the validity and usefulness of this approach to anomaly detection in this domain. In general, sources such as [65] loosely suggest that the autoencoder reconstruction error correlates with anomalous data points, but the initial findings have not been able to confirm this. Thus, an assumption of outlier probability can be viewed as follows:

$$P(OL) \propto RE, \tag{15}$$

$$P(OL) \propto d, \tag{16}$$

Where $P(OL)$ is the outlier probability, RE is the autoencoder reconstruction error, and d is the Euclidean distance to its nearest neighbor. Equation 1 is a finding in various sources on anomaly detection that utilize autoencoders. The second relation will be verified by results from the anomaly detection approach in this work.

The threshold parameter is one of if not the most important parameters chosen for an unsupervised learning application. The reason for this, specifically with regards to anomaly detection, is because regardless of how effective an algorithm is, in order to automate the final classification of outlier or inlier, this parameter must be set to make that final decision. In [75], the authors briefly discuss the importance of threshold selection and go further by offering a technique based on their work that is more

sophisticated than the standard. The standard method is the so-called “elbow method” where a scientist would select the threshold by eye based on a subjective choice of slope change on the PR curve. There appears to be very little information available about threshold selection beyond this simple method in the literature, but the solution offered by the authors provides insight into this as well as what the elusive outlier probability might look. They suggest using the Bayes Decision Rule to offer a more robust way of selecting threshold. The rule is given below:

$$(\lambda_{21} - \lambda_{11})P(w_1|x) > (\lambda_{12} - \lambda_{22})P(w_2|x) \quad (17)$$

Where λ_{ij} is the cost of misclassifying w_j as w_i and $P(w_i|x)$ is the probability that data point x is class i. The subject of interest for this work is the class probability, however the best explanation of how this relates to the problem is that when the probability of class 1 equals the probability of class 2, the threshold is 0.5, a 50-50 chance, and setting the loss such that the probability of the inlier class is greater, a greater false positive rate is incurred whereas setting the probability of outlier class greater a greater false negative rate is incurred. This mathematically defines the advantages and disadvantages of a higher or lower threshold value. Typically, the threshold parameter exists in industry standard packages such as scikit-learn for python.

Few sources found in the literature clarify threshold parameter settings of results for entirely unsupervised approaches. Rather, the threshold is commonly calibrated by selecting a result from a ROC curve of supervised models that return results from labeled datasets [76].

3.9 Evaluation, Interpretability, and Explainability

3.9.1 Evaluation

In the previous section, the connection between threshold selection and common evaluation methods was explained. While this method (ROC) is reasonable to quantify the performance of a model, it forces a requirement of labeled data. In this unsupervised approach, the technique suggested in [77] is drawn from, where the predictions are ranked according to the anomaly score and then iteratively a threshold is applied from the first to the last rank, resulting in N tuple values for true positive and false negative rates that can be used to generate the ROC and AUC curves. In [78], the authors interpret the AUC in the anomaly detection domain as the probability an anomaly detection algorithm will assign a randomly chosen normal instance a lower score than a randomly chosen anomalous instance. That is, higher AUC scores suggest higher ability of the algorithm's predicted scores to represent outlier probability. To facilitate this evaluation, the precision_recall_curve, auc_curve, and the classification_report classes from the sklearn library will be used.

Since each prediction of the algorithm only tests outliers in a single histogram and to get a better overview of the performance, the model will be run over the calibrated histogram for several runs and/or ftags. It will be determined if calibration needs to be run on a histogram-by-histogram basis by comparing model results between different types of histograms. Each calibration takes time to find optimal threshold values but inter histogram comparisons are quicker using the same threshold configurations. The classification report metrics are explained in **Figure 39**.

Besides the models themselves, it is recommended that researchers ask the following questions to identify key differences of works in this problem space when comparing results:

- Is the data labeled? If so, how much of it is labeled and what is the quality of the labels? (Have they been generated? From a model? From experts?)
- With regards to model architecture, how specifically was the model constructed?
- Do labels identify outlier/inlier values with respect to occupancy values of specific coordinates, entire strips, entire histograms, or entire sections of the detector?

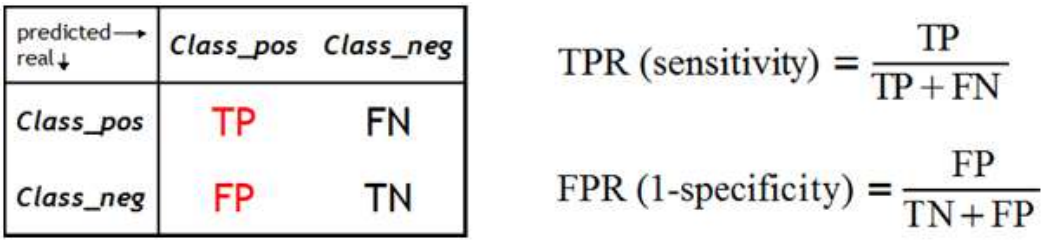

Figure 39: A diagram of how the classification matrix returns classification results. The true positive rate (TPR) and false positive rate (FPR) formulas are given based on the true positive (inliers classified as inliers, TP), false negative (inliers classified as outliers), false positive (outliers classified as inliers), and true negative (outliers classified as outlieres).

3.10 Dashboard and Deployment

Individuals who followed the initial tracking of the COVID-19 pandemic spread have likely been exposed to a dashboard using epidemiological information. A dashboard system allows for live monitoring and forecasting (depending on the requirements) for the user to benefit from the achieved objectives. They can include analytic as well as predictive information [79]. This type of system has been identified as a valuable, first iteration tool for individual users and, perhaps in the future, shifters in fulfillment of the

previously mentioned objectives. The dashboard is deployed using Dash by Plotly (see 3.2) with all its requirements using Docker (see 3.2) to allow consistent, uninterrupted use of the solution. Future iterations of this type of solution could be integrated on the main ATLAS server end rather than operating as a local tool.

CHAPTER 4

RESULTS

The results of these experiments rapidly converged on the patterns apparent in each part of the system. The first part being the resultant patterns observed and leveraged by the autoencoder for use in the second part of the system used to make the final classification of the outlier vs. inlier. The following information provides the results of those experiments.

4.1 Validation Dashboard Initial Phase

The validation dashboard currently runs as a locally hosted server but can be deployed similarly to an externally available online server. The current format provides examples of what the dashboard can do for users to make the final decision on features that will be implemented in upcoming versions as well as the current core functionality of an efficient plot to determine differences of all relevant histograms between 2 specific runs. This plot is of chi squared differences between 2 runs vs the histogram itself - one chi squared difference per histogram type. Hovering over any datapoint gives all necessary meta information including the exact chi squared value and specific histogram and run information. Future updates will include the anomaly detection system. The validation dashboard in its current state can be seen in **Figure 40**.

Figure 40: A visualization of the Dashboard's current version. One analysis plot for each of the histogram types TH1, TH2, and TProfile are provided. On the "Distributions" tab are found the distributions for the respective distributions.

4.2 Initial Experiment Results

Initial experiments began implementing a CNN approach, but the limitation of having very few ground truth labels was quickly apparent. ATLAS's defect database is not currently in the state to provide detailed enough information about ground truth labels that would be used for this type of application. Following this realization, The decision was made to approach the problem from an unsupervised point of view. Therefore, the results of these tests provided information as to the nature of limitations going into this research. Later, it was discovered that the information reported in chapter 3 recommended manual generation of anomalies during the design of the evaluation of the experiments. By that time, the infrastructure for the unsupervised approach was already developed and there appears to be no advantage to reviewing the CNN approach. This also means this research will provide an advantage in a different scenario for data quality

assessment since it can be implemented without the need for ground truth labels. The model will be tested and calibrated similar to the initial experiments before moving on.

4.3 Results of Complete Autoencoder for Model Construction, Validation, and Choice of Dataset

After preparations for the unsupervised experiments were ready, a basic check was made to make sure the setup behaves as expected. For this, the identity function was modelled with minimally invasive parameter setups. Some variation was expected due to leaving some parameters unmodified, but it was expected for this setup to yield near identity like reconstruction.

Parameter configuration for training:

- Adam optimizer
- MSE loss
- Input layer 3 neurons, output layer 3 neurons
- 10 epochs
- 32 batch size

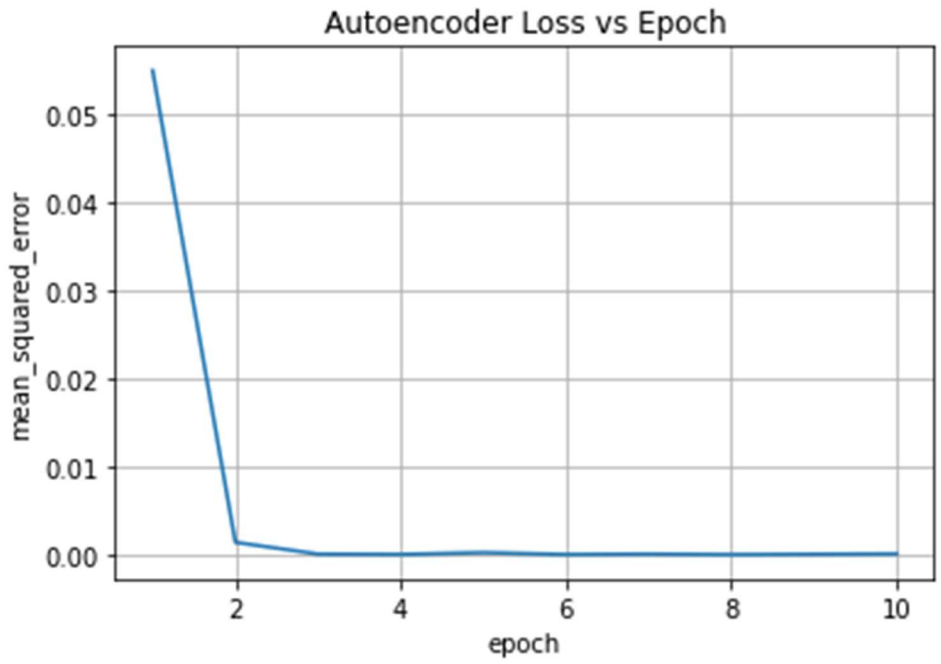

Figure 41: Reconstruction error of identity function on the dataset.

The total reconstruction error in the training set was about 0.01% and the total reconstruction in the test set was about 0.31% using this configuration.

Of the 10 runs, the run with the lowest reconstruction errors were selected in order to reconstruct the input as close as possible. In the 10 runs, the reconstruction error was consistently close to what would be expected from this minimally invasive setup. It can be further optimized to achieve less error, but this is an acceptable benchmark.

4.4 Results of ANN and DBScan Approach

The following information follows the anomaly detection system that was focused on following initial experiments.

The reconstruction error applied as the scoring function for the autoencoder can be seen in **Figure 42** and **Figure 43**. Applying a high end threshold to these values should yield the hot spot outliers as expected, but the reconstruction error with the MSE

has an effect on the low end inputs that will exploited in the upcoming results to be useful in identifying outliers containing cold spots as well.

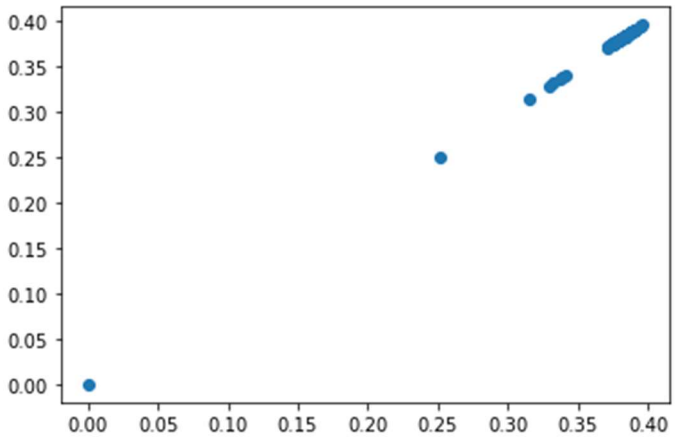

Figure 42: This plot is of "occ_0to1" vs "occ_0to1" simply for demonstration. This is an input strip from an example histogram. Assuming these inputs, see **Figure 43** for the affects of applying the Autoencoder based MSE.

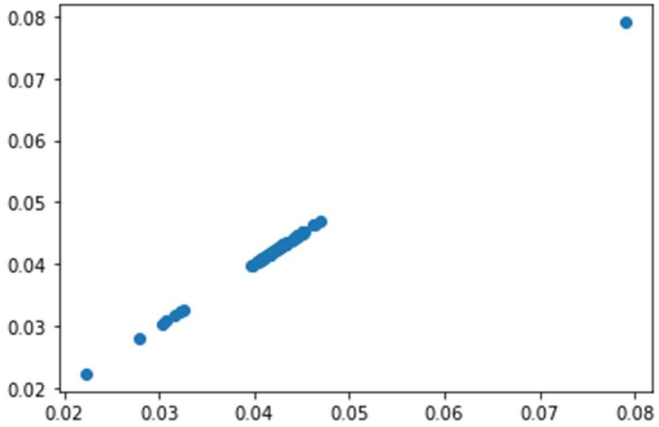

Figure 43: The Autoencoder applies the Mean Squared Reconstruction Error(MSE) as the anomaly scoring function and the resulting values applied from the inputs of Figure 42 change as shown here. The y and x axis are of the same MSE result data, visualized in this manner rather than on a single axis as an aesthetic choice.

Figure 44 shows an example histogram that the model is applied to. **Figure 45** is the heatmap of reconstructed values of those inputs through the Autoencoder part of the

model. **Figure 46** shows an alternative 2d visualization of the example histogram input values. **Figure 47** shows that same alternate 2d visualization, but with the reconstructed occupancy values instead. **Figure 48** shows the heatmap of reconstruction error values calculated from the MSE for this part of the model.

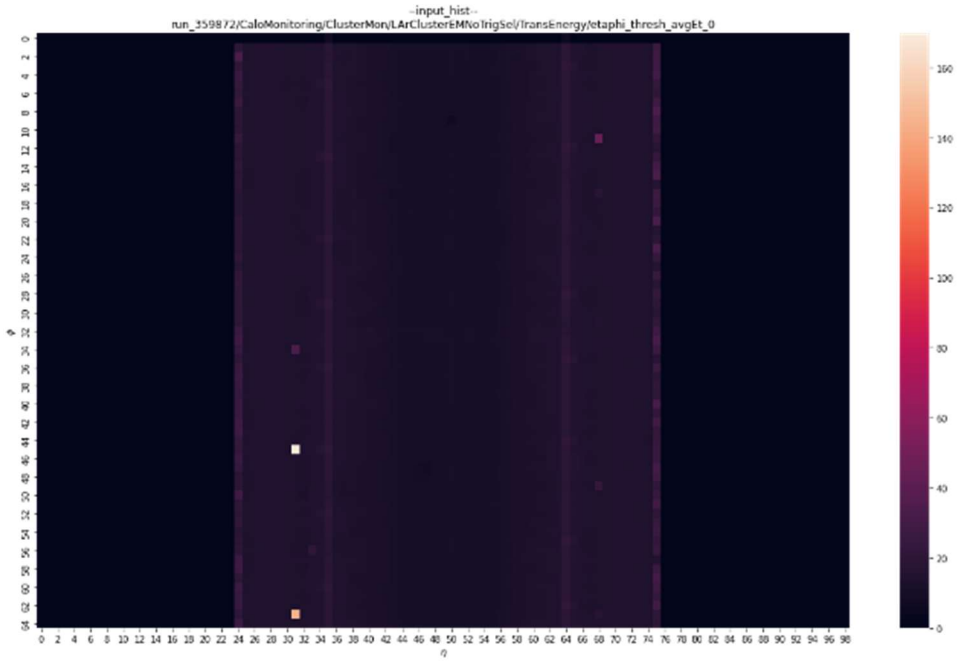

Figure 44: An example input histogram that will be reconstructed with the Autoencoder. The original data of the example histogram is shown here. Compare with **Figure 45** to estimate results.

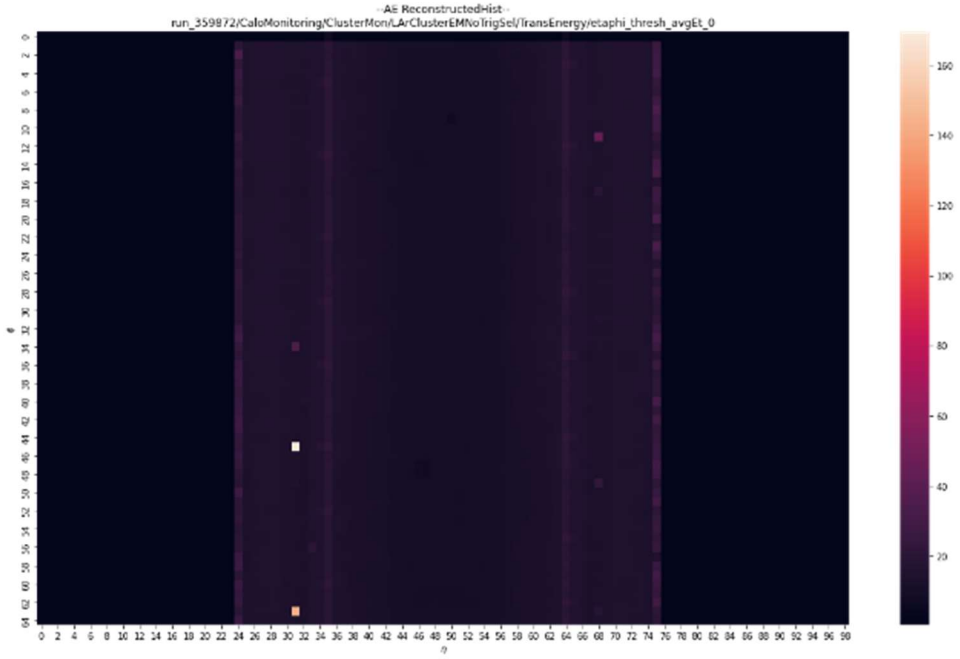

Figure 45: A reconstruction of the example histogram from **Figure 44**. The reconstructed values are shown here. Compare with **Figure 44** to estimate results.

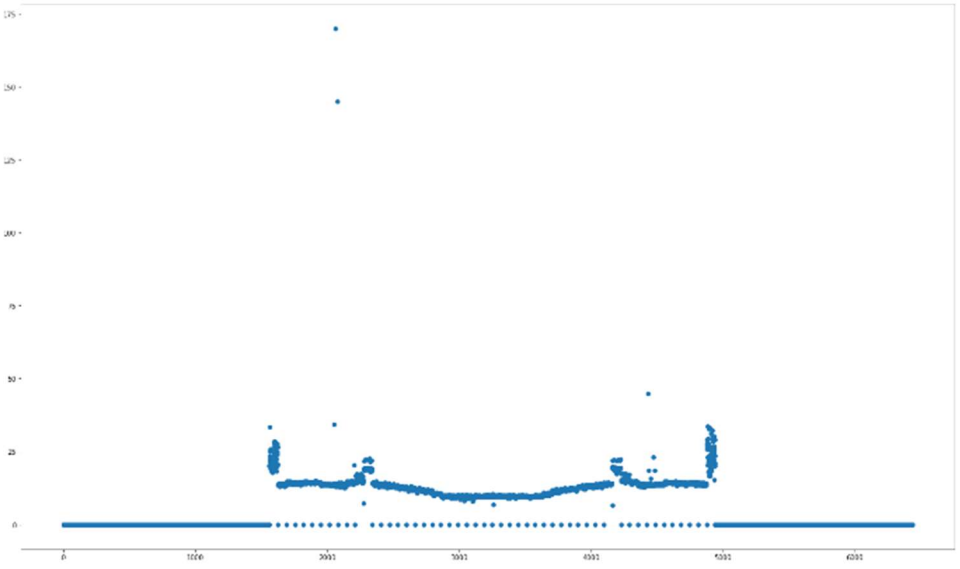

Figure 46: A more detailed look at the data values in the reconstruction gives insight into the effect the autoencoder has. The horizontal axis is the coordinate index of each coordinate in the histogram. The vertical axis is the occupancy at that coordinate. Comparing this and **Figure 47** clearly demonstrates the autoencoder's reconstruction error spreading out clusters of data points. Original data given here in this figure.

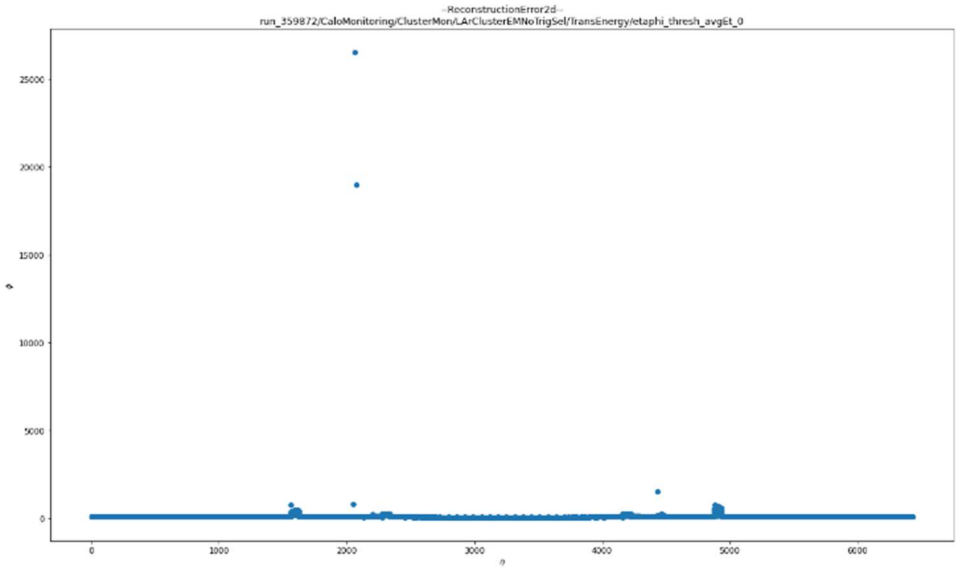

Figure 47: A more detailed look at the data values in the reconstruction gives insight into the effect the autoencoder has. The horizontal axis is the coordinate index of each coordinate in the histogram. The vertical axis is the reconstructed occupancy value. Comparing this with **Figure 46** clearly demonstrates the autoencoder's reconstruction error spreading out clusters of data points. Reconstruction values given here in **Figure 47**.

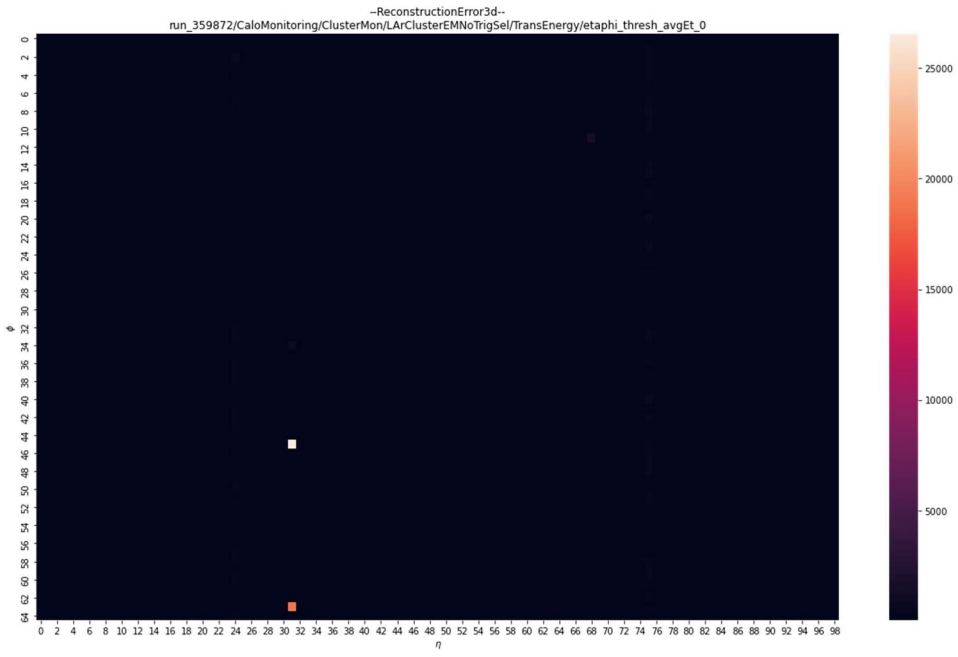

Figure 48: Reconstruction error relates to the function between the original and predicted occupancy values. Here, a visualization is given of the reconstruction error rather than the predicted values or reconstructed values. A high reconstruction error for the 2 highest occupancy coordinates in the histogram can be seen.

This next part of the experiment's results shows a technique that was used to easily see where large clusters of data points exist for the distribution and the smallest clusters describe anomalous clusters. The clusters are identified with DBScan. This identification process has 3 types of identification modes: “global”, “stripwise”, and “zonewise”. The “global” mode looks at the clusters that appear in this manner and picks out the anomalies that are of concern with respect to the entire histogram’s occupancy values. The “stripwise” mode returns anomalies detected in these clusters for each eta strip and is useful for picking out anomalous behavior relative to individual strips. The geometry of the detector creates a flat normal distribution along each eta strip such that these are ideal targets for anomaly detection. Finally, the “zonewise” mode returns the anomalous data points strip by strip as before, but this time taking additionally taking into consideration the adjacent strip occupancy values to determine if the cluster in question is anomalous or not. The results of these modes are depicted in the following plots. Note: These plots are of the reconstruction error values from the above plots **Figure 45** and the input occupancy values **Figure 44.** The choice of plotting these axes is mostly an aesthetic choice so the clusters can be better seen, and if desired, could be presented in 1d as **Figure 49**.

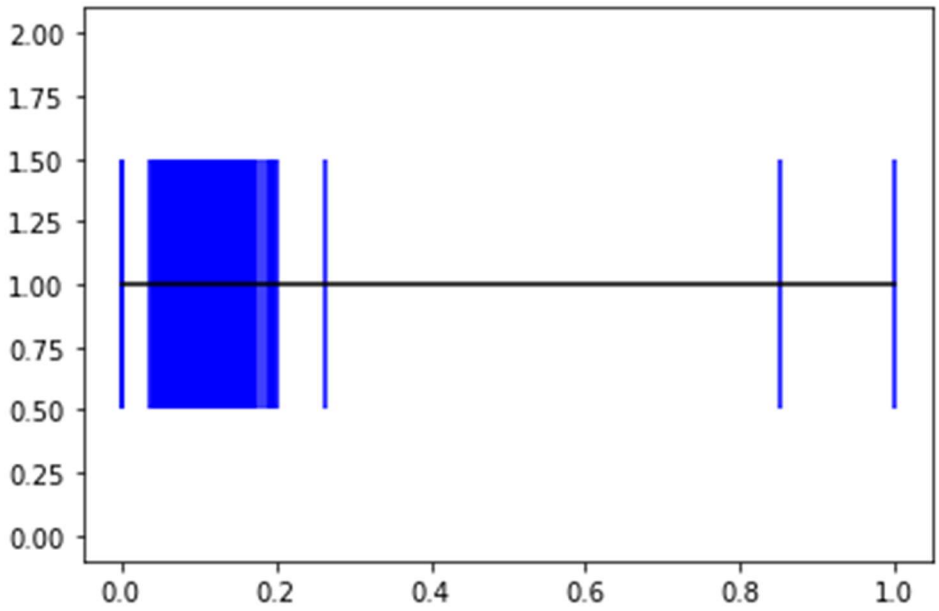

Figure 49: 1d version of the suspect anomalous clusters according to the reconstruction error. The y axis is meaningless here as it is the height of the blue location lines. The x axis is the normalized reconstructed occupancy values from the example histogram.

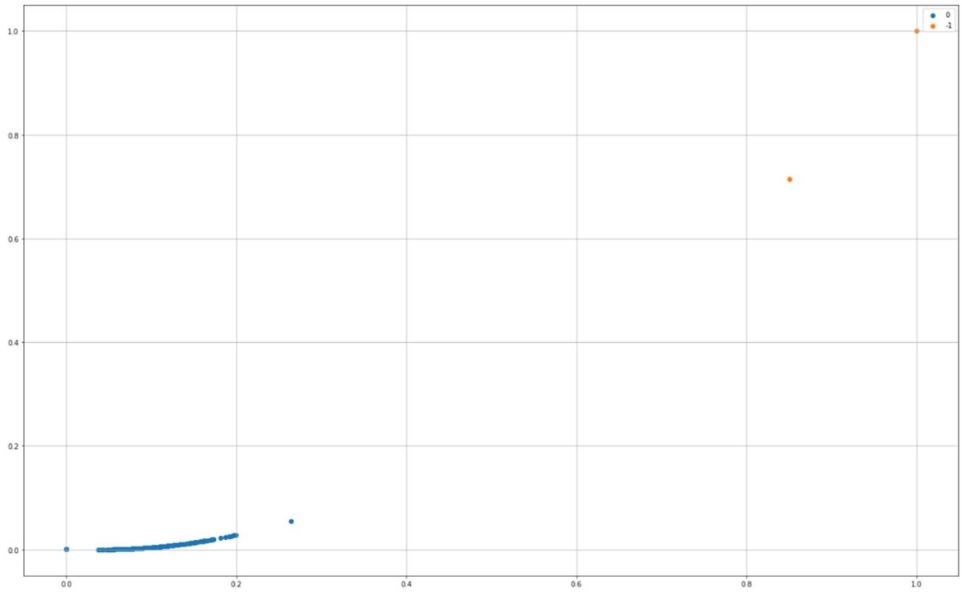

Figure 50: Plot demonstrating how the system classifies anomalies in the global mode. The system outputs the anomaly details by coordinate as [(η, ϕ, normalized occupancy)]. Here, it returns [(31, 45, 1.0), (31, 63, 0.85125035)].

In the “stripwise” mode, since there are 99 strips to each histogram, it would add no extra value to include so many resulting histograms, but several cases are included here to demonstrate how this mode classifies the clusters and anomalies. In this example histogram, there are several strips of entirely 0 value occupancies. These return no anomalies as expected and are skipped for convenience

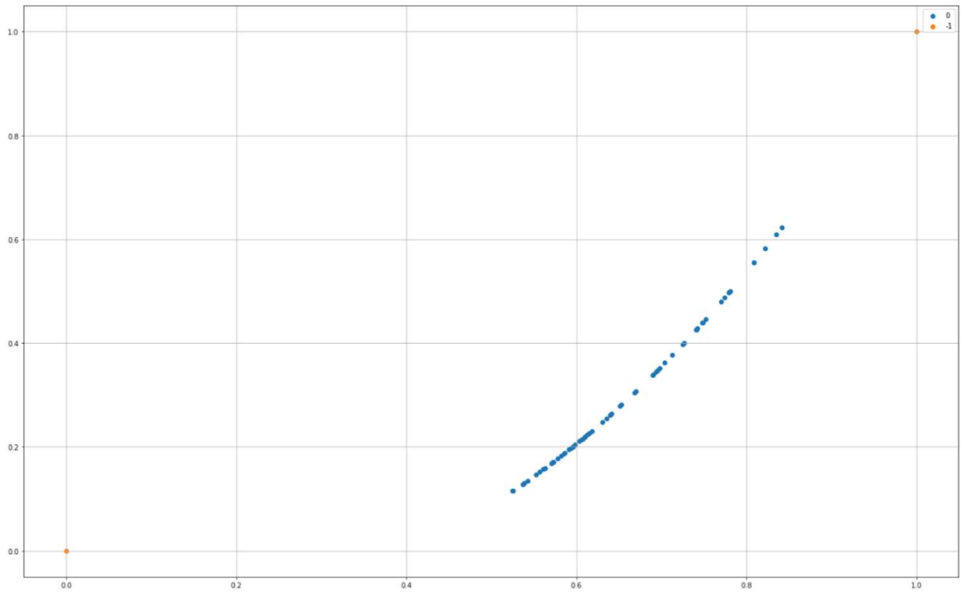

Figure 51: Plot of “stripwise” identified anomalous clusters using the developed system. The returned anomalous coordinates in the identified cluster are [(24, 0, 0.0), (24, 2, 1.0)]. In the same format as before mentioned. According to the η value, this shows s

In cases where naturally occurring clusters that are still in the distribution occur, several clusters are identified. It is then the assumption is enforced that anomalous values occur in least frequency and declare a “minpts” parameter to indicate the maximum number of values that can occur in a cluster for it to be deemed anomalous. As this threshold gets tuned, it affects the overall performance of the model and is a target for optimization as will be described later. For this example histogram, the “minpts” value is

set to a maximum of 10 occupancy values. The strip in **Figure 52** indicates this behavior of the system.

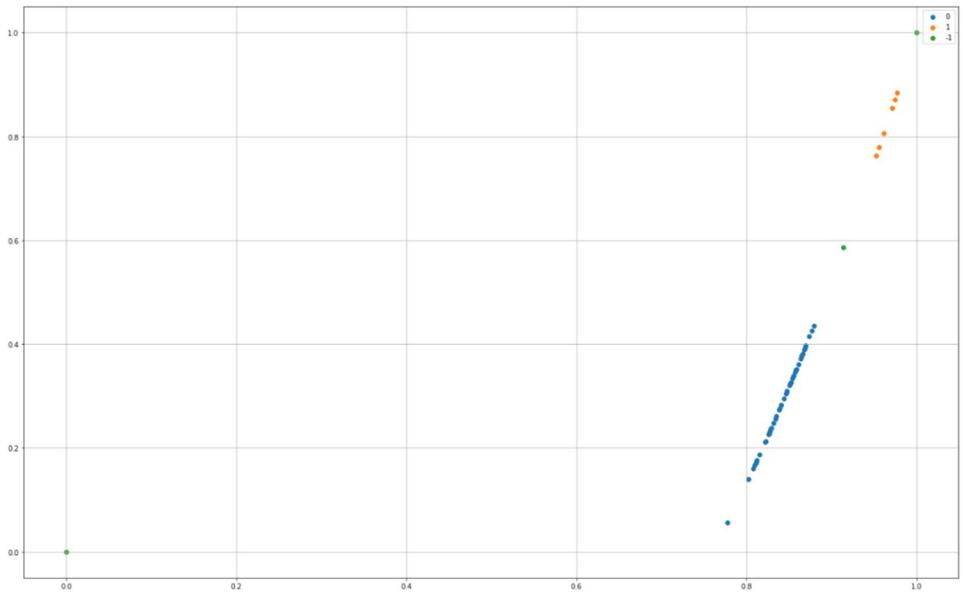

Figure 52: Plot where multiple clusters are identified in strip 34 of the example histogram. The clusters with the lowest frequency are the green and yellow clusters as shown. The system reports anomalous values for clusters that contain less than 10 values. In this case those are indeed the green and yellow clusters. Yellow cluster anomalies: [(34, 5, 0.96149087), (34, 22, 0.9559057), (34, 29, 0.97456884), (34, 36, 0.9712373), (34, 45, 0.97742224), (34, 52, 0.95238775)]. Green cluster anomalies: [(34, 0, 0.0), (34, 13, 1.0), (34, 60, 0.9141569)].

The “zonewise” mode has no additional special characteristics besides considering the adjacent strip’s occupancy values for clustering. Focusing again on strip 24, **Figure 53** demonstrates the difference between the detection settings during the anomaly detection. **Figure 53** will differ in its look due to inclusion of strips 23, 24, and 25’s occupancy values.

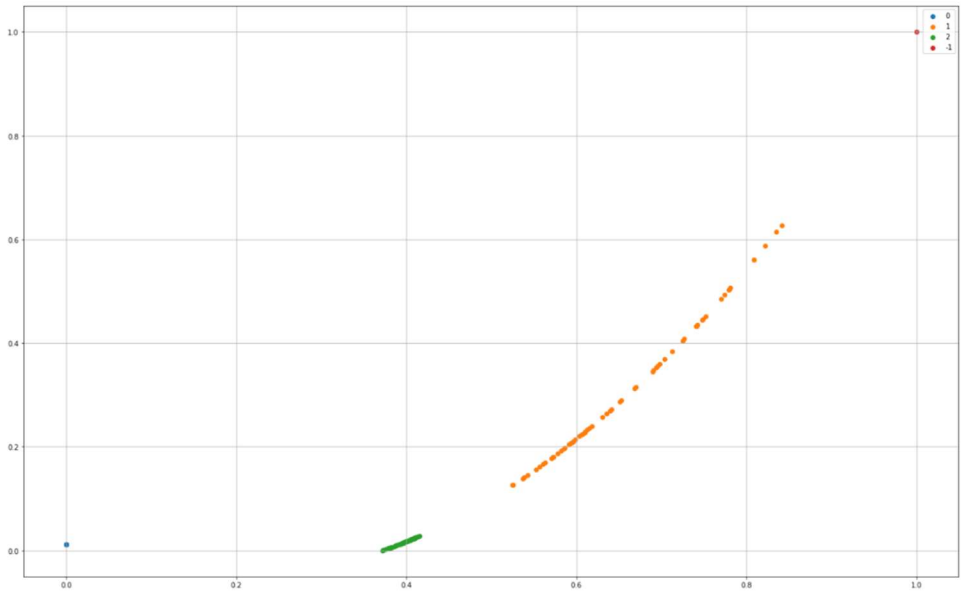

Figure 53: Plot of how the anomaly detection system classifies potentially anomalous information in strip 24 of the example histogram. The mode is set to “zonewise”.

Here, the “zonewise” mode has differences in what anomalous clusters it detects vs the stripwise mode. The two smallest clusters here are blue and red, but due to the threshold settings, only one data point is reported suggesting one of the two clusters contains more than 10 data points. The anomalous data point reported in this case is [(24, 2, 1.0)] which does not include the =0 coordinate values due to there being several present from multiple strips’ 0 values being present and the distance these values are away from the next closest cluster.

One important note is that, while the experiments yielded no advantage to using one normalization method over the other, the necessity of normalizing the values was recognized during the DBScan part of the system so that the distances over all test histograms, strips, reconstructed occupancy values, and occupancy values will be of the same range so the distance between clusters can be properly measured.

In addition to the “minpts” there is also a distance-based parameter called epsilon for DBScan or “eps_threshold” in the program. This is another key parameter to optimize the performance of the model and will be discussed further in later results.

All anomalies identified by all settings of the system will now be reported for this example histogram. The 3rd value in the tuples reported are based on the normalized reconstructed error for the histogram or strip depending on the cluster setting. Therefore, these should not be judged on the histogram against the input histogram occupancy values. In the example above using the “global”, “stripwise”, and “zonewise” settings, binary masks are constructed as heatmaps of the occupancies in **Figure 54**, **Figure 55**, and **Figure 56**. Outliers are visualized in this way for comparison with the inputs and expected results.

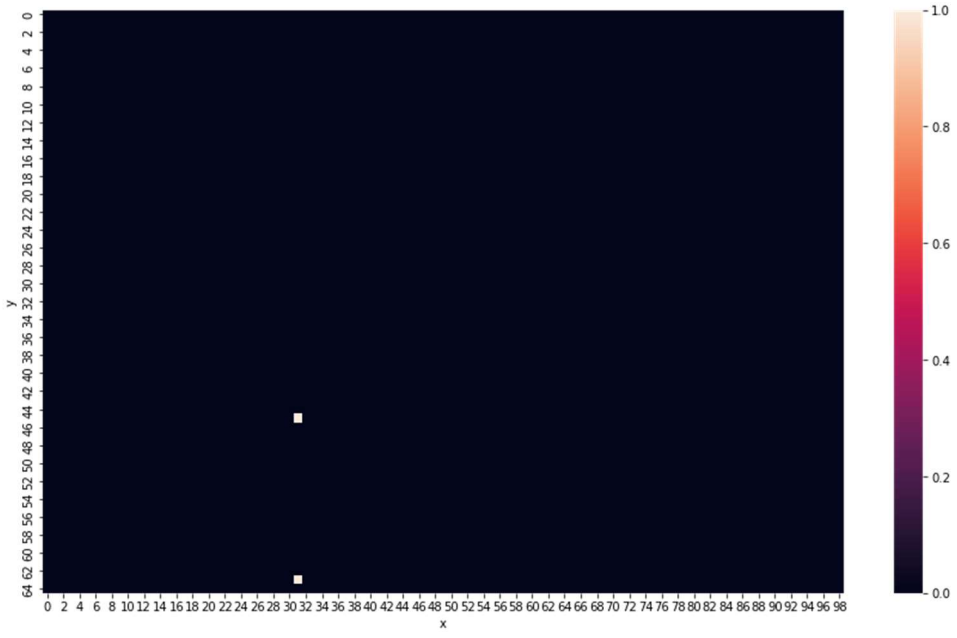

Figure 54: Binary Mask of classification results from the model set to the “global” setting on the example histogram.

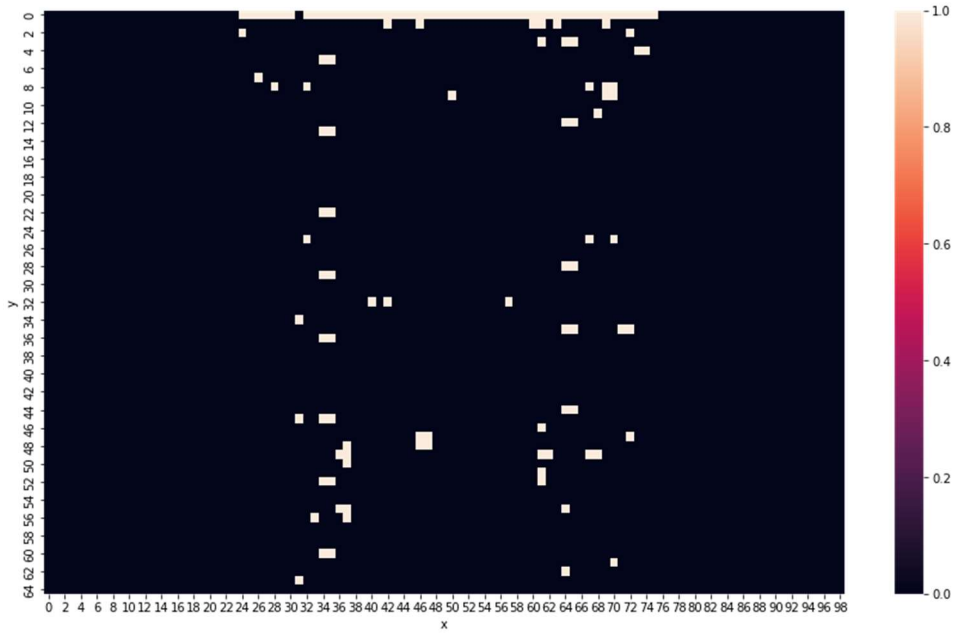

Figure 55: Binary Mask of classification results from the model set to the "stripwise" setting on the example histogram.

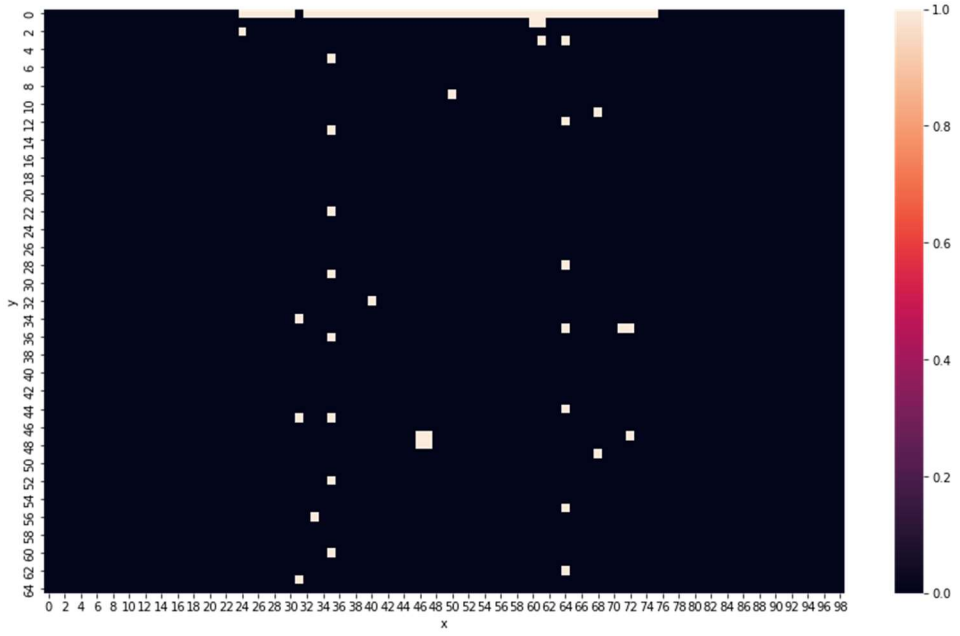

Figure 56: Binary Mask of classification results from the model set to the "zonewise" setting on the example histogram.

In the program, the output is currently given as coordinate tuples that can be further processed but may be adapted this way in the future.

When comparing the reported outliers by all the modes with the original input histogram, the model can be seen to pick up the hot and cold spots, strips, and layers for this histogram at a very fine level of detail. The question of whether or not the reported values are indeed anomalous depends on the requirements of the shifter and this sensitivity of the model can and should be adjusted in the "eps_threshold" and "minpts" parameter settings and/or by selecting the best or all cluster settings (global, stripwise, or zonewise).

Attention will now be given to explaining the classification report based on the generated anomalous data points in this histogram.

As it happens, this histogram has a single anomalous data point that was generated in it. In **Figure 57**, it appears to properly classify this anomalous data point with the system, but there appears to be an additional 131 false positive (true inlier, but predicted outlier) from the system. The reason for this is due to limited access to ground truth and clean data. With careful review of the previous figures, what appears to be happening with the detection system is that values that are likely to be truly anonymous (and not indicated as such up front by the dataset) have been identified. This, of course, is subjective, but results can be reviewed by subject matter experts to further determine the utility this unsupervised approach provides. This configuration can of course render an unnecessarily high number of false positives as is the case here, but the choice of "epsilon threshold", "minpts" settings and "global", "stripwise", or "zonewise" cluster settings allows the system to adjust the user tolerance for false positives and false negatives as will be shown momentarily.

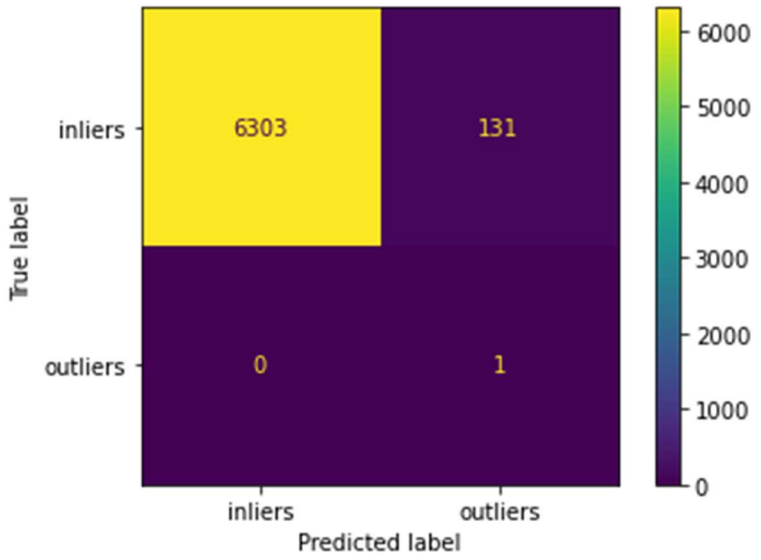

Figure 57: Classification report using the histogram data from **Figure 53** and previous figures. 6303 True Positive classifications, 131 False Negative classifications, 0 False Positive classifications, and 1 True Negative classification was reported in this evaluation. The threshold settings to achieve these numbers was "eps_threshold" = 0.1 and "minpts" = 10

In order to demonstrate the model's true ability to classify outliers, it was previously mentioned hot and cold spots, strips, and layers have been randomly generated in the datasets. The example histogram from above demonstrates the model's potential ability to predict outliers that were both generated and internal to the dataset. The following examples will demonstrate the model's performance on significantly more anomalies generated in the datasets to pressure test the system. In order to do this, the need to optimize the threshold parameters and report the model's abilities with the well-known ROC curve and AUC metric has been recognized [81, 82].

After a random search through a range of threshold values, the values that converged and offered the best results are shown **Figure 58**. As a note, this ROC curve has been calculated for the two thresholds rather than the typical single threshold. Having searched and found a convergent pattern for the best scores, "eps_threshold" = 0.0044

has been in this case set for the first threshold yields the following ROC curve over a range of “minpts” values.

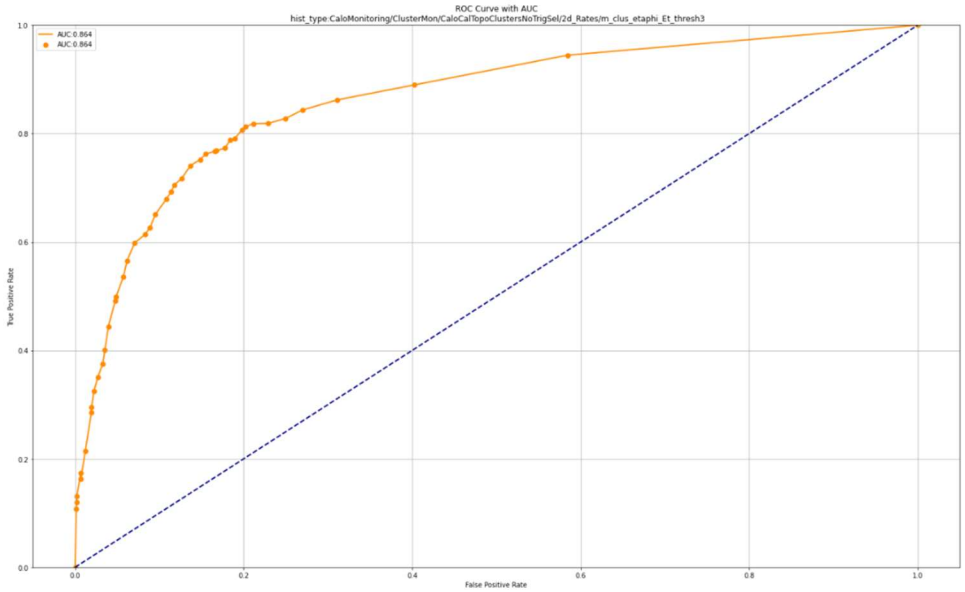

Figure 58: ROC curve of the example histogram from **Figure 43** and **Figure 44**. The AUC is calculated as 0.864 over the threshold iterations.

This graph demonstrates with some interpretation how effective the algorithm is at anomaly detection. An AUC of 0.864 is presented. It is said that an AUC score of 0.5 classifies positive and negative values about as good as a random guess. According to Homer et. al., an AUC of 0.5 suggests no discrimination, between 0.5 and 0.7 suggest poor discrimination, between 0.7 and 0.8 we consider as acceptable discrimination, between 0.8 and 0.9 we consider as excellent discrimination, and greater than 0.9 as outstanding discrimination [83].

On the other hand, the context of this evaluation must also be taken into consideration. As one user suggests, “If you are a trader and you can get an AUC of 0.501 in predicting future financial transactions, you're the richest man in the world. If you are a CPU engineer and your design gets an AUC of 0.999 at telling if a bit is 0 or 1,

you have a useless piece of silicon.", so the true importance of this metric depends on the domain in which it is applied [unknown quote author]. How important is a high true positive rate versus a low false positive rate for this anomaly detection system? For physicists working on certified data, a high true positive rate means there exists a lower type 1 error removing potentially useful bits of data. A low false positive rate means there exists a lower type 2 error removing potentially harmful bits of data. Since these thresholds can be set as desired with the automatic system ahead of time, the final decisions on which type of error is better to minimize are better left for the subject matter experts. Assuming the type 2 error in this case is primarily what the data quality assessment process aims to optimize, with type 1 error second to that, one option is to first select the tolerance for false positive rate, then select the highest true positive rate that suits that limit. Although, the reverse is also possible.

At this point in the evaluation, notice the example histogram that has been being used so far is of the type 15 according to section 3.6's numbering of histogram types in the dataset "CaloMonitoring/ClusterMon/LArClusterEMNoTrigSel/TransEnergy/etaphi_thresh_avgEt_0". All 18 types of histograms are based on the calorimeter sub detector systems and share similarities that suggest their results will be similar. To verify this assumption, similar ROC-AUC based results will be provided as above for one example of each of the remaining 17 types of histograms and determine any differences in performance based on histogram type. Hist type 15 has already been evaluated so it is skipped in **Figure 59** through **Figure 75**.

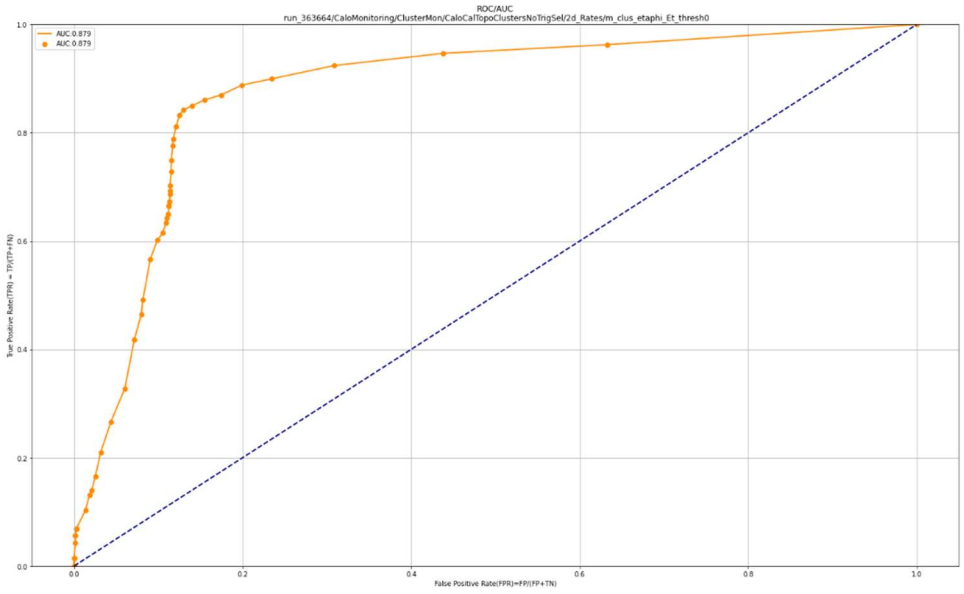

Figure 59: ROC curve of histogram type 0. The AUC is calculated as 0.879 over the threshold iterations.

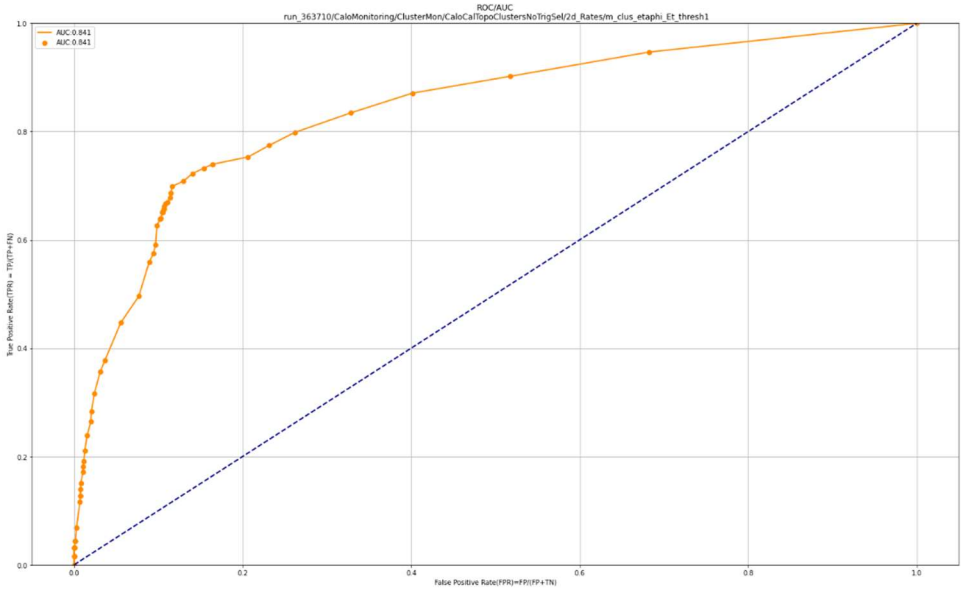

Figure 60: ROC curve of histogram type 1. The AUC is calculated as 0.841 over the threshold iterations.

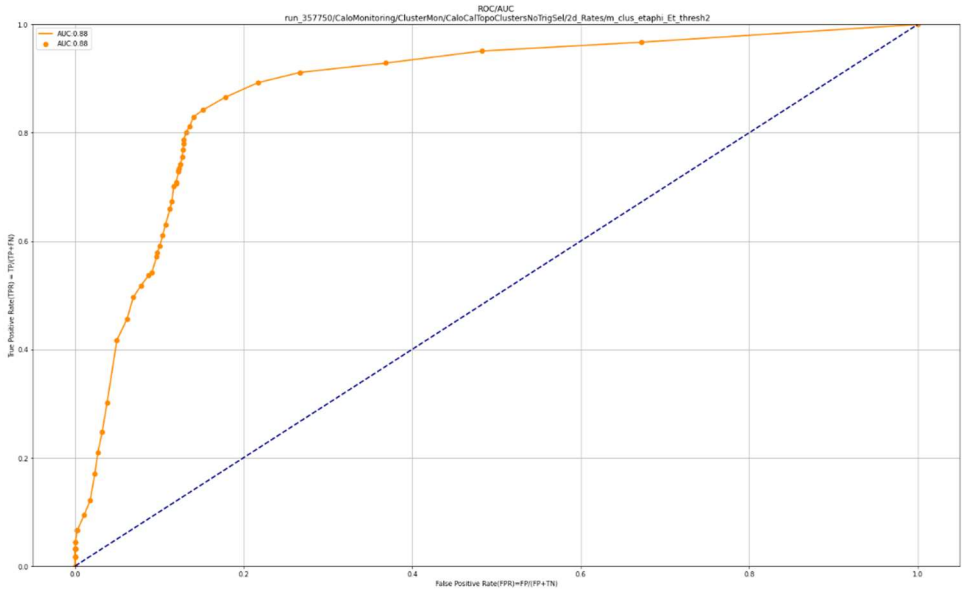

Figure 61: ROC curve of histogram type 2. The AUC is calculated as 0.88 over the threshold iterations.

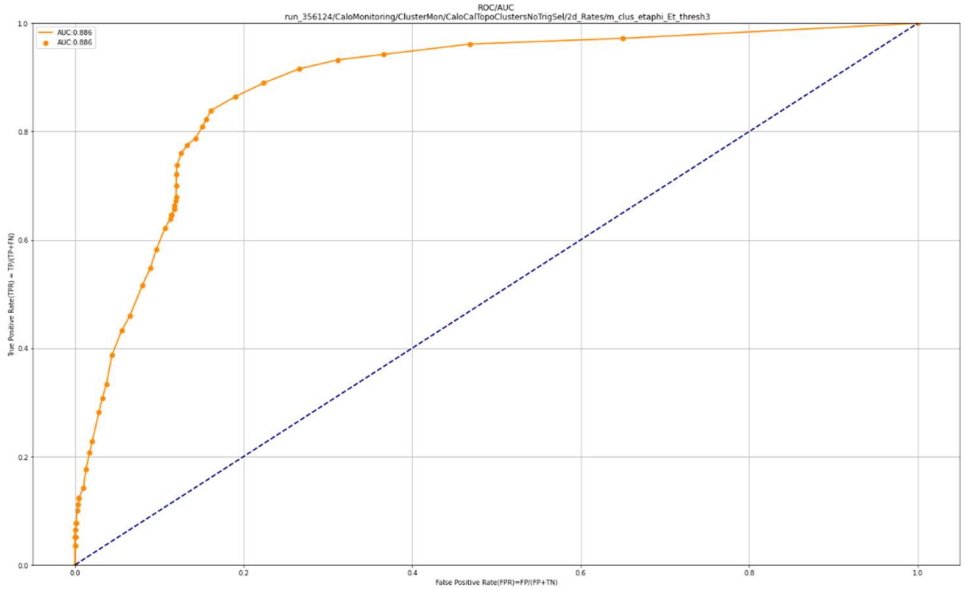

Figure 62: ROC curve of histogram type 3. The AUC is calculated as 0.885 over the threshold iterations.

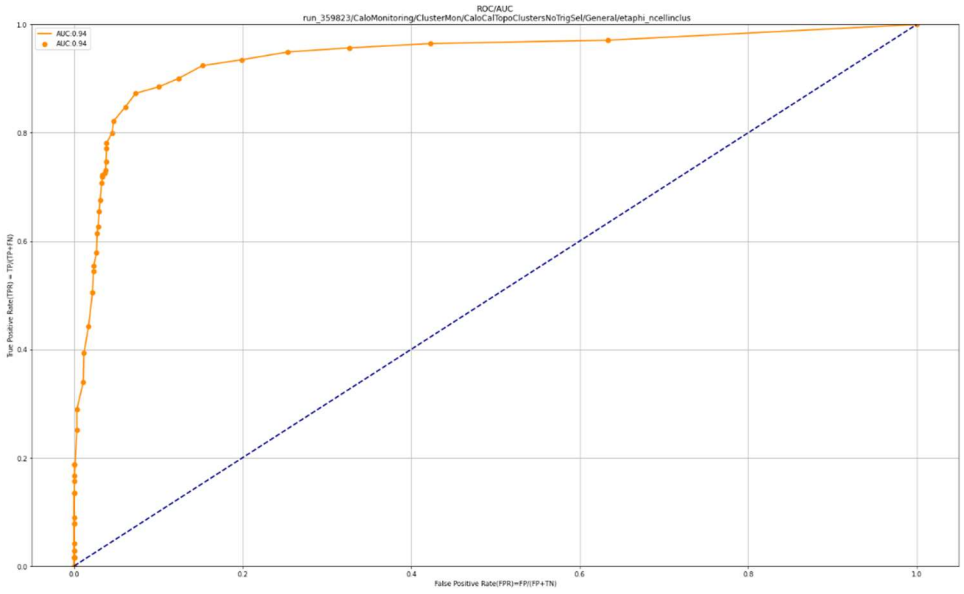

Figure 63: ROC curve of histogram type 4. The AUC is calculated as 0.94 over the threshold iterations.

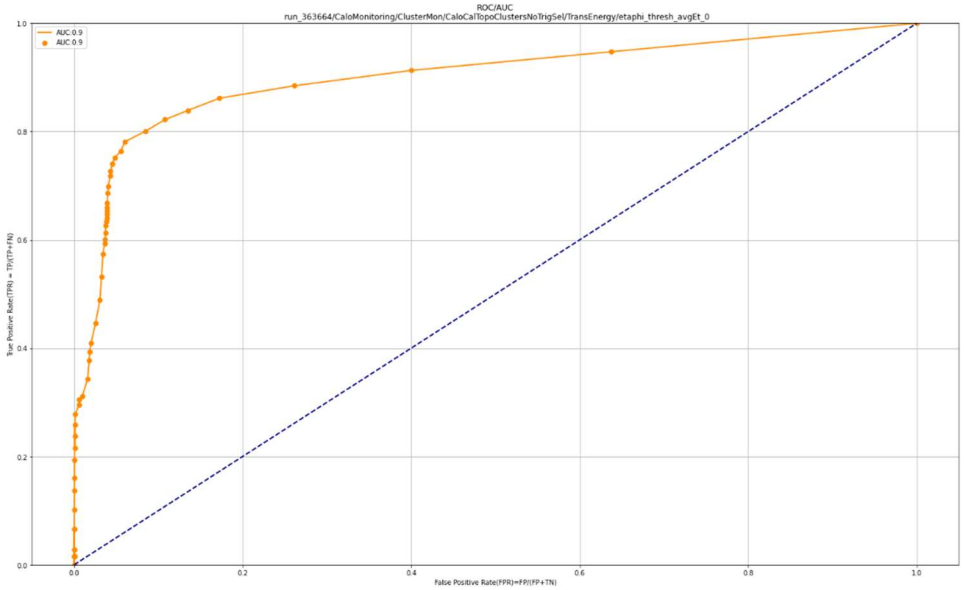

Figure 64: ROC curve of histogram type 5. The AUC is calculated as 0.9 over the threshold iterations.

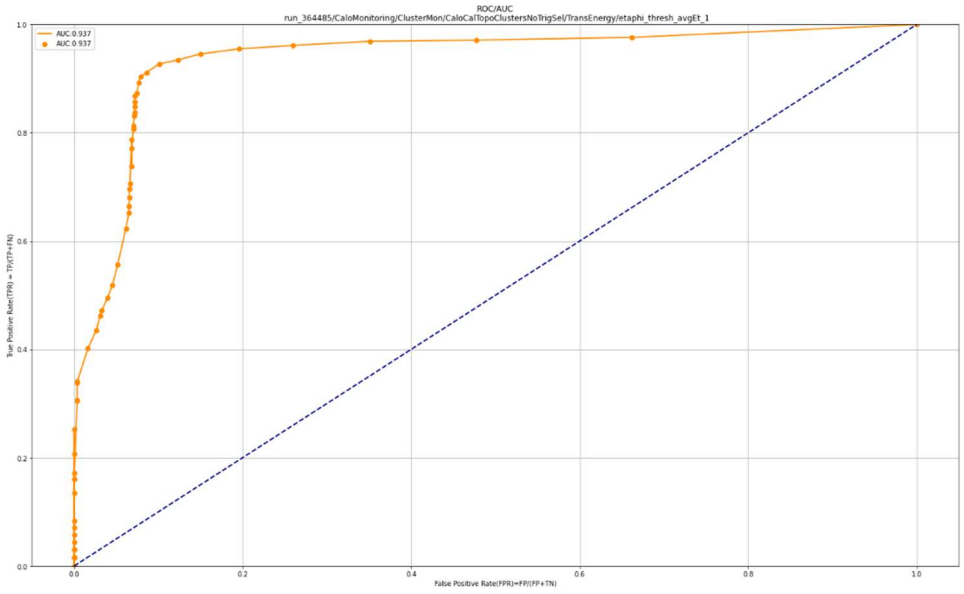

Figure 65: ROC curve of histogram type 6. The AUC is calculated as 0.937 over the threshold iterations.

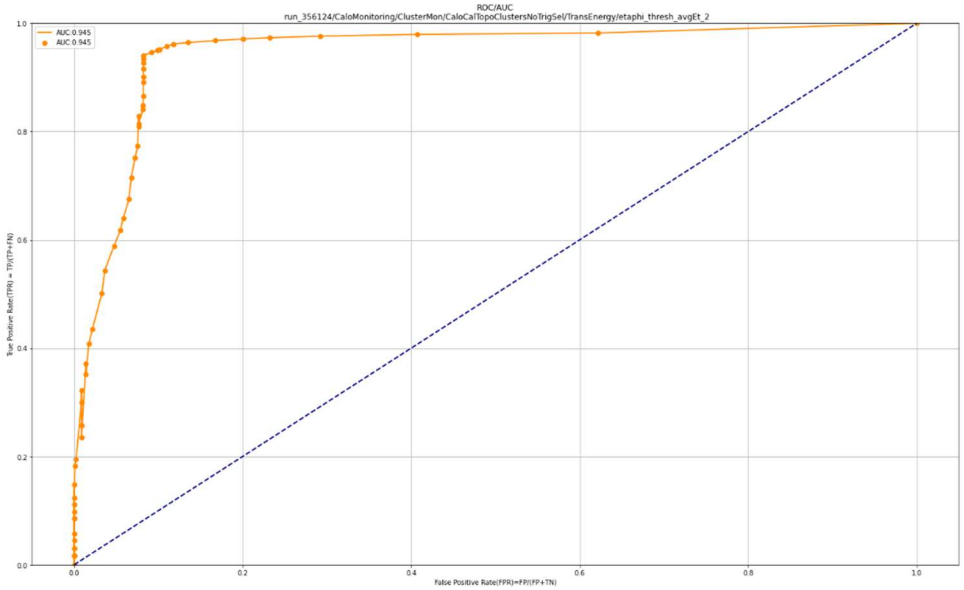

Figure 66: ROC curve of histogram type 7. The AUC is calculated as 0.945 over the threshold iterations.

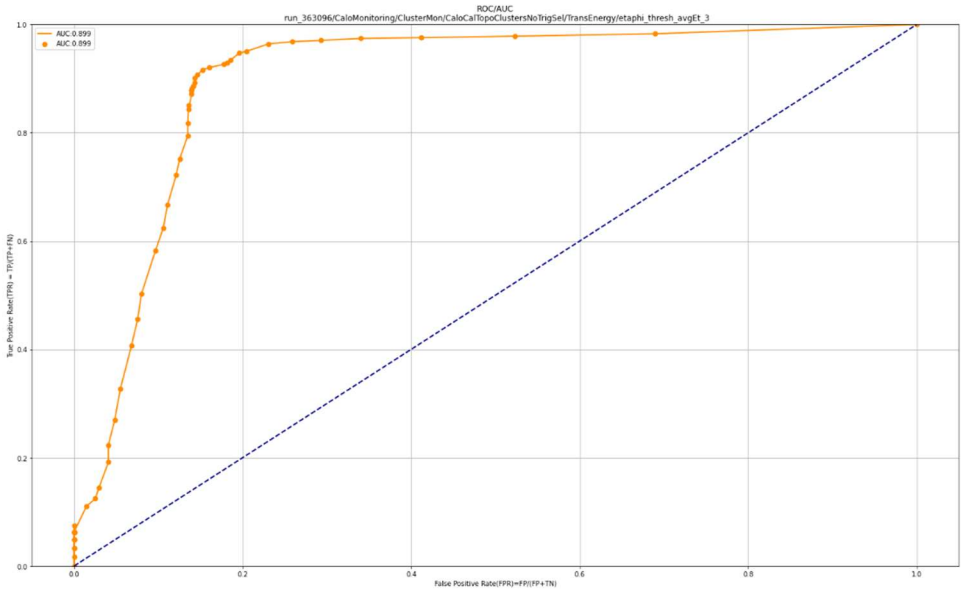

Figure 67: ROC curve of histogram type 8. The AUC is calculated as 0.899 over the threshold iterations.

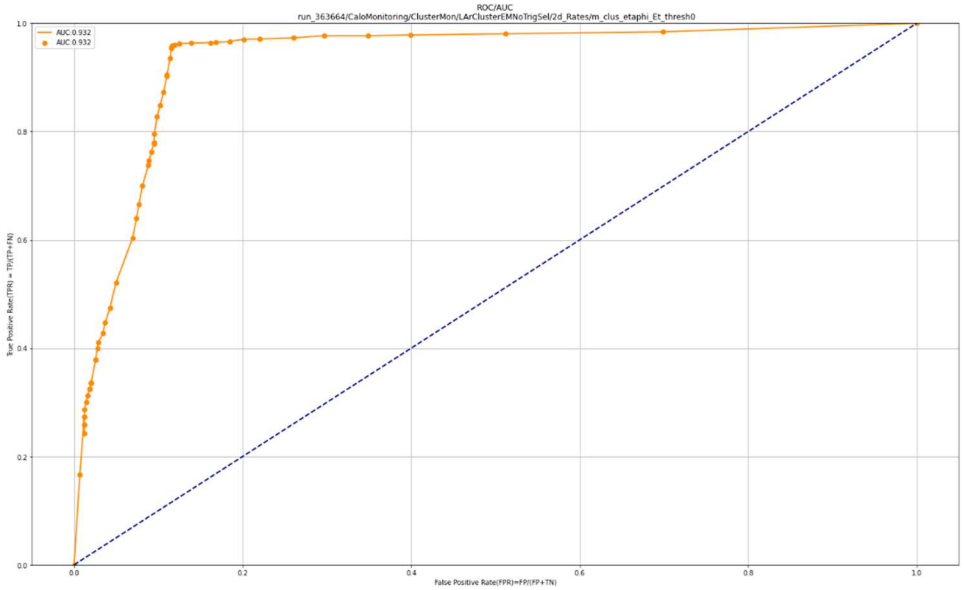

Figure 68: ROC curve of histogram type 9. The AUC is calculated as 0.932 over the threshold iterations.

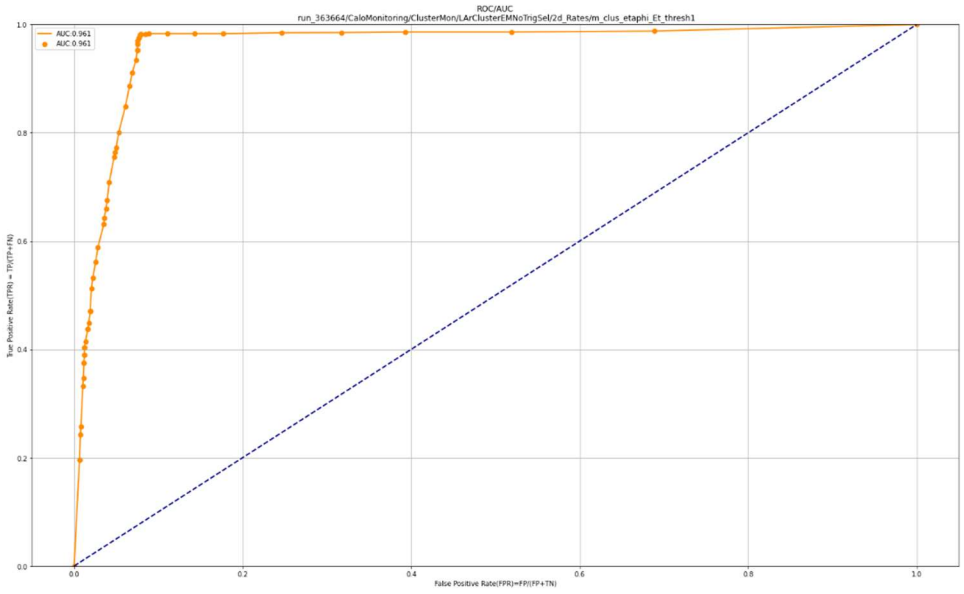

Figure 69: ROC curve of histogram type 10. The AUC is calculated as 0.961 over the threshold iterations.

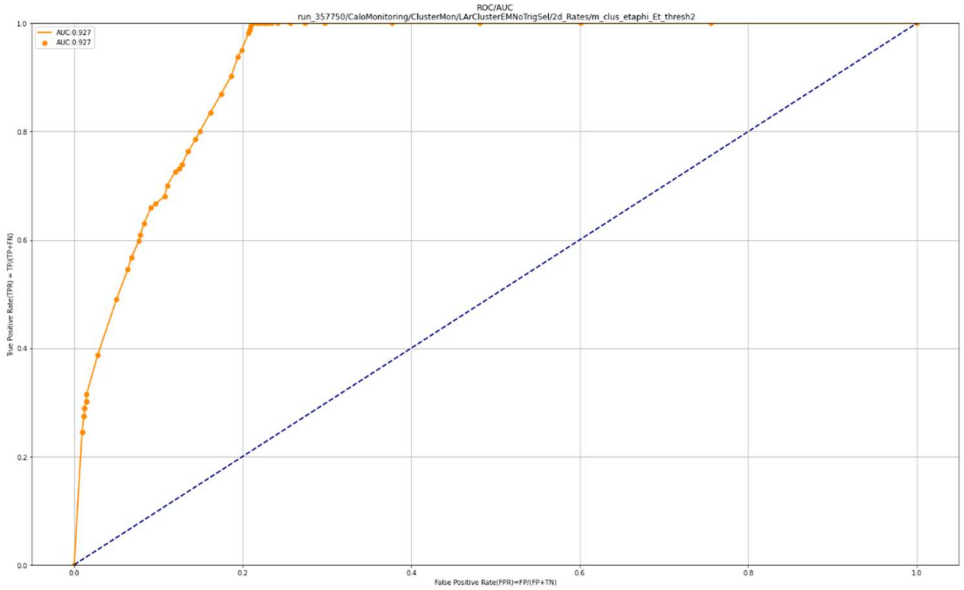

Figure 70: ROC curve of histogram type 11. The AUC is calculated as 0.927 over the threshold iterations.

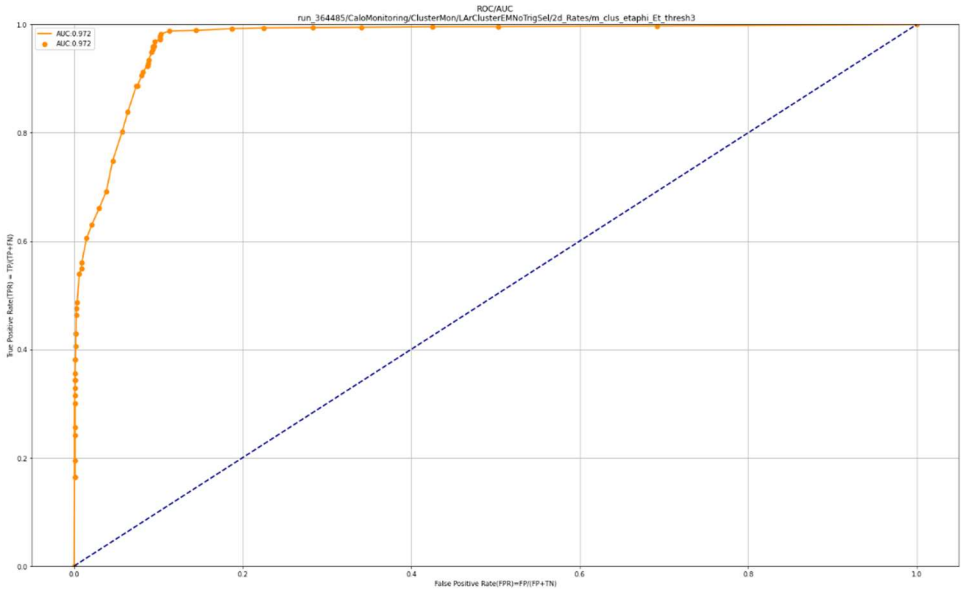

Figure 71: ROC curve of histogram type 12. The AUC is calculated as 0.972 over the threshold iterations.

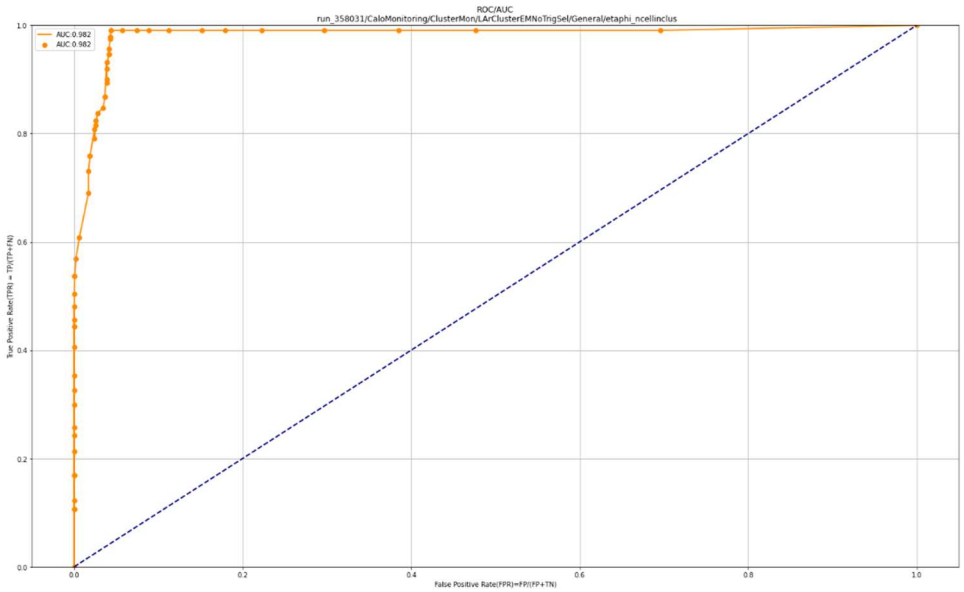

Figure 72: ROC curve of histogram type 13. The AUC is calculated as 0.982 over the threshold iterations.

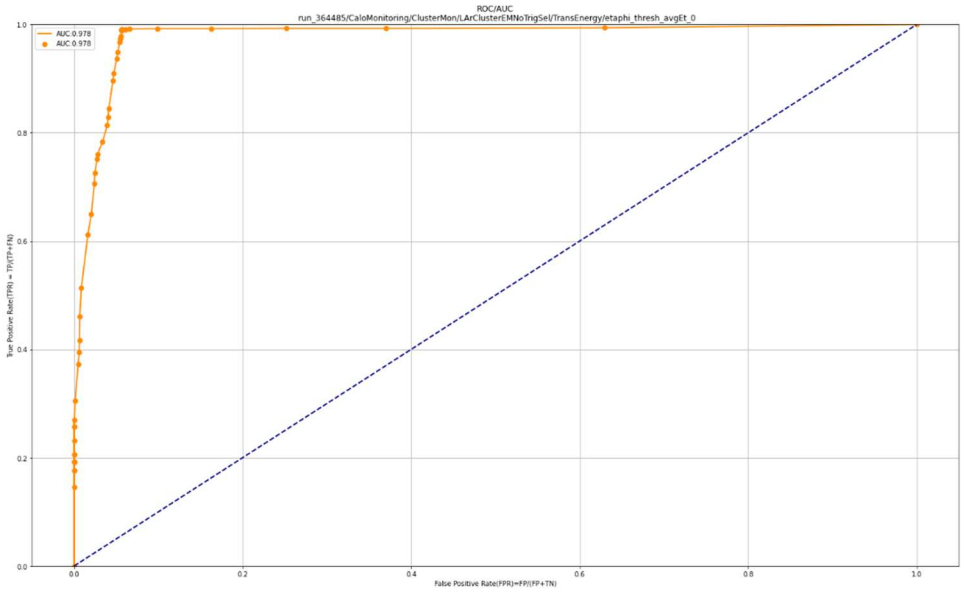

Figure 73: ROC curve of histogram type 14. The AUC is calculated as 0.978 over the threshold iterations.

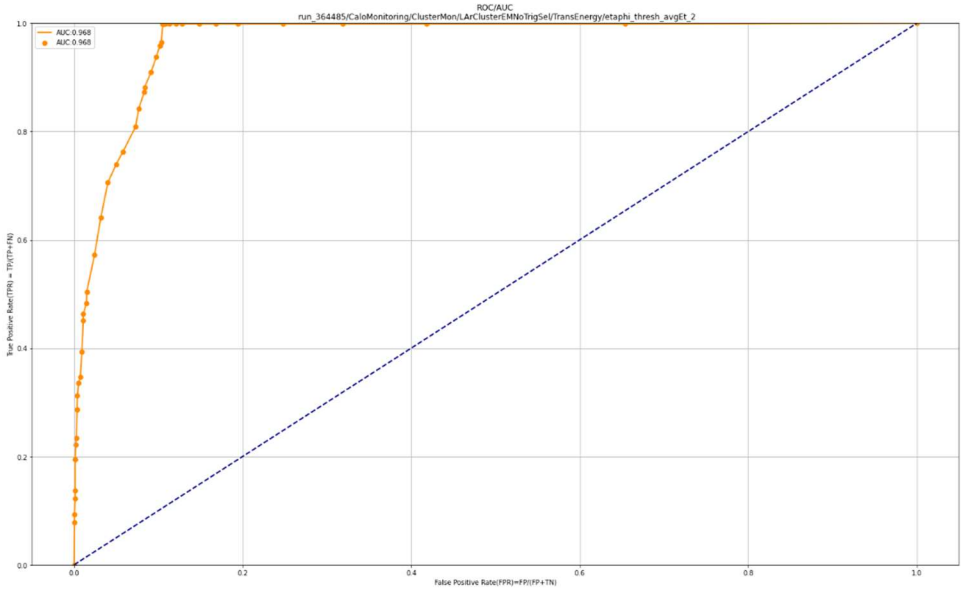

Figure 74: ROC curve of histogram type 15. The AUC is calculated as 0.968 over the threshold iterations.

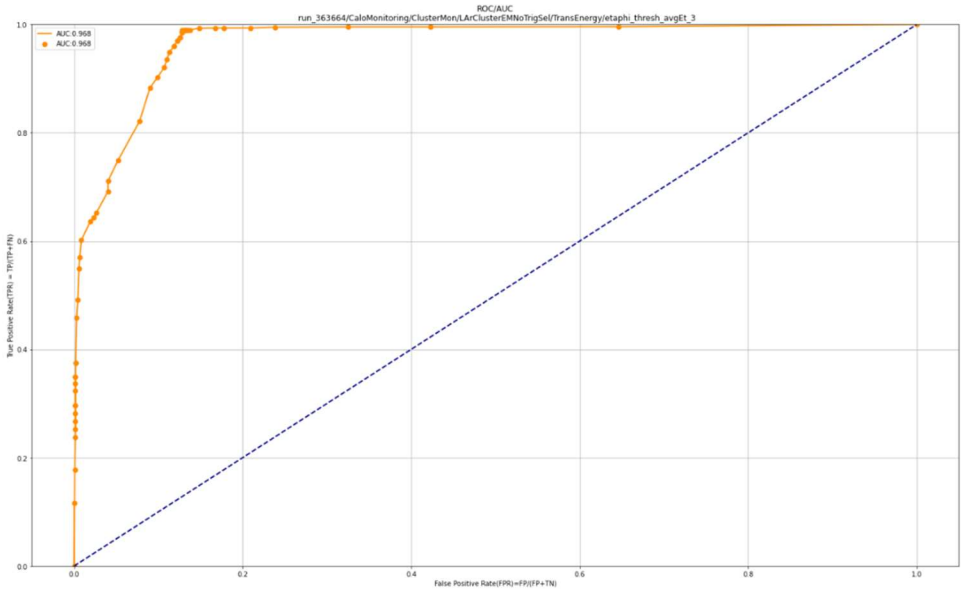

Figure 75: ROC curve of histogram type 17. The AUC is calculated as 0.968 over the threshold iterations.

The average calibration time for each histogram (depending on the threshold search size), was approximately 6 hours. Having provided these results, it appears the threshold settings differ from histogram type to histogram type. This means there will be at minimum one full threshold calibration required for this system to detect anomalies prior to production use. The final detail to verify is that the results do not differ within the same histogram type, but with different run numbers and/or ftags. This would mean that every single histogram would need known anomalous values to calibrate on and time to calibrate prior to running the system rendering it impractical. However, it is possible that the normal distribution from histogram type to histogram type is sufficiently different that despite the generated anomalies being of the same character, a threshold calibration will be required once for each histogram type of interest by the detector. The following results will clarify whether a single calibration holding good performance results can be depended on for different runs and/or ftag histograms of the same histogram type. With

the same calibration that generated its ROC curve, do the results seem stable for a single histogram over different runs? The results of this are given in **Figure 76**, **Figure 77**, and **Figure 78**.

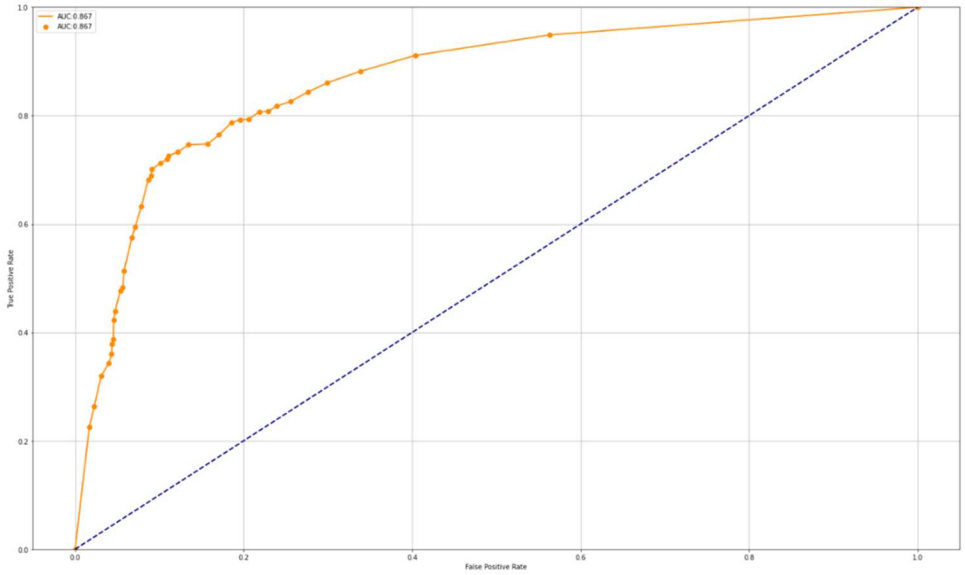

Figure 76: ROC curve of histogram type 3. The AUC is calculated as 0.867 over the threshold iterations. The specific run and ftag_id for this result is run_356124 and 6 respectively.

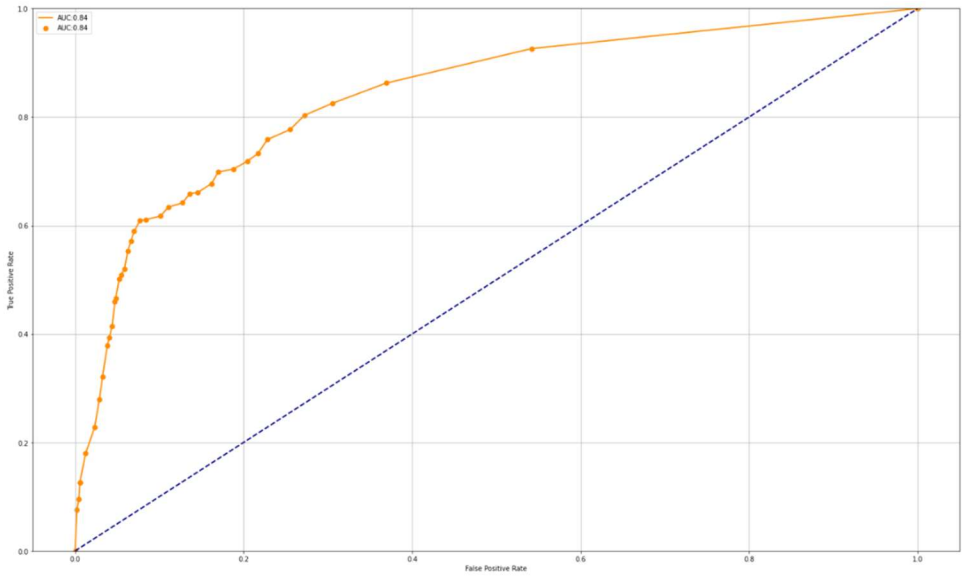

Figure 77: ROC curve of histogram type 3. The AUC is calculated as 0.84 over the threshold iterations. The specific run and ftag_id for this result is run_357750 and 8 respectively.

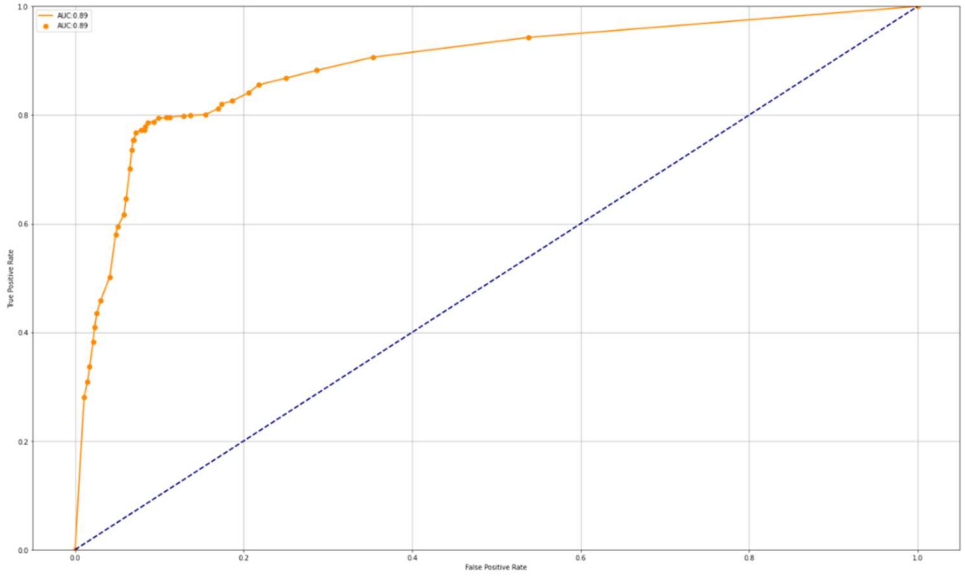

Figure 78: ROC curve of histogram type 3. The AUC is calculated as 0.89 over the threshold iterations. The specific run and ftag_id for this result is run_358031 and 42 respectively.

Stable behavior is reported for a single histogram type calibrated for a specific "eps_threshold" (0.0044) varied over "minpts" (0-40). The AUC is consistently above

0.8. More histograms have been tested and all return an AUC score of over 0.8 some reaching into the low 0.9 score area. The example histogram from earlier in this chapter can be found in the web display with the information according to **Figure 79**.

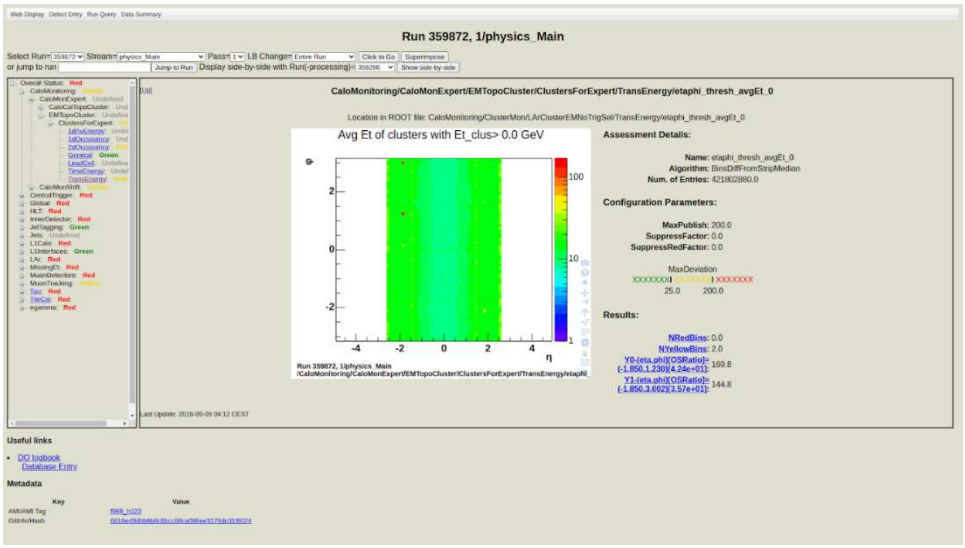

Figure 79: Note that the heatmaps of monitoring histograms are plotted such that the y axis is reversed for convenience and increases vertically downward rather than upward in this figure [84].

CHAPTER 5

DISCUSSION

In **Figure 41** and **Figure 42**, the MSE has an interesting effect on the input values from the monitoring histogram. On the high end the values can be used to identify hot spots, but on the low end the affected values can be used to classify cold spots as well. This behavior has been exploited in later figures where cold spots can be seen as properly identified as expected.

In **Figure 43** and its reported total error, the reconstruction error is extremely low, meaning that the Complete Autoencoder has learned a near identical representation of the input.

Experiments have shown that the autoencoder trained in this way reconstructs the input histogram to a great degree of accuracy. In **Figure 44** and **Figure 45** the input histogram is reconstructed very accurately by the autoencoder used by the system. **Figure 46** and **Figure 47** give a 2-dimensional look of what exactly happens to the absolute values of the occupancy values that are reconstructed from the input histogram. The figures show the relative distance of the points from one another are the same (leading to a near exact heat map or image reconstruction), but the exact values of occupancy between the input histogram and reconstructed histogram differ. **Figure 48** shows that only “hot” coordinates tend to show a higher reconstruction error while “cold” coordinates have no such behavior.

In the second part of the system where the DBScan algorithm takes over, **Figure 49**, **Figure 50**, **Figure 51**, **Figure 52**, and **Figure 53** show the cluster behavior the algorithm takes advantage of as well as gives insight into how the algorithm classifies

anomalous data points in each setting the system can be put in (**Figure 50**, the global setting; **Figure 51** and **Figure 52**, the stripwise setting; **Figure 53**, the zonewise setting).

Figure 54 evaluates a non-clean histogram that has had a single anomaly manually generated into it and with the threshold settings set to high sensitivity, the system picks up the global anomalous which are the same as the current in use algorithm "BinsDiffFromStripMedian" and goes further to identify more suspect data points. The data points generated in this way have been provided in their entirety for further interpretation. The results for the stripwise and zonewise settings can be similarly evaluated in **Figure 55** and **Figure 56** respectively. **Figure 57** reports that of the 6,435 datapoints in the histogram, 6,303 true inliers were classified, 1 true outlier was classified, 0 false inliers were classified, and 131 false outliers were classified.

The overall model performance ability of the model is measured using the same dataset with significantly more anomalies generated. The "eps_threshold" is calibrated by scanning over well performing threshold values, and after being selected the ROC curve is generated by varying the "minpts". The model shows very good performance with an AUC of 0.864 from **Figure 58**. To determine if the model is stable with the same calibrated threshold values for different histogram types, more ROC curves were generated. In **Figure 59** through **Figure 75,** the AUC values differ with respect to unique threshold combinations, but following a calibration process, show high performance with any histogram type**.** This suggests the model does not require recalibration for each histogram it is used for, and a calibration exists for each histogram type allowing for high performance throughout at least the 18 histogram types analyzed in this work.

In **Figure 76**, **Figure 77**, and **Figure 78**, the model consistently performs well for the same histogram type with the same threshold calibration values. Several more curves were generated of the same histogram type with different run and/or ftag values, but all demonstrated the same stable performance.

It is likely that the developed system generally scores high in performance due to the types of anomalies being generated in all histograms being of roughly the same character (spots in random locations, horizontal strips, horizontal layers, limitations on the length of anomalous strips and layers, intensity of the anomaly being between three and four sigma for the entire map rather than for that strip). This could mean that a more effective classification system would need to more accurately generate anomalous data points in a way that would best resemble detector anomalies. If the assumption can be made that the anomalous data we generated would be of that same character, then we can say the system's performance should be in line with the evaluation results presented.

Referring to the current data quality algorithm in use from chapter 2, BinsDiffFromStripMedian, this system can be calibrated to achieve a high-performance rating, and it can be adjusted to suit the user in terms of sensitivity to less visible anomalies or to only identify highly invasive anomalous data points. The DQAlgorithm's thresholds for the tolerable number of differences to render a yellow or red flag on the histogram is set to a constant value (one for yellow, one for red) for the algorithm and is not adapted to each histogram type. The DQalgorithm renders yellow flags for the two anomalies identified by this system as outliers with the global setting as can be seen in **Figure 79**.

Since the objective of this work was not to create a state-of-the-art machine learning architecture, the focus has been on developing a functional machine learning system that can improve on the manual systems in place for data quality assessment. Refinements that compete with hardware and software on the cutting edge of machine learning are beyond the scope of this work.

Previous research in this domain leads one to believe that the general performance of machine learning based anomaly detection systems often are on par or exceed the performance of standard methods of anomaly detection in high energy physics, but the absolute performance of the same architecture varies with the exact control established when constructing a dataset, selecting and engineering features, and developing a solution to a unique problem. This suggests it is possible a simple machine learning based anomaly detection architecture could outperform more complex methods in terms of absolute accuracy if the experimental setup is superior in design but requires confirmation with further research. The scope and limitations in this work, for example, differ from that of others work in that most Data Quality HEP ML models are designed to work with TH2 histograms rather than TH1 histograms. For works that also utilize the TH2 histograms, other input features are considered beyond the topological features that this scope is limited to. In future iterations, complex features could be engineered with respect to the TH2 histogram that positively impact model performance.

During the experimental phase, an issue was ran into an issue where resulting occupancy heatmaps seem to get stuck in a sweeping set of zero values. This issue is commonly known as the “dead Relu” problem caused from certain values taken in by the Relu function returning many zero values. The issue was noted and fixed by switching

the Relu activation function in the model architecture to the LeakyRelu activation function with default settings.

Is it important to use the autoencoder model for this task at all considering it seems the clustering model is doing most of the heavy lifting? Truly, a simplified version of this could use the clustering algorithm itself. Having demonstrated how the reconstruction error impacts the distances between occupancy clusters, it is likely important to improve the performance of the model beyond what a simple cluster algorithm would provide because it is those distances specifically the clustering algorithm targets when it becomes the outlier decision function. The comparison can be made between this system and other systems directly without the need for such experiments, but if there were interest in developing this technique, further study could be of value to the interested researcher.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

In this work, infrastructure has been constructed, initial theoretical foundations have been established, and first-generation software has been built to achieve the provided objectives. The infrastructure will allow collaborators of different backgrounds in the ATLAS experiment access to tools and a system that will greatly simplify the initial work that would be required to begin experimenting on this application for future developments. The established theoretical concepts include the data quality assessment background in ATLAS, autoencoders, DBScan, the outlier decision function or threshold for anomaly classification, and other fundamental concepts to build on. An anomaly detection system has also been provided that can achieve high quality, automatic anomaly detection results with proper threshold calibration (AUC ~ 0.85) built with a uniquely constructed dataset (local feature descriptors or histogram coordinates versus the one input per pixel approach which was necessary due to histogram area and availability of data). This model has been provided as well as a dashboard for validation deployed by Docker to become part of the first generation of machine learning based data assessment tools for the ATLAS experiment.

In future work, recommendations include using one calibration per histogram type assuming the types of anomalies will be of the same character (density, frequency, shape relative to the histogram 2d map, etc.). The pressure tested system where many anomalies were generated is calibrated for the “eps_threshold” distances that often occur due to how it was generated in the code. Natural occurring outliers should have their own pattern that can be calibrated for. In **Figure 57**, using “eps_threshold” = 0.1 and “minpts”

= 10, the system picks up suspect anomalous data points that are internal to the dataset prior to anomaly generation. The “eps_threshold” in this case is about 3 orders of magnitude different from the calibrated value found for the generated anomalies in the same histogram type (0.0044). Again, it is essential to tune these parameters for future work with and use of this model to reproduce these results.

This calibration step assumes that enough data exists to gather classification information. It is possible to use previous historical data to calibrate each histogram type between run periods, but some consideration should be made for what kind of differences in that histogram type appear from data taking period to period. However, since the shape of the detector and natural laws will hold constant, it should be possible to use this historical data with these considerations.

In terms of raw performance, the typical CNN supervised method has been shown to be superior to this method (AUC=0.99 [5]), but due to the number of inputs being equal to the number of coordinates for this setup, ATLAS’s large TH2 histograms would require a significant number of ground truth labeled (detailed defect tagged) anomaly information before this high performing technique would be possible. While the model has been trained on a large number of histograms, the developed technique allows for a significantly lower number of training histograms (the number of data points is in far excess to the number of features present in the dataset) to render above average results and can be a significant advantage earlier in a run period where high amounts of expert labeled data is not present. The CNN model’s behavior could be more directly trained to reproduce subject matter expert behavior while this model could provide superior performance for more detailed or less detailed anomaly detection depending on how its

thresholds are calibrated. The choice of which model to use is not black and white, but a gray area where the requirements of the shifters and physicists will dictate which method better fits the application.

BIBLIOGRAPHY

[1] - Froidevaux, D., Sphicas, P. (2009, April 13). CMS vs ATLAS, Comparison of Detectors. Retrieved from https://twiki.cern.ch/twiki/bin/viewfile/Sandbox/MitPartIIIPracticeRoster2009?rev=1;filename=cms_vs_atlas_overview.pdf

[2] - Catmore, J., Heinrich, L., Köneke, K., & Torrence, E. (2016, August 2). *The ATLAS data processing chain: from collisions to papers*. Indico.Cern.Ch. https://indico.cern.ch/event/472469/contributions/1982677/attachments/1220934/1785823/intro_slides.pdf

[3] - Aad, G., Abbott, B., Abbott, D., Abud, A., Abeling, K., Abhayasinghe, D., Abidi, S., AbouZeid, O., Abraham, N., Abramowicz, H., Abreu, H., Abulaiti, Y., Acharya, B., Achkar, B., Adachi, S., Adam, L., Bourdarios, C., Adamczyk, L., Adamek, L., . . . Plotnikova, E. (2020). *ATLAS data quality operations and performance for 2015–2018 data-taking*. Journal of Instrumentation, 15(04), P04003. https://doi.org/10.1088/1748-0221/15/04/p04003

[4] – The ATLAS Collaboration (2020). Operation of the ATLAS trigger system in Run 2. *Journal of Instrumentation*, *15*(10), P10004. https://doi.org/10.1088/1748-0221/15/10/p10004

[5] - Fidalgo Rodriguez, G. (2018, June 18). *Using Machine Learning Techniques for Data Quality Monitoring at CMS Experiment*. Indico.Cern.Ch. https://indico.fnal.gov/event/16384/contributions/37221/attachments/23114/28654/Copy_of_Fermilab_Talk.pdf

[6] - Fernando, T., Gammulle, H., Denman, S., Sridharan, S., & Fookes, C. (2022). Deep Learning for Medical Anomaly Detection – A Survey. *ACM Computing Surveys*, *54*(7), 1–37. https://doi.org/10.1145/3464423

[7] - le Meur, J., Vigen, J. (2009, September 21). *CERN - History and achievements from 1954 to 2009: great successes - missed opportunities*. Indico.Cern.Ch. https://indico.cern.ch/event/65380/attachments/1014158/1443126/Butare_CERN_history-achievements.pdf

[8] - Mobs, E. (2019, July 29). *The CERN accelerator complex - 2019 Complexe des accélérateurs du CERN - 2019* [Illustration]. https://cds.cern.ch/record/2684277

[9] - Froidevaux, D., Sphicas, P. (2006). General-Purpose Detectors for the Large Hadron Collider. *Annual Review of Nuclear and Particle Science*, *56*(1), 375–440. https://doi.org/10.1146/annurev.nucl.54.070103.181209

[10] – The ATLAS Collaboration. (2008). The ATLAS Experiment at the CERN Large Hadron Collider. *Overview of the ATLAS Detector*, 1–18. https://jinst.sissa.it/LHC/ATLAS/2008_JINST_3_S08003.pdf

[11] – Garutti, E. (2012, June 17). *Calorimeters Energy Measurement* [Slides]. Deutsches Elektronen-Synchrotron DESY. https://www.desy.de/~garutti/LECTURES/ParticleDetectorSS12/L10_Calorimetry.pdf

[12] - Sidoti, A. (2014). Minimum Bias Trigger Scintillators in ATLAS Run II. *Journal of Instrumentation*, *9*(10), C10020. https://doi.org/10.1088/1748-0221/9/10/c10020

[13] - *ATLAS Calorimeter | Brookhaven and the LHC*. (2013). Brookhaven National Laboratory. https://www.bnl.gov/atlas/lar.php

[14] - *UChicago ATLAS*. (2008). University of Chicago. https://hep.uchicago.edu/atlas/tilecal/

[15] – The ATLAS Collaboration. (2019). *ATLAS Data Quality in Run 2* [Diagram]. https://cds.cern.ch/images/ATLAS-PHOTO-2019-039-2

[16] - *Ensuring Quality of ATLAS Data*. (2009, September 7). ATLAS E-News. https://atlas-service-enews.web.cern.ch/2009/features_09/features_dataqual.php

[17] - Corso-Radu, A., Hadavand, H., Illchenko, Y., Kolos, S., Okawa, H., Slagle, K., Taffard, A. & The ATLAS Collaboration. (2011). Data Quality Monitoring Framework for the ATLAS experiment: Performance achieved with colliding beams at the LHC. *Journal of Physics: Conference Series*, *331*(2), 022027. https://doi.org/10.1088/1742-6596/331/2/022027

[18] - Madhuri, G., Rani, M. (2018). Anomaly Detection Techniques. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.3167172

[19] - *Chebyshev's & Empirical rules*. (2014). Sacramento State University. https://www.csus.edu/indiv/s/seria/lecturenotes/chebyshev.htm

[20] – The ATLAS Collaboration. (2015). *Projects Dashboard GitLab*. CERN ATLAS GitLab. https://gitlab.cern.ch

[21] - Allison, P. (2018). *Standard Error*. University of Pennsylvania. https://www.sas.upenn.edu/~allison/Oct8.pdf

[22] - *Oracle Crystal Ball Reference and Examples Guide*. (2015). Oracle Documentation. https://docs.oracle.com/cd/E40248_01/epm.1112/cb_statistical/frameset.htm?ch07s02s10s01.html

[23] – *Semisupervised learning with hadoop for understanding user web behaviours*. (2017). [Diagram]. Analyticsvidhya. https://cdn.analyticsvidhya.com/wp-content/uploads/2017/09/20182516/dataiku-hadoop-summit-semisupervised-learning-with-hadoop-for-understanding-user-web-behaviours-12-638.jpg

[24] - *A typical supervised learning algorithm*. (2020). [Diagram]. Analyticsvidhya. https://cdn.analyticsvidhya.com/wp-content/uploads/2020/04/A-typical-supervised-learning-algorithm.png

[25] - *Semi Supervised Learning Method Example*. (2020). [Diagram]. https://miro.medium.com/max/700/0*90nN8wUn68pME4Yw.png

[26] - Breunig, M., Kriegel, H., Ng, R., & Sander, J. (2000). LOF. *ACM SIGMOD Record*, *29*(2), 93–104. https://doi.org/10.1145/335191.335388

[27] - *Unsupervised Learning*. (2014). [Graph]. https://www.ecloudvalley.com/wp-content/uploads/2019/09/Unsupervised-learning.png

[28] - Ester, M., Kriegel, H., Sander, J. & Xu, X. (1996). A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. *Proc. of 2nd International Conference on Knowledge Discovery and Data Mining.* (p./pp. 226-231)

[29] – Kotary D., Nanda S. (2021, August 17). *A Distributed Neighbourhood DBSCAN Algorithm for Effective Data Clustering in Wireless Sensor Networks*. https://link.springer.com/article/10.1007/s11277-021-08836-y

[30] - London & Fountas, Z. (2022). Imperial College Spiking Neural Networks for Human-like Avatar Control in a Simulated Environment. [Diagram] https://www.researchgate.net/figure/The-simplest-mathematical-model-of-a-neuron-called-the-Perceptron-30_fig2_266485234

[31] - Rosenblatt, F. The Perceptron — A Perceiving and Recognizing Automaton. Tech. Rep. 85-460-1 (Cornell Aeronautical Laboratory, 1957). https://blogs.umass.edu/brain-wars/files/2016/03/rosenblatt-1957.pdf

[32] - Rumelhart, D., Hinton, G., & Williams, R. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536. https://doi.org/10.1038/323533a0

[33] - Pradhan, M., Pradhan, S., & Sahu, S. (2012). Anomaly Detection Using Artificial Neural Network. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.429.1280&rep=rep1&type=pdf

[34] -Andreas, A., Zombanakis, G. (2003). Intelligent information systems for defence problems. https://www.researchgate.net/publication/241765869_Intelligent_information_systems_for_defence_problems

[35] - Musiol, M. (2016). Speeding up Deep Learning Computational Aspects of Machine Learning. https://www.researchgate.net/publication/308414212_Speeding_up_Deep_Learning_Computational_Aspects_of_Machine_Learning

[36] - Musiol, M. (2016). Speeding up Deep Learning Computational Aspects of Machine Learning. [Diagram] https://www.researchgate.net/profile/Martin-Musiol/publication/308414212/figure/fig1/AS:409040078295040@1474534162122/A-general-model-of-a-deep-neural-network-It-consists-of-an-input-layer-some-here-two.png

[37] - Hosny, A. (2018, June 19). *Priming neural networks with an appropriate initializer.* Medium. https://becominghuman.ai/priming-neural-networks-with-an-appropriate-initializer-7b163990ead

[38] - Kalita, D. (2022, March 3). *Basics of CNN* [Diagram]. analyticsvidhya. https://editor.analyticsvidhya.com/uploads/89175cnn_banner.png

[39] - Ahmed, H., Wong, M., & Nandi, A. (2018). Intelligent condition monitoring method for bearing faults from highly compressed measurements using sparse over-complete features. Mechanical Systems and Signal Processing. 99. 459-477. [Diagram] https://www.researchgate.net/figure/Autoencoder-architecture_fig1_318204554

[40] - Jordan, J. (2018). *Complete Autoencoder* [Diagram]. Jeremy Jordan. https://www.jeremyjordan.me/content/images/2018/03/Screen-Shot-2018-03-06-at-6.09.05-PM.png

[41] - *Autoencoders*. (2018). [Slides]. School of Computer Science and Engineering Australia. https://www.cse.unsw.edu.au/~cs9444/17s2/lect/12_Autoencoders4.pdf

[42] - Jordan, J. (2018b). *Sparse Autoencoder* [Diagram]. https://www.jeremyjordan.me/content/images/2018/03/Screen-Shot-2018-03-07-at-1.50.55-PM.png

[43] - Feng, D., Wang, X., Wang, X., Ding, S., & Zhang, H. (2021). Deep Convolutional Denoising Autoencoders with Network Structure Optimization for the High-Fidelity Attenuation of Random GPR Noise. *Remote Sensing*, *13*(9), 1761. https://doi.org/10.3390/rs13091761

[44] - O'Shea, T., Corgan, J., & Clancy, T. (2016). Unsupervised Representation Learning of Structured Radio Communication Signals. [Diagram] https://www.researchgate.net/figure/Convolutional-Autoencoder-Architecture-Used_fig2_301816780

[45] - Nugroho, H. (2020). Fully Convolutional Variational Autoencoder For Feature Extraction Of Fire Detection System. Jurnal Ilmu Komputer dan Informasi. 13. [Diagram] https://www.researchgate.net/figure/Basic-structure-of-Variational-Autoencoder-VAE_fig2_340049776

[46] - Barisits, M., Beermann, T., Berghaus, F., Bockelman, B., Bogado, J., Cameron, D., Christidis, D., Ciangottini, D., Dimitrov, G., Elsing, M., Garonne, V., di Girolamo, A., Goossens, L., Guan, W., Guenther, J., Javurek, T., Kuhn, D., Lassnig, M., Lopez, F., Magini, N., Molfetas, A., Nairz, A., Ould-Saada, F., Prenner, S., Serfon C., Stewart, G., Vaandering, E., Vasileva, P., Vigne, R., & Wegner, T. (2019). Rucio: Scientific Data Management. *Computing and Software for Big Science*, *3*(1). https://doi.org/10.1007/s41781-019-0026-3

[47] - Labrador, H., Alexandropoulos, G., Bocchi, E., Castro, D., Chan, B., Contescu, C., Lamanna, M., lo Presti, G., Mascetti, L., Moscicki, J., Musset, P., Karavakis, E., Pelletier, R., & Valverde, R. (2019). CERNBox: the CERN cloud storage hub. *EPJ Web of Conferences*, *214*, 04038. https://doi.org/10.1051/epjconf/201921404038

[48] - Mościcki, J. (2019, November 4). *SWAN Integrating Powers* [Slides]. Indico.Cern.Ch. https://indico.cern.ch/event/773049/contributions/3476172/attachments/1937733/3211862/SWAN-CHEP2019.pdf

[49] - Merkel, D. (2014). Docker: lightweight linux containers for consistent development and deployment. Linux Journal, 2014(239), 2.

[50] - Anaconda Software Distribution. (2020). Anaconda Documentation. Anaconda Inc. Retrieved from https://docs.anaconda.com/

[51] - Van Rossum, G., Drake, F. L. (2009). Python 3 Reference Manual. Scotts Valley, CA: CreateSpace.

[52] - R. Brun, F. Rademakers, S. Panacek. (2000). ROOT, an object oriented data analysis framework. https://doi.org/10.5170/CERN-2000-013.11

[53] - McKinney, W., & others. (2010). Data structures for statistical computing in python. In Proceedings of the 9th Python in Science Conference (Vol. 445, pp. 51–56).

[54] - Caswell, T., Droettboom, M., Lee, A., Sales de Andrade, E., Hoffmann, T., Hunter, J., Klymak, J., Firing, E., Stansby, D., Varoquaux, N., Nielsen, J., Root, B., May, R., Elson, P. Seppänen, J., Dale, D., Lee, J., McDougall, D., Straw, A. Hobson, P., Hannah, Gohlke, C., Vincent., A., Yu, T., Ma, E., Silvester, S., Moad, C., Kniazev, N., Ernest, E., & Ivanov, P. (2021). matplotlib/matplotlib: REL: v3.5.1 (v3.5.1). Zenodo. https://doi.org/10.5281/zenodo.5773480

[55] - Waskom, M., (2021). seaborn: statistical data visualization. Journal of Open Source Software, 6(60), 3021, https://doi.org/10.21105/joss.03021

[56] - Bayer, M. (2012). SQLAlchemy. In A. Brown & G. Wilson (Eds.), The Architecture of Open Source Applications Volume II: Structure, Scale, and a Few More Fearless Hacks. aosabook.org. Retrieved from "http://aosabook.org/en/sqlalchemy.html"

[57] - Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., & others. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12(Oct), 2825–2830.

[58] - Chollet, F., & others. (2015). Keras. GitHub. Retrieved from https://github.com/fchollet/keras

[59] - Dash Documentation https://dash.plotly.com/ ; Inc., P. T. (2015). Collaborative data science. Montreal, QC: Plotly Technologies Inc. Retrieved from https://plot.ly

[60] - Zhao, Y., Nasrullah, Z., & Li, Z. (2019). PyOD: A Python Toolbox for Scalable Outlier Detection. *Journal of Machine Learning Research*. https://www.jmlr.org/papers/volume20/19-011/19-011.pdf

[61] – The ATLAS Collaboration. (2018). *Glossary · GitBook*. Opendata.Atlas.Cern. http://opendata.atlas.cern/books/current/openatlasdatatools/_book/glossary.html

[62] - *ROOT: Histogram classes.* (2016). Root.Cern.Ch. https://root.cern.ch/doc/master/group__Histograms.html

[63] – The ATLAS Collaboration. (2009). *atlasdqm web display tier 0*. Atlasdqm. https://atlasdqm.cern.ch/webdisplay/tier0/

[64] - Daneshi, M., Guo, J. (2011). Image reconstruction based on local feature descriptors. https://stacks.stanford.edu/file/druid:my512gb2187/Daneshi_Guo_Image_Reconstruction_from_Descriptors.pdf

[65] - Popovic, D., Fouché, E., & Klemens, B. (2019). Unsupervised Artificial Neural Networks for Outlier Detection in High-Dimensional Data. https://doi.org/10.1007/978-3-030-28730-6_1.

[66] - Merrill, N., Eskandarian, A. (2020). Modified Autoencoder Training and Scoring for Robust Unsupervised Anomaly Detection in Deep Learning. IEEE Access. PP. 1-1. https://ieeexplore.ieee.org/document/9099561

[67] - Astrid, M. Zaheer, Z., Lee, J., & Lee, S. (2021). Learning Not to Reconstruct Anomalies. https://arxiv.org/abs/2110.09742

[68] - Franco, E., Rana, P., Cruz, A., Calderon, V., Azevedo, V., Ghosh, P., Ramos, R. (2021). Performance Comparison of Deep Learning Autoencoders for Cancer Subtype Detection Using Multi-Omics Data. https://www.mdpi.com/2072-6694/13/9/2013

[69] - Brownlee, J. (2019, August 12). *Embrace Randomness in Machine Learning*. Machine Learning Mastery. https://machinelearningmastery.com/randomness-in-machine-learning/

[70] - Brownlee, J. (2019b, August 19). *How to Get Reproducible Results with Keras*. Machine Learning Mastery. https://machinelearningmastery.com/reproducible-results-neural-networks-keras/

[71] - *If Keras results are not reproducible, what's the best practice for comparing models and choosing hyper parameters?* (2019, November 27). Stack Overflow. https://stackoverflow.com/questions/59075244/if-keras-results-are-not-reproducible-whats-the-best-practice-for-comparing-mo

[72] – *How to get reproducible results in keras.* (2020, February 7). Stack Overflow. https://stackoverflow.com/questions/32419510/how-to-get-reproducible-results-in-keras

[73] – *Reproducible results neural networks keras.* machinelearningmastery. https://machinelearningmastery.com/reproducible-results-neural-networks-keras/

[74] - Ting, J., D'Souza, A., & Schaal, S. (2007). Automatic Outlier Detection: A Bayesian Approach. Proceedings - IEEE International Conference on Robotics and Automation. 2489-2494. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.75.9906&rep=rep1&type=pdf

[75] - Gao, J. Tan, P. (2006). Converting Output Scores from Outlier Detection Algorithms into Probability Estimates. 6th IEEE International Conference on Data Mining (ICDM 2006). 212-221. http://www.cse.msu.edu/~ptan/papers/ICDM2.pdf

[76] - Pol, A., Cerminara, G., Germain, C., Pierini, M., & Seth, A. (2018). Detector monitoring with artificial neural networks at the CMS experiment at the CERN Large Hadron Collider. https://cds.cern.ch/record/2683825/files/Pol2019_Article_DetectorMonitoringWithArtifici%20(1).pdf

[77] - Goldstein M., Uchida S. (2016) A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data. *PLoS ONE* 11(4): e0152173. https://doi.org/10.1371/journal.pone.0152173

[78] - Fawcett, T. (2006). Introduction to ROC analysis. *Pattern Recognition Letters*. 27. 861-874. https://people.inf.elte.hu/kiss/13dwhdm/roc.pdf

[79] - *Dash Overview*. (2020). Dash Documentation. https://plotly.com/dash/

[80] - *ROC Example*. (2018). [Graph]. http://algolytics.com/wp-content/uploads/2018/05/roc1_en.png

[81] - Wicklin, R. (2011, July 29). *Computing an ROC curve from basic principles*. The DO Loop. https://blogs.sas.com/content/iml/2011/07/29/computing-an-roc-curve-from-basic-principles.html

[82] - Krzanowski, W., Hand, D. (2009). ROC Curves for continuous data. https://www.researchgate.net/publication/268067577_ROC_Curves_for_continuous_data

[83] – Hosmer Jr., D. (2013). *Applied Logistic Regression* (3rd ed.). Wiley.

[84] – The ATLAS Collaboration. (2018b). *Run 359872, 1/physics_Main CaloMonitoring/CaloMonExpert/EMTopoCluster/ClustersForExpert/TransEnergy/etaphi_thresh_avgEt_0*. Atlasdqm Web Display. https://atlasdqm.cern.ch/webdisplay/tier0/1/physics_Main/run_359872/run/CaloMonitoring/CaloMonExpert/EMTopoCluster/ClustersForExpert/TransEnergy/etaphi_thresh_avgEt_0

[85] - Passalis, N., Tefas, A., Kanniainen, J., Gabbouj, M., & Iosifidis, A. (2019). Deep Adaptive Input Normalization for Price Forecasting using Limit Order Book Data. https://deepai.org/publication/deep-adaptive-input-normalization-for-price-forecasting-using-limit-order-book-data