



Anomaly detection using an Autoencoder for the High-Granularity Calorimeter

CERN Summer Student Report 2024

Summer Student:

Lam Giang Tran^{*}

Technische Hochschule Mittelhessen (THM)

Supervisors:

Chiara Amendola, Pedro Silva

Conseil européen pour la recherche nucléaire (CERN)

^{*}corresponding email adress: lam.giang.tran@lsc.thm.de



Contents

1	Introduction	3
1.1	Experimental Environment	3
1.2	Aim of this CERN Summer Student Project	3
2	Theoretical fundamentals	4
2.1	Proton-Proton Collisions at the LHC	4
2.2	High-Granularity Calorimeter	5
2.3	ROOT Data structure	6
2.4	Autoencoder	6
3	Anomaly Detection Strategy	7
3.1	Training Data preparation	7
3.2	Anomaly Detection Implementation	9
3.3	Autoencoder Training	10
4	Results	11
4.1	Training Performance	11
4.2	Reconstruction Performance	11
4.3	Conclusion	12
5	Summary	13
	References	14

1 Introduction

In order to answer essential scientific questions regarding to the structure of matter and fundamental physics, high-energy physics research is carried out at the *European Organization for Nuclear Research (Conseil européen pour la recherche nucléaire - CERN)*, using large-scale particle accelerators to collide particles and measuring the resulting subparticles by big detector complexes.

1.1 Experimental Environment

The so called *Large Hadron Collider (LHC)* is one such accelerator at CERN, providing four crossing points where the particles collisions occurs. At one of these collision points, the *Compact Muon Solenoid (CMS)* is located to provide multi-purpose experiments, like the investigation of the Higgs sector physics, measurements for the Standard Model or the search beyond the Standard Model. The hermetic CMS detector is constructed in several sub-systems, which enables five-dimensional measurements of the resulting particles.¹

In the forward region of the CMS experiment, the detector end cap is located, consisting currently of an electromagnetic and a hadronic calorimeter. The planned LHC upgrade to the *High-Luminosity LHC (HL-LHC)* will result into a qualitative and quantitative increase in collision-rate, meaning that the detector complex will have to deal with higher center-of-mass energies and a higher number of collisions. It also leads to more *event pileups*, which refers to the circumstance where multiple collisions occur during the same data-taking within the CMS detector. This makes it more challenging to isolate the data-taking from the collision of interest.

Therefore the current detector component must be replaced with the new *High-Granularity Calorimeter (HGCAL)*, to ensure that the material, the electronics and the readout sensors are able to withstand the high radiation dose and to differentiate distinct events being recorded simultaneously within the detector. By substituting the current endcap from the CMS system with the HGCAL, the new calorimeter is able to provide a finer granularity to discriminate pileups and resist the ten times harsher radiation environment of the HL-LHC. [1, 2]

1.2 Aim of this CERN Summer Student Project

The aim of this Summer Student project is the implementation and testing of an **anomaly detection** system, based on an artificial neural network. Deep Learning techniques are currently used in both detector as well as high-energy physics, and are useful approaches for anomaly detection in particle physics.

This project is focused on the **autoencoder (AE)** implementation and testing for the silicon sensors. The AE is designed to enable the possibility to detect anomalies by compressing and decompressing the measurement signal. As a result, this development could improve the process of testing and validating of the test beam Runs before the upgrade. The following chapters will provide the reader with a basic understanding of the functionality and development of this summer student project.

¹The five information are the three location coordinates x , y , z , as well as the time t and energy E .

2 Theoretical fundamentals

To comprehend the project development during this summer school period, a helpful background is presented in the following chapters. The relevant knowledge ranges in the field of *detector physics*, *data structures* and *deep learning* basics, with the aim to provide a helpful foundation for possible further developments.

2.1 Proton-Proton Collisions at the LHC

The first proton-proton collision at the LHC has occurred 2009 with a center-of-mass energy of 0.9 TeV. 2010, the accelerator operated during the first Run at 7 and 8 TeV, increasing the center-of-mass energy to 13 TeV at the second Run. The ongoing third Run provides energies further up to 13.6 TeV, started in 2022.

During each Run, each proton beam with 2500 packed bunches of $\sim 10^{11}$ protons within 25 ns bunch spacing are traveling through the LHC beam pipe. In order to provoke a particle-particle collision, the beams are accelerated in opposite directions and deflected to the collision points, where the center of the CMS detector is located. This interaction between two protons is referred as an *event*, resulting to a production of hundreds of different sub-particles.

The produced subatomic particles then interact with the CMS detector (presented in Figure 1), where the particle information is measured by the concentric layers of the sub detector components, like the HGCAL. [3]

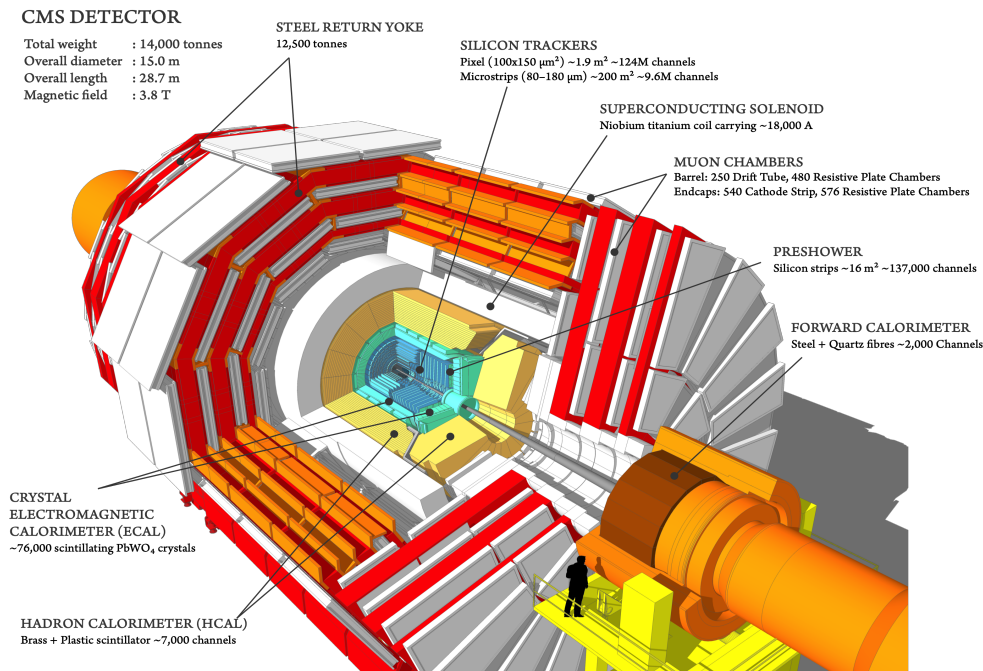


Figure 1: View of the *Compact Muon Solenoid* with the current forward end cap, consisting of a separate *electromagnetic calorimeter* and a *hadron calorimeter*. [3]

2.2 High-Granularity Calorimeter

The whole calorimeter (Figure 2a) is thermally shield and is maintained at $-30\text{ }^{\circ}\text{C}$. The longitudinal structure of the HGCAL consists of two different compartments, the *electromagnetic component (CE-E)* followed by the *scintillator layers* within the *hadronic component (CE-H)* (compare to Figure 2b).

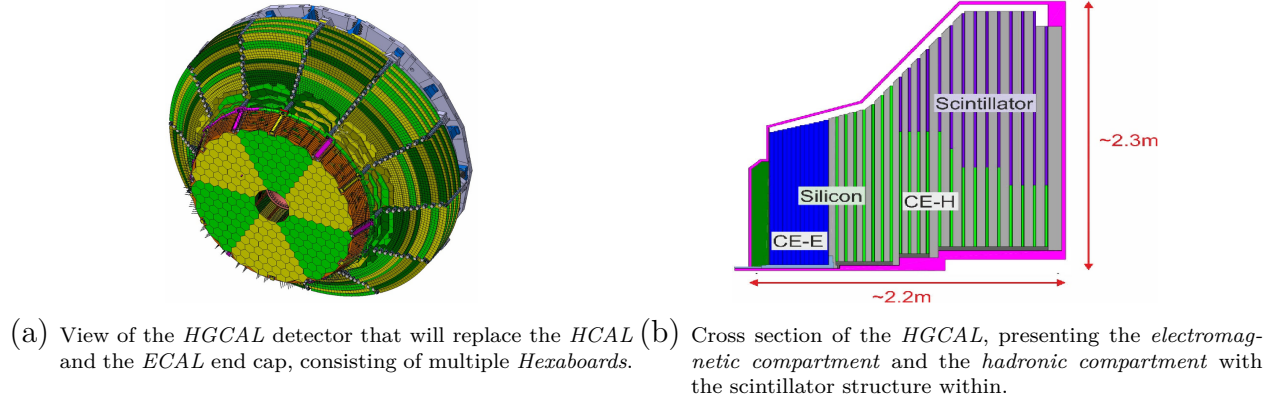


Figure 2: Display of the HGCAL endcap [3]

The CE-E is build in several circular layers, in which multiple *hexagonal silicon sensors* are embedded, as shown in Figure 3a. Two different radii are defined in order to distinguish between the different silicon densities, providing cells of approximately 0.5 and 1.1 cm^2 . The planar silicon sensors are sandwiched between a Cu-W baseplate and a printed circuit board, forming the composition into a *silicon module*. The CE-E module is a stack of four different components: the Cu-W baseplate, the Kapton-gold sheet, the silicon sensor and the printed circuit board (Figure 3b).

The silicon sensor and the circuit board are in a hexagonal shape, in order to use the available area of the layer from the CE-E more efficiently. Each so-called *hexaboard* is provided with a HGROC front-end readout ASIC. They provide three measurements: the *ADC* charge, the *Time of Arrival (ToA)* and the *Time over Threshold (ToT)*. The information are stored at 40 MHz in a 512 buffer (compare to Figure 4). Each HGROC is capable to read out 78 channels, although 4 channels are only in use for common-mode noise estimation, meaning that 74 channels from the HGROC are used for this summer student project. It should also be pointed out, that only the ADC values are used in this project.

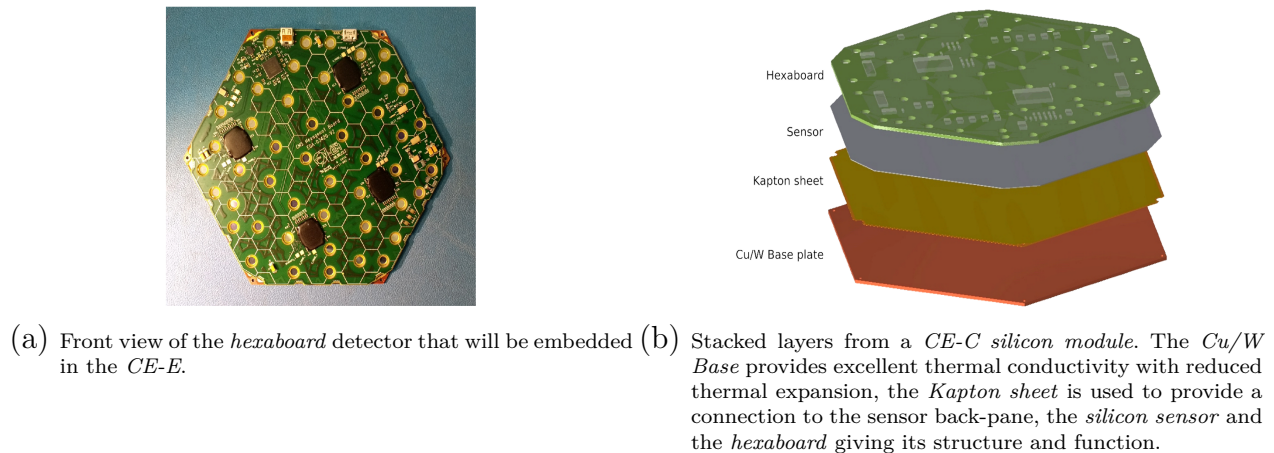


Figure 3: Display of the Hexaboards [3]

The measurements and data from the 74 channels of the hexaboard, which are embedded in several circular layers of the CE-E, and these in turn are embedded in the HGCAL subsystem of the CMS, are providing the data set for this AE development, stored in the *ROOT DataFrame*. [1, 3, 4]

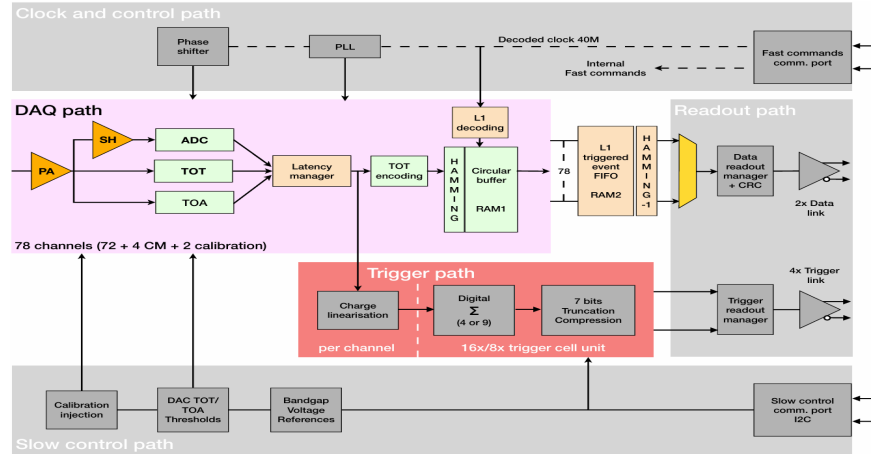


Figure 4: Front-end ASIC HGROC chip for measuring the charge and time. The *ADC*, *ToT* and *ToA* are transmitted to a concentrator ASIC, linking electrically via 1.28 Gb/s for each HGROC chip. [5]

2.3 ROOT Data structure

ROOT is a software framework developed at CERN for data analysis, mostly used in High-Energy Physics. Its object format provides a structure to deal with a high quantity of data for easy data access and data saving.

The events generated during a test beam Run are stored in a binary format into the *ROOT file* system, calling it a *tuple*. Its hierarchical data container are based on a *tree structure*, storing all relevant information including the already mentioned *ADC*, *ToT* as well as the *ToA* data in one single *ROOT file*.

By accessing each sub storage systems in a *TBranch structure* and using the *uproot* Python package, the relevant information like the *ADC* or the channel quantity can be read out into a *pandas DataFrame*. The advantage of using the *ROOT* hierarchy structure is having no needs to allocate memory and avoiding object creations. However, it requires just a one-time conversion of a *ROOT TTree-file* via a *panda DataFrame* into multiple *numpy Arrays*.

The subsequent conversion into a *numpy array* allows the easy manipulation and processing in *python*, but this requires a conversion in multiple batches, implemented in loops which results into a longer conversion time.

2.4 Autoencoder

The fundamental structure of an AE lies within their artificial neural network, making it to a special subclass of a *deep learning* system.

The basic idea of a computational neural network is to calculate the weights between the input and output layers. By applying the *backpropagation* technique to train the AE, the system minimizes the prediction error by adjusting the weights and biases between the layers of the hidden layer. [6, 7]

This particular deep learning system is designed for unsupervised learning and is used to transform the information in the input layer into a reduced representation in a smaller dimension, called a “bottleneck”. This process is referred as the “latent space” or “encoding”, which is why the first part of the AE is also called the *encoder*. Its complementary *decoder* reconstructs this reduced representation from the *bottleneck* into its original dimension. To enable the network to recognize meaningful patterns - or anomalies in this use case - a process of concatenated encoding and decoding facilitates the learning of the essential features.

The goal of such system is therefore to reconstruct the high-dimensional data set based on the reduced dimensions based on it’s trained feature, making it to a useful tool for anomaly detection.² A representative example of such an architecture is shown in figure 5. [8]

²Popular use cases of the AE are also noise reduction and classification.

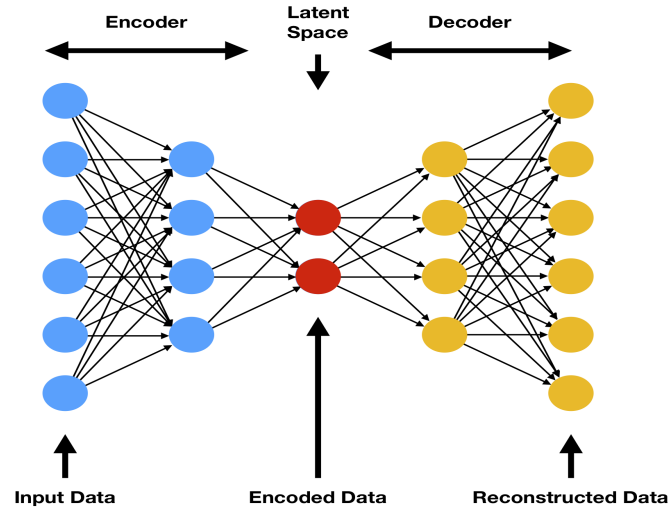


Figure 5: Schematic structure of an AE, combining the *Encoder* for the input data compression and the *decoder* for data reconstruction, based on the encoded data in the *latent space*.

3 Anomaly Detection Strategy

Since the test runs provide a large number of *events*, hence a large amount of data and information, it appears likely to develop a deep learning system for the anomaly detection. Earlier results and publications has shown the feasibility and potential of AE for this task, tested on the predecessor of the HGCal [9, 10, 11]. By using an unsupervised approach, the system is able to detect anomalies without the need of labeled data and introducing it to data with anomalies.

The data preparation and manipulation of the event as well as the development, training and validation of this AE environment are performed in *CERN SWAN*, a online platform providing **S**ervice for **W**eb based **A**Nalysis. All *python3* developments as well as resulting data and models are saved in the [CERNBox](#) as well as in [Gitlab](#).

3.1 Training Data preparation

For the data preparation the procedure is visualized in the Figure 6. The data preparation consists of following steps:



Figure 6: Visualized Steps for the Data preparation for the AE-Training.

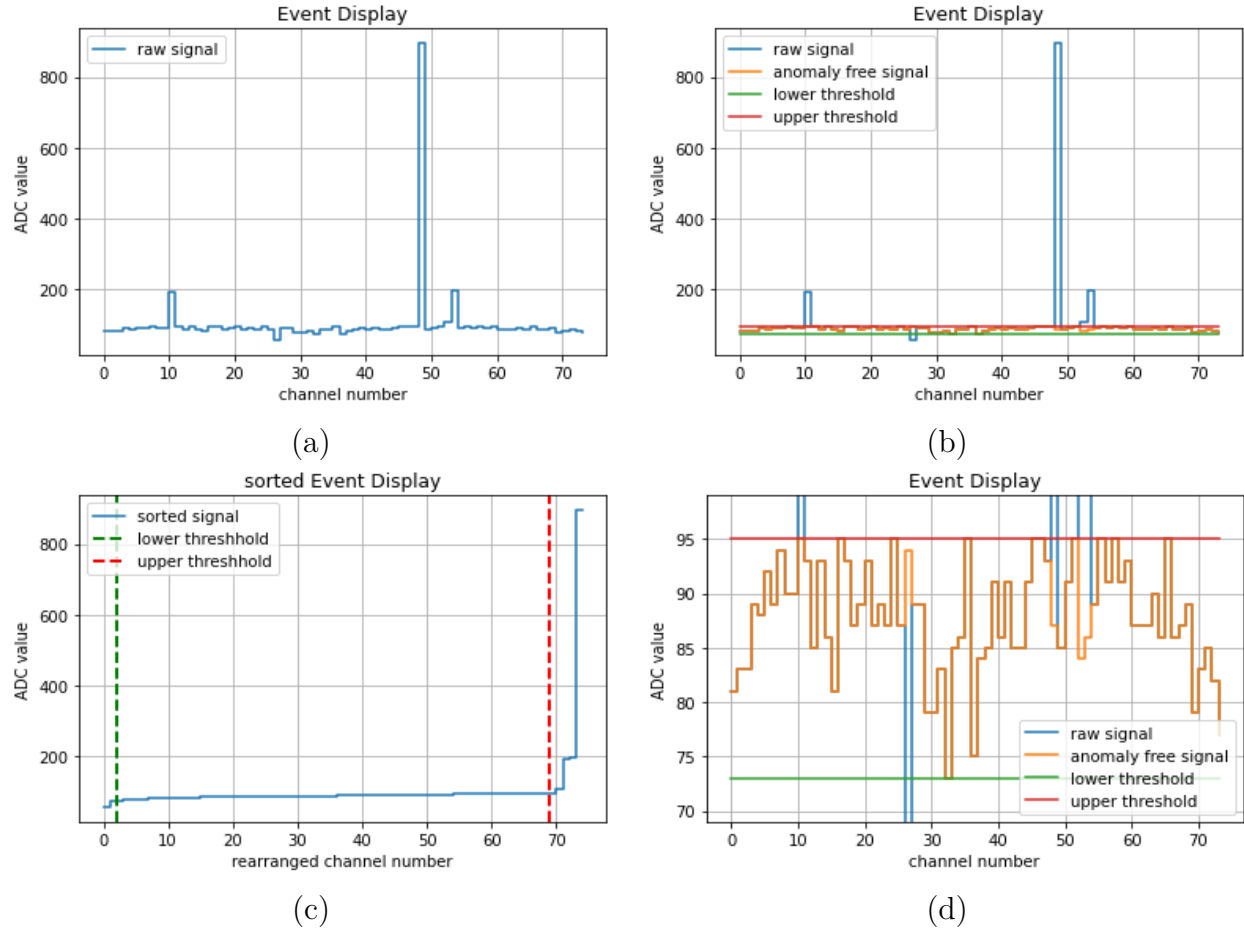


Figure 7: (a) Raw signal (unprocessed event). Those events are used for data preparation and AE training.
(b) Processed Signal (with substituted pedestal values).
(c) Event sorted by ADC value (pedestal). Values fluctuating outside a range are defines as outliers.
(d) Zoomed processed Signal.

The first step of the data preparation is the extraction of the events information from the ROOT storage using the python package *uproot*. By doing this, the event data are stored in a .ROOT-file. To extract the relevant information from the detector, the corresponding segment of the HGROC must be first defined and selected. For this usecase, one segment from the HGROC (0 and 1 has been selected). The way of definition and calling of the HGROC are documented on this [webpage](#)³. One example for such event in displayed in the Figure 7a.

For the data selection, one million events from one single HGROC, equally to 74 channels, was extracted, providing a dataset of 74.000.000 values. After extracting the data, the .ROOT-file is converted into a *numpy*-array for the easy use of data processing and array manipulation. This will also allow the usage of the data-format for the AE training.

After extracting and converting the data, two thresholds - an upper and a lower threshold - are set for the anomalies for each single event. The thresholds are set a priori with a pedestal difference bigger than five. Since the review and statistical pre-analysis on the mean and standard deviation, the values for setting thresholds a set at a step difference of five (compare to Figure 7c). This step is processed by sorting each event individually by pedestal value and determine any signal differences bigger five. The determined values are set as the lower and upper threshold values at these two points.

³courtesy of Pedro Silva.

Once the thresholds are set for each event, each pedestal of every single event is compared with the event specific threshold by a simple boolean function. Values beyond the scope of validity are then marked for processing. All values outside this range⁴ are substituted with a random generated noise, based on the Gaussian distribution, as seen in picture 7b.

Those marked values respectively the channels will be substituted by a simulated noise value, generated by a random number generator based on the Gaussian distribution. The *grand-mean* and *grand-standard deviation* are calculated by the mean value over all events, excluding the marked values in each event. This two grand statistical parameters define the μ and σ of the Gaussian distribution for the substitution values. By doing this procedure over all events individually, a systematical bias can be eliminated.

After all anomalies are substituted, the cleaned events are saved in a .ROOT-Dataset for the use of training. They are stored in a random order to ensure an even distribution in a set of training and test data.

3.2 Anomaly Detection Implementation

For the anomaly detection method, an unsupervised AE was build with totally twelve hidden layers. The AE architecture is implemented using *PyTorch* and includes an encoder, decoder, and a bottleneck layers based on four nodes. To quantify the difference between the input file and the output file, the reconstruction loss is calculated using the *Mean Squared Error (MSE)* of the input data x and the reconstructed output x' , according to equation

$$L(x, x') = ||(x - x')||^2. \quad (1)$$

In both the encoder and the decoder, the hidden layers compresses and the input data into the bottle neck and decompressed the latent space using six multiple *affine linear transformations* according to the equation

$$y = xA^T + b, \quad (2)$$

where T contains the weights and b the bias. Each layer is followed by a *Rectified Linear unit function*, defined as

$$ReLU(x) = \max(0, x). \quad (3)$$

As the activation function, the *sigmoid function* from equation 4 was used in the penultimate layer.

$$S(x) = \frac{1}{1 + e^{-x}} \quad (4)$$

Those transformation are implemented in the *PyTorch* library [12]. The input layers consists of 74 nodes, corresponding to the ADC values of one single event. The dimension is then expanded from 74 to 148 nodes, and then continuously halved until the neck bottle has an account dimension of 2. The decoder then reconstructs the compressed information in the reverse direction back to the original dimension.

⁴This is for this use case and training defined as Anomaly.

3.3 Autoencoder Training

Based on the AE implementation described in the previous chapter, the deep learning system is trained with a dataset based on a test Run from 2023, in which the HGCAL is tested with a pedestal Run. The training dataset are the prepared events declared as anomaly free, in which the outliers were substituted by the generated noise signal (compare to chapter 3.1). The total event batch was split into a training dataset consisting 600.000 events, a validation dataset of 300.000 and a test dataset consisting 100.000 events.

For the training, following parameters has been set:

- batch size: 4096
- shuffle parameter: true
- training quantity: 600.000 events
- optimizer: Adam optimizer
- learning rate: 1e-3
- weight decay: 1e-11
- epoch number: 50
- loss function: MSE Loss function

All the processed and unprocessed Training data has to be transferred to the AE by creating a *DataLoader*. PyTorch's DataLoader enables simple processing of large amounts of data and access to functions such as batching or shuffling. For the training process, the training efficiency has been increased by moving the dataset as well as the AE model to the *graphic processing unit* (GPU) of *SWAN*, using the version *105a Cuda 11.8.89*. The *Compute Unified Device Architecture* (CUDA) from NVIDIA allows a parallel computing and faster training progress, based of the GPU.

4 Results

To verify the functionality of the AE, a simple analysis based on histogram examination has been performed. To investigate the training quality of the AE, the loss functions of training and validation data are analyzed and compared.

4.1 Training Performance

For the Analysis of the Training performance, the values of the lost function are plotted for each epoch-Run during the training session. The same procedure has been performed during the test session.

Since the training loss function quantify the dissimilarity between the input and output value, the loss curve are giving insights about the improvement of the model performance over epoch respectively over time. The loss function is normalized by dividing Equation 1 by the number of channels.

As shown in picture 8, the Loss Function decreases from 1.017 to 1.005, showcasing that the AE is able to learn from the Training Data and minimizing the differences from input- and output layer by optimizing the connection between the layers. The Loss function shows a high fluctuation, which can be attributed to possible instability during learning. The AE reached it's plateau of training at a epoch of 40, where an average loss of 1.005 is within the stochastic noise of each individual channel.

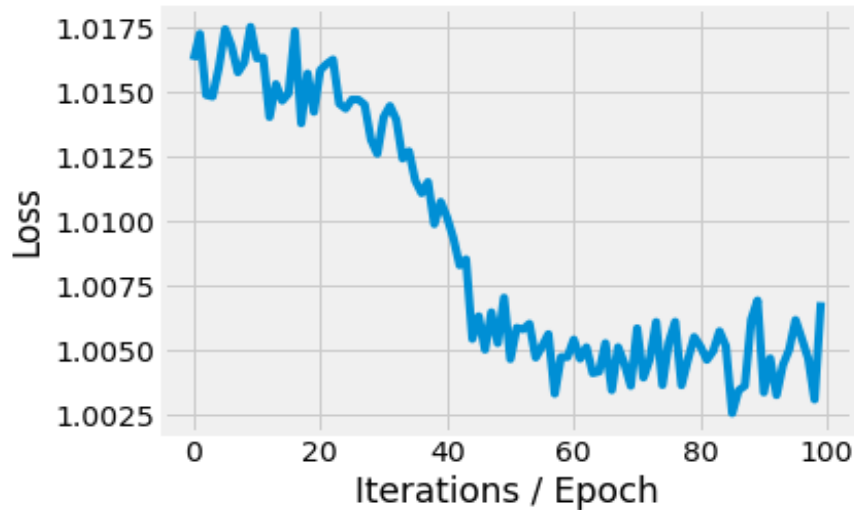


Figure 8: Normalized *Training Loss function* decreasing with each Epoch, showing the progress of reconstructing the input signal.

4.2 Reconstruction Performance

As already discussed in chapter 4.1, the training loss decreases with each epoch while the validation loss reaches a plateaus, suggesting that the AE may be over fitting. The adapted AE consisted of 14 layers, reducing with each layer the node size by 10 nodes.

It could have happened, that the AE has become too specialized with optimizing of the neural network and fitting to much the weights and biases. The model fails to generalize the new (validating) input data, as seen in the validation loss function in Figure 9.

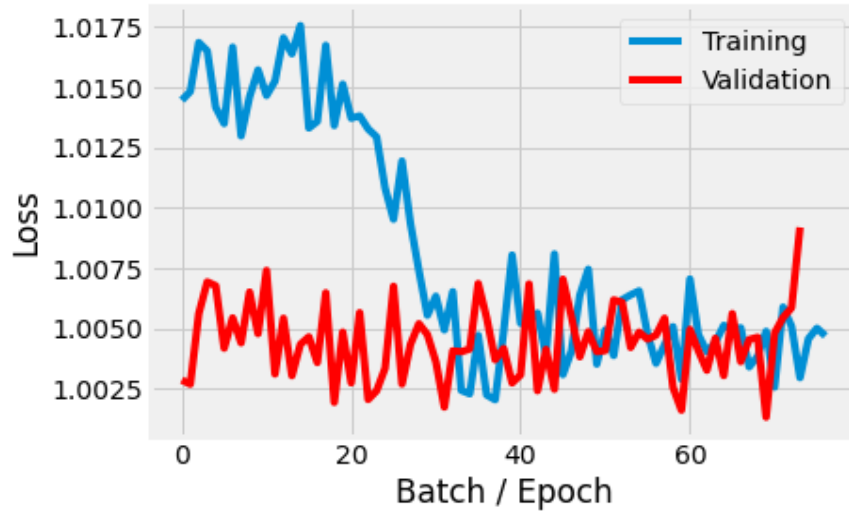


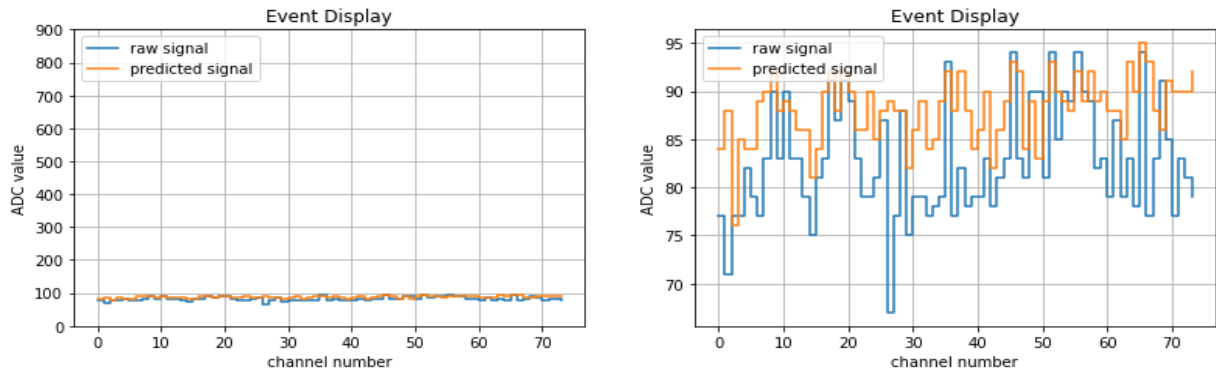
Figure 9: Progress in the *Training Loss Function* but the *Validation Loss Function*, showing the current issue of the AE to generalize the new input data.

4.3 Conclusion

Based on the loss function, a training success is possible through the processed data. The decrease in the loss curve shows the learning process over time respectively epoch. In order to be able to train the AE appropriately, a high number of possible training data was selected (600,000 events). In the course of the AE implementation, the focus was also placed on a simple hiding layer and a simple parametrization was taken into account.

Nevertheless, there is an overfitting according to the interpretation of the various loss functions. Further work should be done on this challenge in order to be able to use the system effectively for anomaly detection. A reduction in the learning rate and a possible adjustment of the regulation are recommended here. It may also be useful to prepare the training data with a greater fluctuation and clearer recognition features so that the AE can extract more structures and information. It has also maybe occurred because the validation dataset could had too few data examples. Ultimately, however, this minimum viable project showed that an AE for the HGCAL silicon sensors is possible, useful and further developed.

The AE is able to reconstruct the information from a processed signal, as seen in Figures in 10.



(a) Comparison of the processed data and it's reconstruction. (b) Zoomed comparison of the input and output data.

Figure 10: Different comparisons of the Loss Functions.

5 Summary

During the CERN Summer Student period, an Autoencoder for the Anomaly Detection has been successfully implemented. The development environment was the CERN SWAN framework, in which all the implementation has occurred and are all available on [Gitlab](#) as well in the [CERNBox](#) (see section 3).

The development of the Anomaly Detection Strategy began with the data preparation, using numerically determined thresholds to spot outliers. The determined outliers were manipulated with a Gaussian-based random value. This Gaussian-based value were calculated by the grand-mean value as well as the grand-standard deviation value over each channel for every single event.

For the for the AE Training, 60 % of the total prepared dataset were used and 30 % of the prepared data for validation of the deep learning system. The remaining 10 % were reserved unprocessed for possible testing the system.

The AE implementation was based on PyTorch, an open-source machine learning framework. The structure of the deep learning system consisted of 18 layers, which were connected by a combination of a linear transformation functions and a rectified linear unit functions. The encoder compresses the input layer into a two nod sized latent space, the optimization was based on the MSE loss function.

The investigation was based on a trivial Loss Function review. This showcased the usefulness, feasibility and efficiency of this AE implementation. But the principle of this AE development should be also tested for further use cases on different types of detector systems, multiple subatomic particle interactions as well as different deep learning structures. It still needs some optimization of the hyperparamters, to fully unlock the potential of the AE.

This brief Summer Student project seeds the investigation of Machine Learning based anomaly detection tools for data quality monitoring of HGCAL. The HGCAL components were presented as an extremely complex detector sub system, part of the overall research complexity at the LHC, but worth every effort in the fundamental endeavor to understand the fundamental physics at CERN. Interdisciplinary cooperation between a wide range of disciplines and international exchange between different nations is the way forward, as practiced here at CERN.

References

- [1] *The Phase-2 Upgrade of the CMS Endcap Calorimeter*. Tech. rep. Geneva: CERN, 2017. DOI: [10.17181/CERN.IV8M.1JY2](https://cds.cern.ch/record/2293646). URL: <https://cds.cern.ch/record/2293646>.
- [2] G L Bayatian et al. *CMS Physics: Technical Design Report Volume 1: Detector Performance and Software*. Technical design report. CMS. Geneva: CERN, 2006. URL: <https://cds.cern.ch/record/922757>.
- [3] Manfred Valentan. “The CMS high granularity calorimeter for the high luminosity LHC”. In: *Nucl. Instrum. Methods Phys. Res., A* 936 (2019), pp. 102–106. DOI: [10.1016/j.nima.2018.10.131](https://cds.cern.ch/record/2718194). URL: <https://cds.cern.ch/record/2718194>.
- [4] Yung-Wei Chang. “Construction and beam-tests of silicon and scintillator-SiPM modules for the CMS High Granularity Calorimeter for HL-LHC”. In: *Nucl. Instrum. Methods Phys. Res., A* 924 (2019), pp. 301–304. DOI: [10.1016/j.nima.2018.06.037](https://cds.cern.ch/record/2717420). URL: <https://cds.cern.ch/record/2717420>.
- [5] CMS Collarboation Chiara Amendola. *An overview of the CMS High-Granularity Calorimeter*. 2024. URL: https://indico.cern.ch/event/1284854/contributions/5951339/attachments/2887956/5061901/iWoRiD_2024_07_01_amendola_v3.pdf.
- [6] Goodfellow Ian, Bengio Yoshua, and Courville Aaron. *Deep learning: Adaptive computation and machine learning*. 2017.
- [7] Sridhar Alla and Suman Kalyan Adari. *Beginning anomaly detection using python-based deep learning*. Springer, 2019.
- [8] Wei Qi Yan. *Computational methods for deep learning: theory, algorithms, and implementations*. Springer Nature, 2023.
- [9] D. Abadjiev et al. “Autoencoder-Based Anomaly Detection System for Online Data Quality Monitoring of the CMS Electromagnetic Calorimeter”. In: *Computing and Software for Big Science* 8.1 (June 2024). ISSN: 2510-2044. DOI: [10.1007/s41781-024-00118-z](http://dx.doi.org/10.1007/s41781-024-00118-z). URL: <http://dx.doi.org/10.1007/s41781-024-00118-z>.
- [10] D. Abadjiev et al. “Autoencoder-Based Anomaly Detection System for Online Data Quality Monitoring of the CMS Electromagnetic Calorimeter”. In: *Computing and Software for Big Science* 8.1 (June 2024). ISSN: 2510-2044. DOI: [10.1007/s41781-024-00118-z](http://dx.doi.org/10.1007/s41781-024-00118-z). URL: <http://dx.doi.org/10.1007/s41781-024-00118-z>.
- [11] Alkis Papanastassiou and Valentina Gori. “AutoEncoders for per-lumisection data quality monitoring at CMS”. In: Jan. 2024, p. 284. DOI: [10.22323/1.450.0284](https://arxiv.org/abs/1912.01703).
- [12] Adam Paszke et al. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. 2019. arXiv: [1912.01703](https://arxiv.org/abs/1912.01703) [cs.LG]. URL: <https://arxiv.org/abs/1912.01703>.

Acknowledgments

This unique experience as a CERN Summer Student hasn't been able and so unforgettable without many people, who I cannot thank all enough and not mention all names, but to mention a few:

First of all, I would like to thank *Prof. Uli Weber* and *Prof. Andy Maun* for supporting me with this opportunity and making a childish dream become unbelievable true!

Without the fundamental academic experience in this field of *Uli*, as well as the full support and the unconditional trust in my abilities from *Andy*, I would not have had the courage to apply for this wonderful Summer Student program at a such unique place like CERN. I am looking forward to keep continuing our collaborations in any kind of research and working with you guys.

Danke euch beiden dafür!

My sincere gratitude goes to my lovely two supervisors during my stay here at CERN: I am thanking deeply *Pedro Silva* for this fascinating and challenging project as well as his academically guidance in the field of deep learning as well as data processing.

The same gratitude goes to *Chiara Amendola*, who not only supported me in every way throughout the project process, but also introduced me deeply to the beautiful world of the CMS detectors and life at CERN.

The unlimited amount of patience, support and kindness of both never ceases to amaze me, and without you two I could not have had this once-in-a-lifetime-experience at all. Thank you for this amazing experience and reinforcing my desire to continue my personal journey in research. I have been blessed to have you both as my supervisors!

Muito obrigado *Pedro*, grazie mille *Chiara*!

I would specially like to thank the wonderful administrative organization of the whole Summer Student Office, without which the whole stay would not have been so smoothly.

Děkuju *Tereza*, paldies *Anastasija*!

Besides of academically and scientific fun I had so far, this experience at CERN wouldn't be the same without the mandatory times beyond the CERN office:

I would like to thank *Janek and Robin* for the many and relaxing coffee dailies. Besides of diverse talks and funny times, the simple breaks would not have been that entertaining and memorable as it would be.

Shoutout to my companions, consisting of *Nicolas, Penelope and Yukai*, for the beautiful once in the lifetime near death experience. It was lovely to enjoy the nature and the life beyond CERN with you guys.

Thank you *Mei and Louis* for being such wonderful roommates and neighbors at the Schumann residence. The talks and collective cooking will be missed, as well as the shared living with you guys.

I would like to take this opportunity to thank *Theresa* for the lovely evenings and take-offs during our stay. The inspiring talks and joyful movie-nights have been a pleasant balance to the intense times at CERN.

Many thanks to *Christine* for introducing the beloved Austria culture as well as *Zsófia* for representing with me the research field of medical physics, and I am looking for a future collaboration with you.

Cheers to *Bernardo* for welcoming me to the CERN DAC society as well as the shared knowledge and experience. It has been a pleasure to finally meet you again after all the time!

I am also thankful for *all my beloved friends*, who are always taking care of me and being supportive during my time abroad and in absence. Your understanding, patience and faithfulness are beyond this world and I am grateful to having you in my life!

I am indebted to *my family*, my parents and my brothers for reminding me of the journey and challenges I have already mastered and overcome, and that no idealistic plans or dreams are limited.

Cảm ơn *bố, mẹ* và các *anh em*!

Last not but least: *Paula*, this work and accomplishment is devoted to you..!