Final Project:
Evaluating the Accuracy of Shifters in CMS Offline Tracker Data Certification

ICOM6015 - Artificial Neural Networks

Roy F. Cruz

November 17, 2023

Abstract

This report presents an analysis of human shifter accuracy at the lumisection level in the Offline Tracker Data Certification (DC) process for the CMS experiment at CERN. With a convolutional autoencoder ensemble, we examined this model's ability to reconstruct lumisection level monitoring element histograms from proton-proton collision runs from 2017. The goal was to determine the extent to which anomalous lumisections, overlooked by shifters due their low granularity in evaluations, are certified as good. We found that while the ensemble could reliably reconstruct monitoring element histograms, its performance showed a dramatic drop for runs that were temporally distant from the training reference. This highlighted the need for dynamic machine learning models which can adapt to the continually changing data taking conitions within CMS. For closer runs, potentially anomalous histograms were observed, highlighting the potential for undetected anomalies due to the low granularity of the human-driven DC approach. This underscored the critical role machine learning can have in enhancing DQM by enabling more granular evaluations.

1 Introduction

The Large Hadron Collider (LHC) is the world's largest and most powerful particle collider in the world [1]. It consists of a 27-kilometer ring along which superconducting magnets accelerate protons or heavy ions close to the speed of light in two oppositely directed beams. Along certain points of the ring, these beams are made to cross in order for these fast moving particles to collide. The resulting interactions products a myriad of highly energetic particles. At each of these collision points, detectors collect information about these particles, allowing us to discover some of the most fundamental aspects of nature.

Notably, the Compact Muon Solenoid (CMS) stands as one of the LHC's major detectors. As a general purpose detector, it consists of several smaller sub-detectors. These include [2] the Silicon Tracker, Electromagnetic and Hadronic Calorimeters, and the Muon System, each specializing in the measurement of particular particle properties[1]. This allows for the measurement of many different properties of the particles produced during collisions. However, while impressive in volume, this data is not immediately conductive to meaningful physics analysis. Like any good experiment, the data needs to go through a validation process in order to ensure it is not compromised in any way and to verify that there are major problems with the CMS detector or any of its subsystems.

The CMS collaboration undertakes a rigorous data certification process which ensures data quality and which detects potential issues with the detector or any of its subsystems [3]. The "boots on the ground" in this monumental effort of data quality monitoring (DQM) are shifters: members of the collaboration which check a great number of monitoring elements (MEs) (i.e. detector-level data which is used to monitor the state and performance of the components of the detector) to certify runs, which are long periods[2] of continuous data taking, as good if nothing is amiss and bad if there is a major issues with the data or the detector. However, with increasing data volumes [4], human driven DQM

[1] The physics aspect of the CMS detector will not be the focus of this work and thus details pertaining to it will not be expanded upon unless necessary.

[2] Long in this context could be from a few minutes to a couple of hours. The length of runs is a highly variable factor.

faces serious challenges. Tiring shift schedules can result is erroneous certifications and oversights, which has the potential of causing the inclusion of compromised data into crucial physics analyses. Moreover, the human driven approach naturally has a limited granularity it its evaluation. Currently, data is certified at the run level, which can often miss finer discrepancies which are only observable at the lumisection (LS) level, where a lumisection is a 23 second of data collection.

To address these challenges, there are ongoing efforts in the CMS collaboration to development of machine learning (ML) tools which seek to enhance and partially automate the DQM process [5, 6]. Such tools aim at anomaly detection, assisting shifters in detecting potential problems with the data with higher precision and efficiency. Our project pivots the question of how to automate the DQM process and instead delves into an evaluating the accuracy of human shifters using a ML approach. The main objective will be to ascertain the amount of anomalous lumisections which are certified as good due to the lower granularity evaluations shifters perform.

This project will utilize a dataset comprising 16 LS-level monitoring elements for a series of proton-proton collision runs conducted in 2017. Each monitoring element is represented by a 1D histogram with 102 bins, which we will cut down to 62 bins and normalize for computational efficiency and to facilitate model training convergence. This will result in an input and output of 16×62bins for each LS. To manage this data, we will deploy an ensemble of autoencoders, each trained on a specific histogram type, to specialize in the reconstruction of a particular monitoring element. Using the mean squared error (MSE) our metric for reconstruction error, we will establish reconstruction MSE thresholds for all the monitoring elements. This will allow us to ascertain if a lumisection is anomalous (MSE $>$ threshold) or not.

2 Methods

2.1 Datasets

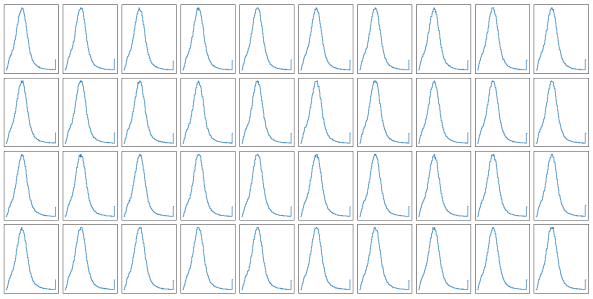
The datasets used in this project were sourced from the CMS's EOS internal storage system, specifically targeting data from the Tracker ML4DQM project, which includes runs from 2017 and 2018. For this proof-of-concept analysis, we focused on a subset of runs from 2018, encompassing runs numbered 297047 to 299329, totaling 135 available runs. The extraction of the data from the files found in EOS involved the use of a script provided by the ML4DQM-DC team, which isolates individual histogram types and stores them into separate CSV files. In order to focus the scope of this work, only monitoring elements specific to the Pixel Phase 1 system were utilized, encompassing a total of 16 different types of histograms, such as `chargeInner_PXLayer_1`, `charge_PXDisk_+1` and `num_clusters_ontrack_PXBarrel`. A simple looping script joined these individual files into a single dataframe containing LS-level histograms encompassing all the runs of interest.

The dataset was further segmented using a file sourced from the Certification Helper and a "golden JSON" file, which lists all run and LS combinations which were certified as good. This segmentation resulted in three datasets:

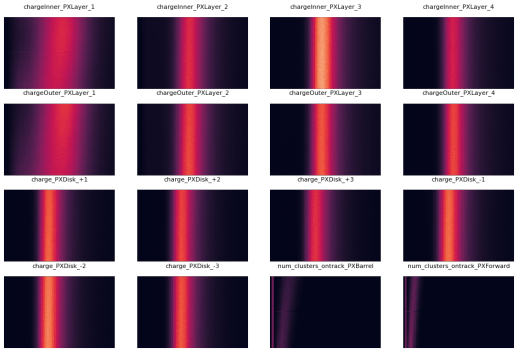
- `meRR_df`: Reference runs available in the dataset.
- `megood_df`: Non-reference runs with available reference in `meRR_df` and certified as good.
- `mebad_df`: Non-reference runs with available reference in `meRR_df` and certified as bad.

For preprocessing, we normalized all histograms to have unity area and cut out all bins above the 62nd. This latter decision was taken mainly due to the necessity to manage data size due to hardware limitations, and because the characteristic feature of all of the histograms was a peak to the left side. Additionally, after the 62nd bin, the histograms decline to zero quickly.

To visually asses the data, we plotted the first 40 LSs of the `chargeInner_PXLayer_1` monitoring element from the reference run 297178 (Figure 1a). These plots reveal a consistent shape in the histograms across LSs. This uniformity is made more evident through the heatmaps for each of the monitoring elements (Figure 1b). These plots show the consistent form of the monitoring elements throughout the run.



(a) `chargeInner_PXLayer_1` for the first 40 LSs



(b) Heat maps of all monitoring elements

Figure 1: Reference run monitoring element plots

2.2 Base Autoencoder

In the initial phase of our model development, we constructed a basic autoencoder with a simple architecture, as shown in Figure 2a. The training utilized the Adam optimizer, with the loss function being the MSE. This selection was based on the precedent set in the ML4DQM domain, where MSE is a standard metric commonly used for autoencoders.

```
# Encoder
self.encoder_layer1 = Dense(32, name='encoder_layer1')
self.leaky_relu1 = LeakyReLU(alpha=0.01, name='lrelu1')
self.encoder_layer2 = Dense(12, name='encoder_layer2')
self.leaky_relu2 = LeakyReLU(alpha=0.01, name='lrelu2')
self.encoder_layer3 = Dense(8, name='encoder_layer3')
self.leaky_relu3 = LeakyReLU(alpha=0.01, name='lrelu3')

# Decoder
self.decoder_layer1 = Dense(12, name='decoder_layer1')
self.leaky_relu4 = LeakyReLU(alpha=0.01, name='lrelu4')
self.decoder_layer2 = Dense(32, name='decoder_layer2')
self.leaky_relu5 = LeakyReLU(alpha=0.01, name='lrelu5')
self.decoder_output = Dense(62, activation='softmax', name='decoder_output')
```

(a) First version of the autoencoder

```
# Convolutional layers
self.leaky_relu1 = LeakyReLU(alpha=0.01, name='lrelu1')
self.conv1 = Conv1D(32, 5, padding='same', name='conv1')
self.leaky_relu2 = LeakyReLU(alpha=0.01, name='lrelu2')
self.conv2 = Conv1D(32, 5, padding='same', name='conv2')
self.leaky_relu3 = LeakyReLU(alpha=0.01, name='lrelu3')

# Encoder
self.encoder_layer1 = Dense(32, name='encoder_layer1')
self.leaky_relu4 = LeakyReLU(alpha=0.01, name='lrelu4')
self.encoder_layer2 = Dense(12, name='encoder_layer2')
self.leaky_relu5 = LeakyReLU(alpha=0.01, name='lrelu5')
self.encoder_layer3 = Dense(8, name='encoder_layer3')
self.leaky_relu6 = LeakyReLU(alpha=0.01, name='lrelu7')

# Decoder
self.decoder_layer1 = Dense(12, name='decoder_layer1')
self.leaky_relu7 = LeakyReLU(alpha=0.01, name='lrelu7')
self.decoder_layer2 = Dense(32, name='decoder_layer2')
self.leaky_relu8 = LeakyReLU(alpha=0.01, name='lrelu8')

# Convolutional layers
self.reshape = Reshape((32, 1))
self.conv3 = Conv1D(32, 5, padding='same', name='conv3')
self.flatten = Flatten()

# Output
self.decoder_output = Dense(62, activation='softmax', name='decoder_output')
```

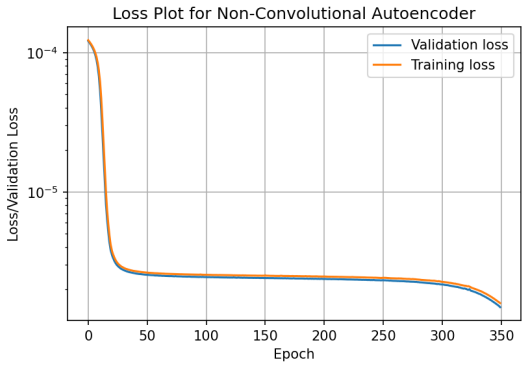
(b) Second version of the autoencoder

Figure 2: Structure of the autoencoders developed

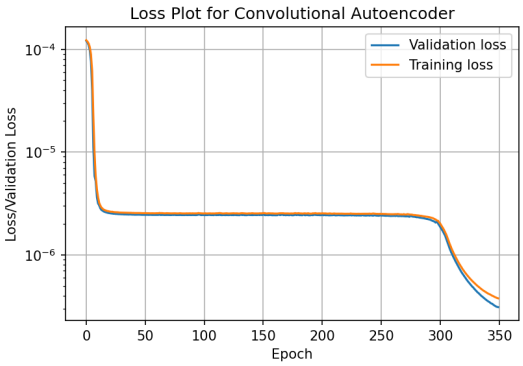
Our first model was trained on `chargeInner_PXLayer_1` histograms from the reference run 297178. During our training the data was split so that 20% would go towards validation. This meant that 1102 LSs went towards training and 276 towards validation from the total 1378 LSs in this reference run. The training was done over 350 epochs with a bath size of 128, based on numerous tests and fine tuning which indicated that the training and validation losses would begin to diverge at around this number of epochs, as seen in Figure 3a. The training took a total of 31.5 s.

Once the model completed training, the model's capability for reconstruction were tested using a good run, namely run 297050. The comparison of the reconstructed histograms against the original test histograms and the average of the training histogram (Figure 4a) indicated that the model had issues reconstructing displaced histograms.

To address this problem, we introduced convolutional layers into the autoencoder design, as seen in Figure 2b. This convolutional autoencoder was trained with the same hyperparameters. Despite the increased complexity, the training time was a reasonable 68 s. Once trained, its performance was also evaluated using the same approach with the pilot model. As can be seen in Figure 4b, there was a slight improvement in terms of the model's handling of translated histograms, although said improvement was not large. Nonetheless, this improvement justified the selection of this convolutional model as the base model for the autoencoder ensemble.

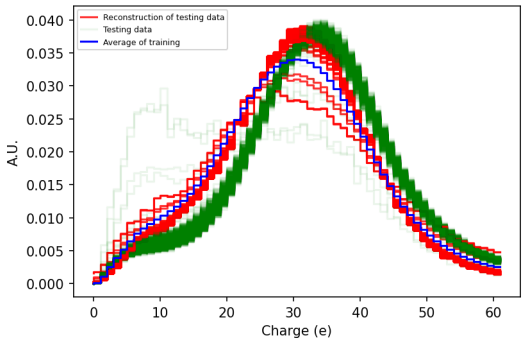


(a) Loss plot for pilot autoencoder

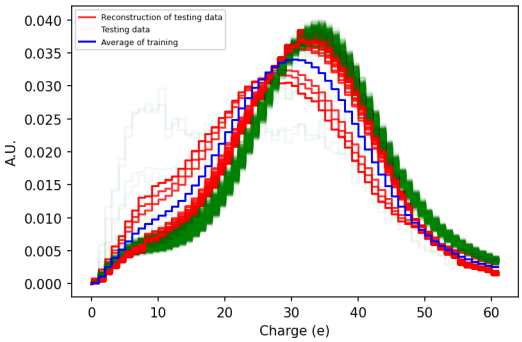


(b) Loss plot for convolutional autoencoder

Figure 3: Loss plots for both models developed



(a) Reconstructions of testing data by pilot autoencoder



(b) Reconstruction of testing data by convolutional autoencoder

Figure 4: Loss plots for both models developed

2.3 Autoencoder Ensemble

Building on the preparatory work described above, we developed an ensemble of autoencoders, each specialized to a specific monitoring element. The models were trained on the reference run 297178, with each autoencoder only seeing data of a particular type of histogram. This training followed the same methodology as the one applied for the individual model. The training took a total of 18 m and 54 s, although it seemed to have benefited from some degree of automatic parallelization seeing as the total amount of CPU time was 29m and 39s. Once trained, the ensemble was used on the set of bad runs and, for each LS for every run, we computed the average MSE across all monitoring elements. This resulted in a set of thresholds which were to be used later on to evaluate the anomalousness of LSs belonging to good runs.

We tested the ensemble with the `megood_df` dataset and the MSEs of each LS were compared against the previously obtained thresholds. The accuracy for a given run i and monitoring element (ME) was then computed using the Equation 1.

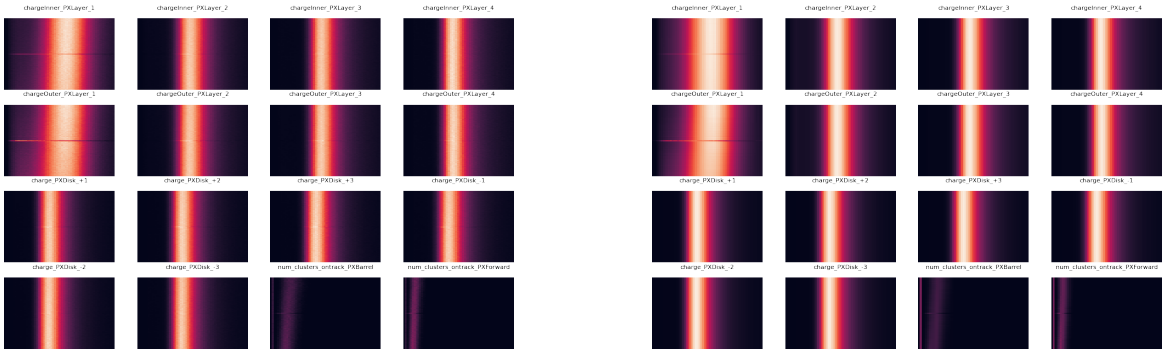
$$\text{Accuracy}_i(\text{ME}) = \frac{\text{Count of LSs with MSE } < \text{ Threshold in run } i}{\text{Total Count of LSs certified as good in run } i} \tag{1}$$

This metric was then used to gain insights into the performance of human shifters in the DC process. One way in which we did this was by taking the average of these accuracies along the monitoring elements of each histogram. This gave us a value which could tentatively be interpreted as a measure of the "goodness" of the a run, as measured by the autoencoder (Equation 2).

$$\text{Goodness}(\text{run } i) = \langle \text{Accuracy}_i \rangle_{\text{ME}} \tag{2}$$

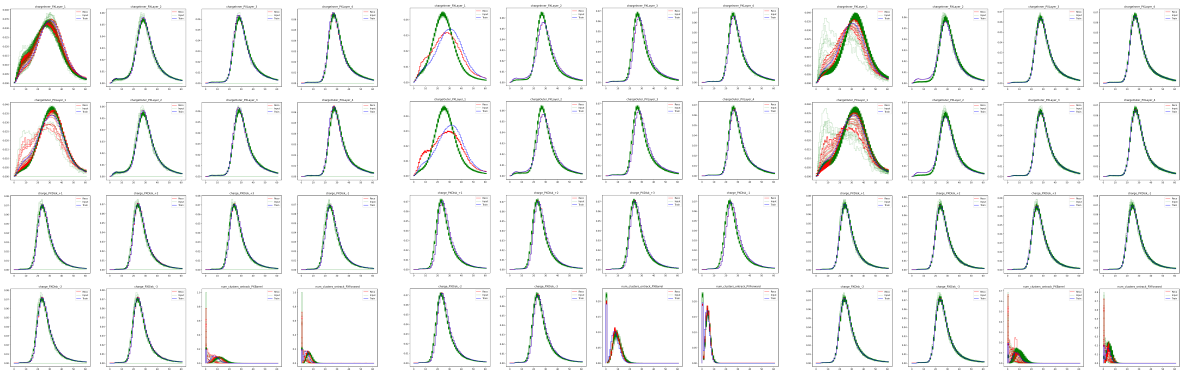
3 Results & Discussion

When testing the ensemble on the good runs dataset, the model's ability to reconstruct histograms was observed to generally be good, which is illustrated in Figure 6a. The model revealed a tendency to smooth out the histograms, indicating that the autoencoders were not overfitted, but that they capture the key features of their respective monitoring element histograms. A more detailed comparison of the histogram shapes against their reconstructions (Figure 6a) demonstrated a high fidelity in the reconstructions, with exception of monitoring elements relating to the Pixel Layer 1 and the cluster number histograms which seem to be the most affected due to changing data taking conditions.



(a) Original (b) Reconstruction

Figure 5: Monitoring elements for run 297050 and their respective reconstruction done by the autoencoder ensemble



(a) Run 297050 (b) Run 299329 (c) Run 297101

Figure 6: Monitoring elements, their reconstructions and the average of the training data

The certification accuracy for each monitoring element was computed using Equation 1. The accuracy scores[3] revealed some interesting results. Firstly, the accuracy data showed a significant portion of monitoring elements obtaining 100% accuracy. However, certain runs, such as 299329, display 0% accuracy in some monitoring elements. A more careful examination of this run (Figure 6b) reveal that the low accuracy is attributable to a significant offset in the histograms due to changing data taking conditions, a limitation of the model and not an indicator of poor data quality. This discrepancy with the training data highlights a key challenge in DQM with ML models: as the data taking conditions inside the detector change and begin to significantly differ from the training reference, simple autoencoders such as these struggle to perform.

To further investigate this issue, we analyzed the goodness score (Figure 7), revealing a trend where runs temporally closer to the reference run used for training had significantly higher scores than runs father away. This serves as evidence that the dwindling ensemble performance in the latter runs

[3]The table of all of the accuracy values can be seen in this project's GitHub page.

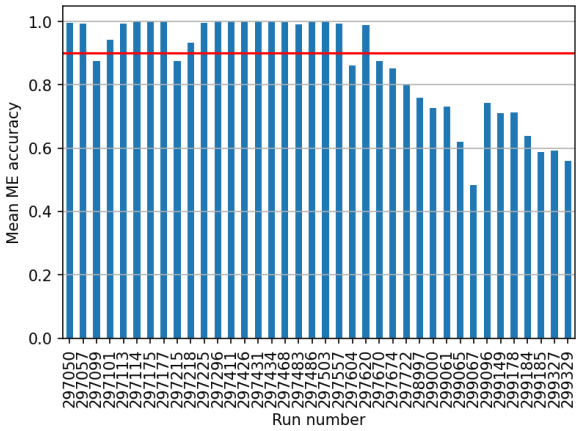


Figure 7: Goodness score for runs with good certification

was due to data-taking conditions and not to bad quality data. These close runs had notably higher goodness, with most exceeding 90%. However, specific runs, namely 297099, 297101, 297215, 297218 and 297604, had notably accuracies in some monitoring elements. Further investigation revealed that for most of these runs, the reduced accuracy was not indicative of anomalous behavior. Instead, they were largely due to translations or minor variations in the shapes of the histograms, factors which are not necessarily indicators of bad data quality. The outlier, run 297101, has distinctly anomalous histograms in around 20 of its LSs, as illustrated in Figure 6c. Determining the impact these LSs would have on overall data quality would require consulting with experts.

4 Conclusion

This report has detailed an analysis of human shifter accuracy in the Offline Tracker DC process for the CMS experiment at CERN. Additionally, through the development and application the a convolutional autoencoder ensemble used to do such an analysis, we investigated the performance of this model in the reconstruction of monitoring element histograms at the LS level for 2017 proton-proton collision runs. Our findings highlight a vital point about the limitations of reference runs when used for training ML models in DQM: due to the continuously changing data taking conditions, reference runs had a limited window of applicability. Beyond this window, models will tend to struggle to properly reconstruct monitoring elements despite there being no problem with the data quality. Changing conditions will tend to shift or warp histograms in ways that the model will not be able to properly process. These problems are exemplary of the difficulties with the application of autoencoders for DQM and demonstrate the need for continuous model training and adaptation.

Regarding human shifter accuracy, our evaluation did not lead to a significantly better understanding in this domain. The primary reason is that, due in part to hardware limitations, the analysis was performed with a limited number of runs. Thus we cannot draw strong conclusions in this regard. What we can say is that, based on the presence of some potentially anomalous histograms, the lack of granularity did seem to lead to *potentially* bad LSs being overlooked and certified as good. So, while our study did not shed much light on human shifter accuracy, it did highlight the potential for overlooked anomalies at the LS level. This underscores the role ML can have in DQM, particularly in identifying subtle discrepancies which would otherwise be overlooked with the current approach.

5 Future Work

Given the limited number of runs use for this analysis, the natural next step would be to run a similar analysis, but with more capable hardware and an increased number of runs. In this way, a better understanding of per-LS human-driven DC accuracy can be reached. Such efforts would likely benefit of the use of different metrics and models such as convolutonal autoencoders of higher complexity, Non-negative matrix factorization models or variational autoencoders.

Additionally, considering the observations made relating to the limited applicability of reference runs, a deep dive into this domain could prove fruitful. Such efforts might provide valuable understanding of the applicability and limits of ML in DQM, and ways in which these limitations can be overcome, such as with data augmentation.

6 Acknowledgments

I extend my gratitude to Richa Sharma and Guillermo Fidalgo for their feedback and I am also thankful to the Tracker ML4DQM team for providing access to the data and data extraction tools essential for this work. Special thanks goes out to Gabriele Benelli the for the enlightening discussions on Data Quality Monitoring.

7 References

[1] CERN, "The Large Hadron Collider," Oct. 2023. [Online]. Available: https://home.cern/science/accelerators/large-hadron-collider

[2] The CMS Collaboration, "The CMS Experiment at the CERN LHC," vol. 3, no. 08, p. S08004, aug 2008. [Online]. Available: https://dx.doi.org/10.1088/1748-0221/3/08/S08004

[3] V. Azzolini, B. Broen van, D. Bugelskis, et. al., "The Data Quality Monitoring Software for the CMS experiment at the LHC: past, present and future," vol. 214, p. 02003, 2019.

[4] S. Sanchez Cruz, "The phase-2 upgrade of the cms inner tracker," Geneva, 2023. [Online]. Available: https://cds.cern.ch/record/2841550

[5] V. Wachirapusitan and collaboration, on behalf of CMS, "Machine Learning applications for Data Quality Monitoring and Data Certification within CMS," *Journal of Physics: Conference Series*, vol. 2438, no. 1, p. 012098, Feb. 2023.

[6] A. A. Pol, G. Cerminara, C. Germain, M. Pierini, and A. Seth, "Detector monitoring with artificial neural networks at the CMS experiment at the CERN Large Hadron Collider," no. arXiv:1808.00911, Jul. 2018, arXiv:1808.00911 [physics, stat]. [Online]. Available: http://arxiv.org/abs/1808.00911