Beam Background Detection in ALICE ITS Data Using Machine Learning

CERN Summer Student Report

Zsófia Jólesz*
Eötvös Loránd University Budapest

Supervisors: Matteo Concas and Fabio Catalano
CERN
(Dated: 27.09.24)

The Inner Tracking System (ITS) is a crucial detector within the ALICE (A Large Ion Collider Experiment) at CERN's LHC, which explores quark-gluon plasma properties through heavy-ion collisions. With the onset of Run 3, the ITS generates a massive influx of data, presenting new challenges in processing and analysis. One key issue is data pollution, caused by residuals from the beam hitting the detector, which can compromise data quality.

This project focuses on developing and testing a machine learning-based approach to detect beam-related background in the context of a Summer Student Project. By automating the detection of such anomalies, the project aims to establish a foundation for more advanced anomaly detection systems in the future. This first step will help evaluate the feasibility of identifying specific features with machine learning, ultimately leading to improved detection of malfunctions, data integrity issues, and unanticipated phenomena affecting the detector's performance.

Keywords: ALICE; Beam Background; Anomaly Detection; CNN

* Correspondence email address: jolesz.zsofia@wigner.hun-ren.hu

I. INTRODUCTION

The Inner Tracking System (ITS) is the innermost detector of the ALICE, consisting of seven layers and containing 12.5 billion pixels across its silicon-based sensors. These pixels are organized on chips, with nine chips placed on structures known as staves (see Figure 1).

With Run 3 and the ITS's latest upgrade, an issue of data pollution has emerged, due to residual particles from the ion beam hitting the detector. Specifically, when some of the ions in the beam lose an atomic number, their mass is slightly reduced. This small change causes their trajectory to deviate slightly when passing through the collimator magnets. As a result, they may strike components of the accelerator, generating secondary particles, such as muons, that create distinct, long, horizontal tracks in the detector.

This beam-induced background, if left unfiltered, compromises the quality of the data collected by the ITS. Accurate and timely detection of such background is essential not only for maintaining data integrity but also for preventing potential malfunctions of the detector. Moreover, implementing robust anomaly detection techniques may help identify unexpected signals, potentially pointing to new phenomena not directly related to the detector itself.

The primary goal of this project is to develop a machine learning approach, specifically a convolutional neural network (CNN) to detect the presence of beam background in ITS data. The successful usage of this model represents a first step toward a more comprehensive, automated anomaly detection system for the ITS. This would pave the way for various benefits, including enhanced data integrity, prevention of detector failures, and the discovery of unusual signals that are not necessarily correlated to the operation of the detector.

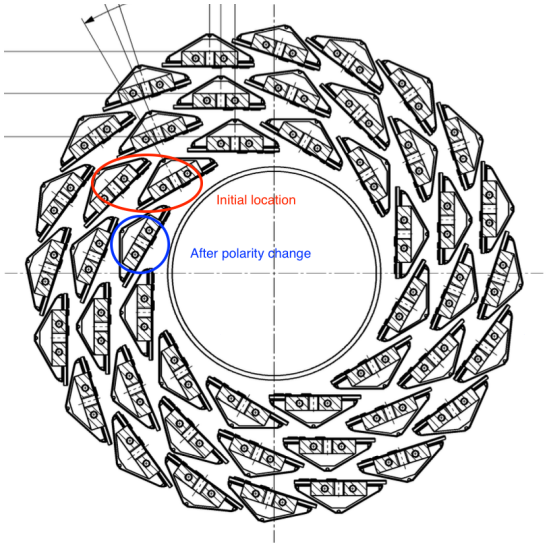


Figure 1: The layout of the ITS.

II. DATA PREPARATION

Data preparation is a critical initial step in developing a reliable machine learning model, as the quality and structure of the dataset directly impact the model's performance and its ability to classify data correctly. For this project, a clean and well-defined dataset was essential to accurately train the network to identify beam-related background. In particular, a dataset was required in which beam background signals could be easily identified and distinguished from other types of data, ensuring clear and interpretable patterns for training.

For this purpose, we selected Run 543612, a test run conducted on 29/09/23 during a week-long session of Pb-Pb collisions. The primary goal of this run was to benchmark beam background in the ALICE Inner Tracking System (ITS). Crucially, this dataset contains only signals from beam circulation and excludes collision events, providing the "clean" and controlled environment necessary for training and testing machine learning algorithms. The absence of collision-related hits makes it ideal for isolating and analyzing the characteristic signatures of beam background.

To better understand the data and identify features useful for training, I developed a ROOT [1] macro that reads, expands, and plots clusters for selected staves and layers, integrating over all timeframes and readout frames (see Figure 2). As shown in the figure, beam background manifests as distinctive long, horizontal clusters, which are

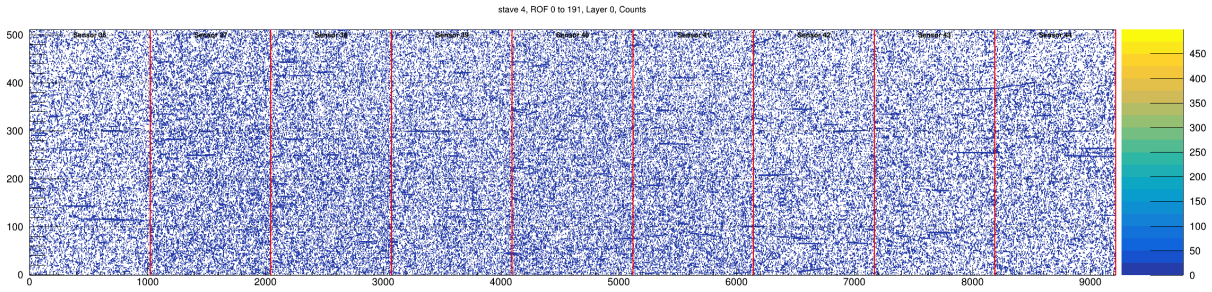


Figure 2: Clusters in stave 4 and layer 0, integrated for 192 readout frames.

easily identifiable when the data is integrated across timeframes and readout frames. However, it is important to note that the challenge of this project lies in identifying which specific timeframes and readout frames contain beam background, as these clusters are not always present simultaneously across different timeframes. Consequently, the data from individual timeframes and readout frames is much sparser compared to the integrated view, making the task of anomaly detection more complex (see Figure 3 for sparse data representation).

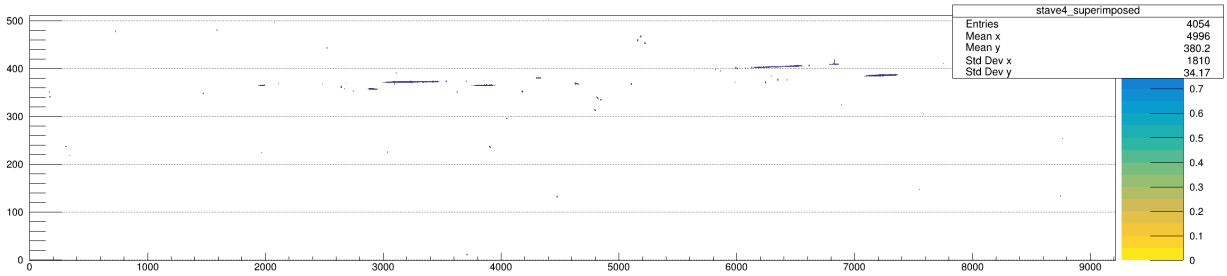


Figure 3: Clusters in stave 4 and layer 0, for a single timeframe and readout frame.

The next step involved converting the raw data into a format suitable for machine learning. The data processing workflow is illustrated in Figure 4. First, the ROOT macro was employed to extract relevant cluster information, which was stored in a TTree object. The key features extracted included the position and size of each cluster (row and column coordinates, and their lengths), sensor IDs, patterns, and metadata such as layer, stave, timeframe, and readout frame numbers.

Figure 4: Workflow of the data processing.

To determine whether individual timeframes and readout frames contained beam background, I analyzed the strongest feature of the clusters: their row length. I modified the ROOT macro to iterate through all timeframes and readout frames, identifying and recording those that contained beam background based on cluster row length. This information was compiled into a list for further processing. Subsequently, I developed a Python script to process the list, saving the cluster data in the form of numpy arrays. Each array had dimensions of (512, 9216), corresponding to the pixel layout of the detector. Pixels hit by the beam background were marked with a value of 1, while all other pixels were set to 0. To ensure a sufficient volume of training data, I repeated this process for all 33 compressed timeframes from the run, generating a robust dataset for model development. Finally, the data was split into training, validation, and test sets to evaluate model performance and avoid overfitting. The training set consisted of 612 examples, the validation set contained 288 examples, and the test set was comprised of 178 examples. This careful preparation of the dataset was crucial to ensure that the model could perform well and accurately detect beam background across different conditions.

III. DEVELOPMENT OF THE CNN

After preparing the dataset, the next step involved designing and implementing a convolutional neural network (CNN) for the task of binary classification of detecting beam background. For this, I used PyTorch, a widely adopted deep learning framework known for its flexibility and efficiency in model development.

The CNN architecture consisted of three convolutional layers followed by three fully connected layers, as detailed in Table I. The network starts with a 2D convolutional layer that processes the input with six filters, followed by progressively smaller filter sizes in subsequent layers, down to eight filters. This progressive reduction helps in efficiently learning relevant spatial features while reducing the computational complexity.

Layer (type)	**Output Shape**	**Param #**
Conv2d-1	[-1, 6, 13, 285]	98,310
Conv2d-2	[-1, 10, 6, 142]	550
Conv2d-3	[-1, 8, 2, 70]	328
Linear-4	[-1, 32]	8,992
Linear-5	[-1, 16]	528
Linear-6	[-1, 2]	34
Total Params		**108,742**
Trainable Params		**108,742**
Non-trainable Params		**0**

Table I: CNN architecture and parameters.

The total number of parameters in the model was 108,742, all of which were trainable. The model was used on a CUDA-enabled GPU for efficient computation.

To optimize the network's performance, I experimented with different activation functions, and decided that the Swish activation function provided the best results for this particular problem. Swish, which is defined as $x \cdot \text{sigmoid}(x)$, offers smooth non-linearity and has been shown to perform better than traditional activation functions like ReLU in certain machine learning tasks.

In addition, I also tested various hyperparameter configurations. Specifically, I varied the batch sizes (10, 30, 32, 50) and the number of epochs (20, 25, 50, 100) to assess their impact on model performance. Through systematic testing, I found that a batch size of 10 and training over 25 epochs, combined with a learning rate of 0.005, yielded the best results. Larger batch sizes or more extended training resulted in diminishing returns and even led to overfitting, as indicated by the increase in the validation loss after 25 epochs (see Figure 5).

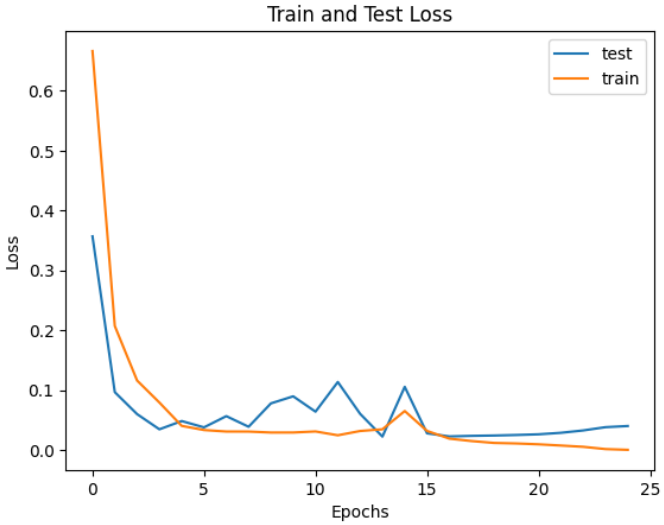


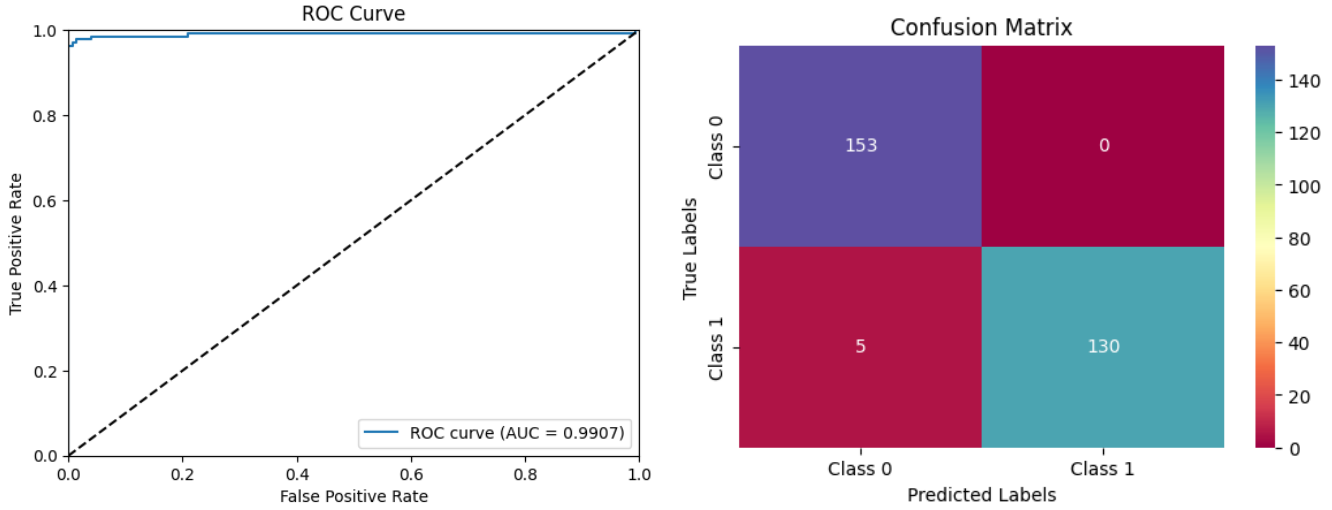
Figure 5: Train and test loss in the final configuration of the model.

The CNN was evaluated using both the training and test datasets. For each configuration, I made checks on both

the training loss and test loss to ensure that the model generalized well and did not overfit. As mentioned above, I observed that after 25 epochs, the training loss continued to decrease, while the test loss began to increase, signaling the overfitting of the model (Figure 5). Consequently, I selected the 25-epoch configuration as the optimal stopping point to prevent further overfitting.

To further evaluate the model's performance, I plotted the receiver operating characteristic (ROC) curve for each configuration. The best-performing configuration (batch size = 10, epochs = 25, learning rate = 0.005) showed excellent performance with an area under the ROC curve (AUC) close to 1.0 and an overall accuracy of 99% (see Figure 6a). Other configurations also performed well, with accuracy ranging from 96-99%, but they did not outperform the selected configuration in terms of minimizing both the false positive and false negative rates.

The confusion matrix for the best configuration showed five misclassified cases during testing, all of which were false negatives (Figure 6b). This indicates that while the model performed exceptionally well in detecting the beam background, a few instances of background were missed.



(a) (b)

Figure 6: Left panel: The ROC curve of the model. Right panel: the confusion matrix during testing.

IV. INTERPRETATION OF THE MODEL

To gain a deeper understanding of the CNN's performance and evaluate its suitability for the specific task of beam background detection, I investigated the classification outcomes using unseen data, which was neither part of the training set nor the test set. This approach allows for an unbiased assessment of the model's generalization ability.

The first step in this analysis was to examine the distribution of cluster sizes in the unseen data. I plotted a histogram (see Figure 7) that depicts the number of images (numpy arrays) containing clusters with sizes greater than the previously established threshold of 100 pixels. By comparing the number of images with clusters larger than 100 pixels against the number of images labeled as containing background, I found that the vast majority of images with beam background were classified correctly. This indicated a strong correlation between large clusters and accurate background classification.

To further investigate the model's performance, I plotted the predicted probabilities of each data sample being classified as background or non-background, for both the test set and the unseen data (see Figure 8). As shown in the figure, only a small number of cases were misclassified, demonstrating the robustness of the model. This analysis was critical in determining whether the model was consistently identifying beam background or mistakenly labeling it as non-background, and vice versa.

While the quantitative metrics provided valuable insights, visualizing the misclassified images was essential for a more detailed understanding of the model's behaviour. I selected and plotted some of the false negative and false positive cases (see Figure 9). In the false negative examples, the misclassification could be related to sparse clusters or clusters located near the edges of the array. The latter cases might have been missed due to the absence of padding in the convolutional layers of the CNN.

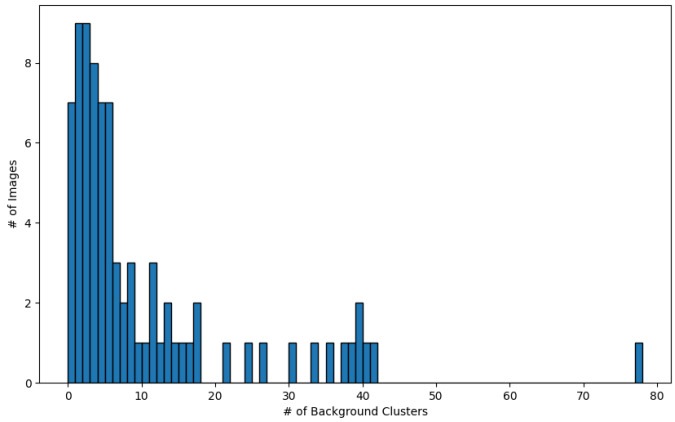


Figure 7: Number of images classified as beam background with cluster sizes greater than 100 pixels.

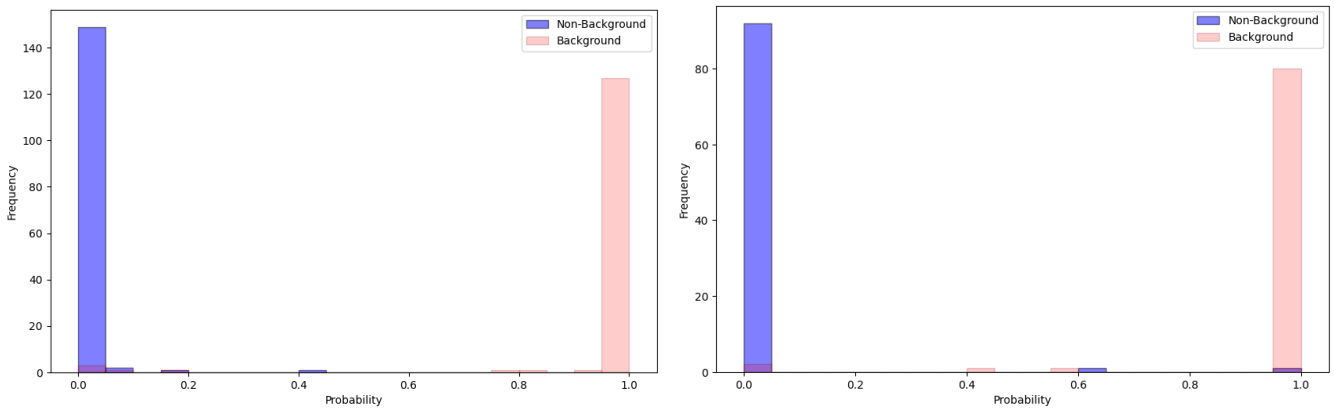


Figure 8: The probabilities of data samples being non-background or background for the test (left) and the unseen dataset (right).

On the other hand, the false positive cases highlighted a promising aspect of the model's capabilities. In these cases, the model successfully recognized beam background even when the clusters were slightly below the 100-pixel threshold. This suggests that the CNN has learned to capture relevant features of beam background that do not satisfy the arbitrarily chosen threshold of row length, demonstrating flexibility in its classification process.

V. SUMMARY

This project focused on developing and testing a convolutional neural network (CNN) for the detection of beam background in ALICE ITS data. A key aspect of the work involved ensuring the use of high-quality, clean input data, which is crucial for the effective training and evaluation of machine learning models. By carefully preparing the dataset, including selecting test runs specifically designed for background benchmarking and processing the data into a usable format, the foundation was set for accurate model development.

The CNN was designed for binary classification, with the objective of distinguishing between background and non-background data. The network architecture was optimized through tests with different hyperparameters, activation functions, and configurations, leading to a best-performing model that achieved 96-99% accuracy during testing. The best configuration was found to be a batch size of 10, 25 epochs, and a learning rate of 0.005, which minimized overfitting and maintained high accuracy.

Despite the overall high performance, some misclassifications were observed. False negatives typically occurred in cases where the data was very sparse or clusters were located near the edges of the array, which could potentially be mitigated by incorporating padding into the CNN architecture, which would preserve important spatial information near the boundaries. False positives, on the other hand, revealed a promising capability of the model: it was able to recognize beam background even in cases where the cluster size was slightly below the 100-pixel threshold. This

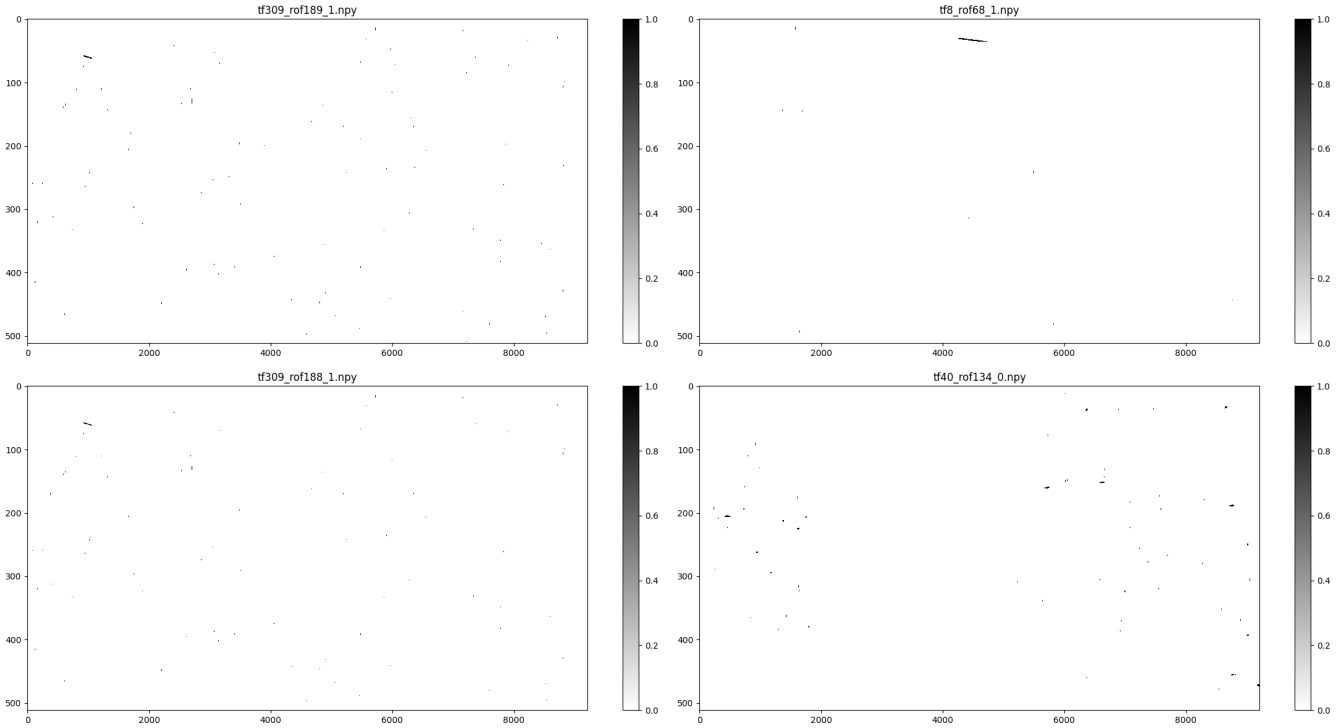


Figure 9: 4 examples of the misclassified cases, 3 false negatives and 1 false positive (lower right).

flexibility demonstrates the model's ability to generalize and detect background features that may not strictly satisfy the predefined size threshold.

Overall, the project successfully demonstrated the feasibility of using a machine learning-based approach for beam background detection, laying the groundwork for future improvements in automated anomaly detection systems within the ALICE experiment.

VI. ACKNOWLEDGEMENTS

I wish to express my great gratitude towards my supervisors, Matteo and Fabio for their exceptional guidance and support throughout this project and the vast amount of knowledge I gained here by them. I would also like to thank my friends: Christine, Moritz, Zobi, Alice, Fabi, Luisa, Katrin, Niklas, Lam and Laura for their insights and the wonderful time I spent with them.

[1] R. Brun and F. Rademakers, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **389**, 81 (1997).