NA61/SHINE online noise filtering using machine learning methods

Anna Kawęcka[1,2], Wojciech Bryliński[1], Manjunath Omana Kuttan[3,4], Olena Linnyk[3,5,6], Janik Pawlowski[3,7], Katarzyna Schmidt[8], Marcin Słodkowski[1], Oskar Wyszyński[9], Jakub Zieliński[1]

[1] Warsaw University of Technology, Warsaw, Poland
[2] Chalmers University of Technology, Gothenburg, Sweden
[3] Frankfurt Institute for Advanced Studies, Frankfurt am Main, Germany
[4] Johann Wolfgang Goethe University, Frankfurt am Main, Germany
[5] Justus-Liebig University Giessen, Giessen Germany
[6] milch&zucker AG, Giessen, Germany
[7] Philipps-University Marburg, Marburg, Germany
[8] University of Silesia, Katowice, Poland
[9] Jan Kochanowski University in Kielce, Poland

E-mail: anna.kawecka@cern.ch

Abstract. The NA61/SHINE is a high-energy physics experiment operating at the SPS accelerator at CERN. The physics program of the experiment was recently extended, requiring a significant upgrade of the detector setup. The main goal of the upgrade is to increase the event flow rate from 80Hz to 1kHz by exchanging the read-out electronics of the NA61/SHINE main tracking detectors (Time-Projection-Chambers - TPCs). As the amount of collected data will increase significantly, a tool for online noise filtering is needed. The standard method is based on the reconstruction of tracks and removal of clusters which do not belong to any particle trajectory. However, this method takes a substantial amount of time and resources. A novel approach based on machine learning methods is presented in this proceedings.

1. Motivation and data preparation

SPS Heavy Ion and Neutrino Experiment (SHINE) [1] is a fixed-target experiment operating at CERN Super Proton Synchrotron (SPS). The NA61/SHINE detector is a multi-purpose spectrometer optimized to study hadron production in various types of collisions. The main subdetectors of the whole setup are the Time Projection Chambers (TPCs). Two Vertex-TPCs (VTPCs), located in the magnetic field, together with two large volume Main-TPCs (MTPCs), are the main tracking devices and are able to register a large number of particle tracks (up to 1500 in central Pb+Pb collisions). More information about the detector can be found in [1]. The primary physics motivation of the NA61/SHINE experiment is to study the phase transition properties between hadronic matter and quark-gluon plasma. Recently, the physics program was extended by the open charm measurements, requiring the major upgrade of detection setup. The main goal of the upgrade is the tenfold increase of the data taking frequency, from 80Hz up to 1kHz. This can be achieved by replacing the read-out electronics of the TPCs. Faster read-out will result in higher data-rates and an enormous amount of data. It will no longer be possible to store all the registered data thus a fast and efficient tool for online noise filtering is needed.

The first step of TPC data reconstruction is clusterization. It is about connecting the signals left by one particle on the neighbouring TPC read-out pads into one cluster. Then, clusters (single TPC points) produced by one particle are reconstructed as TPC track. The example reconstructed collision is presented in Fig. 1. The TPC clusters reconstructed from noise contribute significantly to the total amount of data – from 50% (for large systems, like Pb+Pb), up to 70% (for small systems, like p+p) of total data size. Thus, by removing the noisy TPC clusters, one can reduce the size of the data. The standard method involves reconstructing the tracks and removing clusters that do not belong to any track. This method, however, is very time and resource consuming. This paper discusses a new approach, based on machine learning methods. Algorithms will learn to classify the TPC clusters and remove the noisy ones from the data.

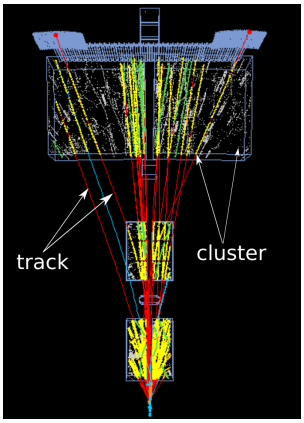

Figure 1. The visualisation of the reconstructed event.

Preparation of the training data sets
The first step of data preparation is the reconstruction of local tracks in order to label the TPC clusters into two groups: good clusters (belonging to track) and noisy clusters. The labeled cluster data can be used to train and test classification algorithms. Two different approaches were tested:

- The first approach (the approach I) uses the reconstructed properties of the clusters (a total charge, positions, the maximal signal in the cluster - max ADC, etc.) as the input data.
- The second approach (the approach II) uses the clusters raw data, i.e. for each cluster, after the pad ID and the timestamp of max charge deposit are found, the signals from this central pad at the time of the maximum deposit as well as from ± 5 neighboring pads and $\pm$ 9 timestamps are included. This results in 2D maps of signals from 11 consecutive pads at 19 timestamps.

2. Machine learning algorithms for the classification

The choice of the machine learning algorithm for the classification of signal vs. noise for TPC clusters depends on the type of the input data: reconstructed properties of the clusters in the approach I or 2D-Maps (raw cluster data) in the approach II. We propose and compare four machine learning algorithms: two for the approach I and two for the approach II.

Approach I. Using the reconstructed properties of the clusters (a total charge, positions, max ADC, etc.) two methods were tested: the decision tree (later referred to as DT) and dense neural network (later referred to as DNN). The decision tree was implemented using the `DecisionTreeCassifier` method provided by scikit-learn [2] library and the dense neural network was built with the `keras` library [3]. The hyperparameters of the decision tree (criterion: gini, max_depth: None, min_samples_leaf: 50, min_samples_split: 200) were found by the `HalvingGridSearchCV` method provided by scikit-learn. The `BayesianOptimization` method provided by `keras-tuner` [4] was used to find the architecture of the neural network, which is presented in Tab. 1

Approach II. The second approach is based on the raw cluster data, which are represented as a 2D pixel array, as described in Sec. 1. As such input resembles an image, it is reasonable to use convolutional neural networks (CNN), which were originally developed for the classification of images [5]. Two different architectures of CNNs have been implemented:

- traditional convolutional neural network architecture (later referred to as CNN) including two 2D convolutional layers, both consisting of 8 filters of size 3×3 each, and a fully connected deep neural network on top of them (details in Tab. 1). Moreover, after each convolutional layer, average pooling was applied, and after each dense layer (apart from the last), a dropout of 0.3 was applied.
- novel architecture, custom developed for the task at hand (later referred to as SplitCNN). This model [6, 7] was inspired by the committee networks [8] and by the channel-split convolution [9]. Using physics principles allowed us to split the convolution network into two parallel threads (details in Tab. 1). The SplitCNN implementation achieves comparably high noise reduction and low signal loss as the residual neural network (ResNet [10]) of three blocks of (conv 32@3x3, conv 64@3x3), but with 20 times fewer trainable parameters and 10 times faster training and prediction times.

Table 1. Chosen neural network architectures.
If not otherwise specified, ReLU activation is used.

Network Name	Architecture description
DNN	Dense 256 - Dense 128 - Dense 1(Sigmoid)
CNN	Conv 8@3x3 - AvPool - Conv 8@3x3 - AvPool - Dense 64 - DropOut(0.3) - Dense 32 - Dense 16 - Dense 1(Sigmoid)
SplitCNN	Split: - [Conv 5@3x1 + Conv 5@1x3] - [Conv 5@3x1 + Conv 5@1x3] - :Concatenate - Flatten - Dense 2(Softmax)

All the models were trained twice, using two balanced data sets of 20 000 samples each from the two reactions: Ar+Sc at 30 GeV/c and Pb+Pb at 150 GeV/c. In these reactions, the multiplicities of produced clusters are the order of 10^4 and 10^5 per event, respectively. The testing of each trained model was performed on three various data sets for the reactions: Ar+Sc at 30 GeV/c, Pb+Pb at 150 GeV/c and Be+Be at 30 GeV/c. The lowest multiplicity characterizes the latter. All three reactions were collected during different data-taking campaigns in 2015, 2018, and 2013, respectively. Each test data set consisted of about 80 000 samples. The purpose of testing on data from various reactions was to check whether models can generalize.

3. Results

The results for all used models, when trained and tested on the same reaction, are presented in Tab. 2. Several conclusions can be drawn here. First of all, the results are very similar regardless the method within the same reaction. There are better results for Ar+Sc reaction, which is characterized by lower multiplicity. The training and testing on different reactions resulted, in general, in lower accuracy of about 3% (not shown in the table). For the Be+Be reaction, the accuracy varied from 90% for SplitCNN method, through 92% for CNN up to 96% for DNN and DT, regardless the training set. The results suggest to perform training for each reaction separately. If, however, adjusting (retraining) the model for some reasons would not be possible, a model trained earlier on a reaction with high multiplicity can be used with good enough accuracy to reduce noise in reactions of significantly lower multiplicity.

Table 2. The table shows the quality of predictions of the four machine learning models. Each model was trained on a subset of the collected data and tested on the remaining data for the same collision system. Top row - for Ar+Sc at 30 GeV/c; bottom row - Pb+Pb at 150 GeV/c.

Dataset	Method	Accuracy[%]	Recall[%]	Precision[%]	f1 score[%]
Ar+Sc	DT	92	87	85	86
	DNN	92	87	84	86
	CNN	90	92	87	90
	SplitCNN	89	95	89	92
Pb+Pb	DT	84	79	79	79
	DNN	83	81	77	79
	CNN	81	79	79	79
	SplitCNN	82	87	80	84

An important aspect related to noise reduction is to have the rate of false negative samples (actual signal estimated as noise) as low as possible, which means the precision as high as possible. In the present study, the number of false negatives normalized to the number of actual positives reached up to a dozen percent for all methods. However, we can decrease the lost signal by modifying the threshold of the decision. The machine learning methods generally return the predictions as probabilities for positive and negative classes. Furthermore, assigning a sample to a definite class is made based on a set decision threshold (initial value 0.5). The decision threshold can be tuned to reject only clusters with a very high probability of noise and save the rest. It is, however, essential to check that the noise reduction remains sufficiently high. Fig. 2 presents the number of false negatives normalized to the actual number of positives as a function of the percentage of all remaining clusters (true positives plus false positives) in one event (Ar+Sc) for different decision thresholds, as predicted with the DT method. The labels on the curve represent the threshold of the decision for the positive class: if the probability is below it, clusters will be predicted as negative (noise). It is seen that using the default threshold, 14% of good clusters are lost, but almost all the noisy clusters are rejected (in Ar+Sc reaction, the actual ratio of signal to noise is 30% to 70%). Decreasing the threshold, the number of false negatives drops, but the number of passed noise increases. Therefore, the event size (all saved clusters) increases. With the threshold of 0.05, the event size reduction is still considerable (55%) while the percentage of lost good clusters is less than 2%. The threshold tuning can be used with all of the presented machine learning methods.

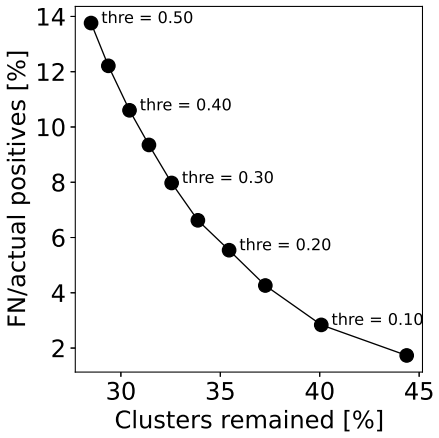

Figure 2. False-negative reduction and increase of the stored event size (true positive plus false positive: identified signal plus passed noise) as effect of varying the decision threshold.

Finally, we are also interested in the computing time performance for the used classification methods. One has to distinguish the training time from the prediction time. The prediction time t_{pred} is the relevant indicator for the applicability of the method as a part of the online noise reduction system (trigger) during the data acquisition (DAQ) in the experiment. In order to

answer precisely, which of the methods presented here is most suited to produce the fast decision, we still have to finish testing all the models on the actual NA61/SHINE DAQ machines. The results of these tests will be reported elsewhere. Current estimates indicate that all described methods have $t_{pred} < 0.02$ ms/cluster per corresponding CPU, i.e. below 2 s/event for Pb+Pb. The periodic retraining of the model parameters takes more computing resources but can be done very effectively on a GPU-based cluster.

4. Summary

The NA61/SHINE experiment needs an online noise filtering tool for future data taking, and machine learning is a very efficient and robust solution for TPC clusters classification. Two approaches were tested — using reconstructed and raw cluster information, and both gave very good results. The algorithms are currently being implemented in the data acquisition software as online tools, and the most efficient one will be used during future data-taking campaigns. Under the method's efficiency, we understand several factors: (a) the expected quality of prediction (noise reduction, event compression, signal loss), which can be evaluated statistically by accuracy, precision, and recall. (b) computation times needed for training and prediction. (c) generalizability, which directs how often the algorithm has to be retrained, whether the distilled decision governing patterns are general or varied from TPC to TPC and from collision system to collision system. We answered some of these questions in the present study, but further detailed investigations are in order.

Acknowledgments

We would like to thank the NA61/SHINE Collaboration. Studies were funded by IDUB-POB-FWEiTE-1 project granted by Warsaw University of Technology under the program Excellence Initiative: Research University (ID-UB), the National Science Centre Poland grant 2018/31/G/ST2/03910, the Norway Grants in the Polish-Norwegian Research Programme operated by the National Science Centre Poland (grant 2019/34/H/ST2/00585), the R&D grant by GSI and the ErUM Data project by BMBF. The resources of the Center for Computation and Computational Modeling at the Faculty of Natural Sciences of the Jan Kochanowski University in Kielce were used for this work.

References

[1] N. Abgrall *et al.* [NA61 Collaboration], JINST **9**, P06005 (2014) doi:10.1088/1748-0221/9/06/P06005
[2] F. Pedregosa *et al.* Journal of Machine Learning Research **12**, 2825–2830, (2011)
[3] F. Chollet, Deep learning with Python. Simon and Schuster, (2021).
[4] Keras Tuner, O'Malley, Tom and Bursztein, Elie and Long, James and Chollet, François and Jin, Haifeng and Invernizzi, Luca and others, https://github.com/keras-team/keras-tuner, (2019)
[5] Y. LeCun,, Y. Bengio, G. Hinton, Nature 521, 436–444 (2015) doi:10.1038/nature14539
[6] J. Pawlowski, "Reducing noisy clusters in the NA61/SHINE project with help of machine learning", Giessen - 2021, CERN-THESIS-2021-041, https://cds.cern.ch/record/2766095
[7] J. Pawlowski, O. Linnyk, "Causality motivated splitting of convolutional neural networks", submitted
[8] U. Meier, D. C. Ciresan, L. M. Gambardella and J. Schmidhuber, "Better digit recognition with a committee of simple neural nets," 2011 International Conference on Document Analysis and Recognition, 2011, pp. 1250-1254, doi: 10.1109/ICDAR.2011.252.
[9] T. Jin, S. Hong, "Split-CNN: Splitting Window-based Operations in Convolutional Neural Networks for Memory System Optimization", ASPLOS'19: 24th International Conference on Architectural Support for Programming Languages and Operating Systems, 2019, pp. 835–847, https://doi.org/10.1145/3297858.3304038
[10] C. Szegedy, S. Ioffe, V. Vanhoucke, A.A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning", Thirty-First AAAI Conference on Artificial Intelligence, 2017 - aaai.org