NA61/SHINE Online Noise Filtering Using Machine Learning Methods

Marcin Słodkowski[1]

on behalf of:

Wojciech Bryliński[1], Kordian Makulski[1], Katarzyna Schmidt[2], Oskar Wyszyński[3]

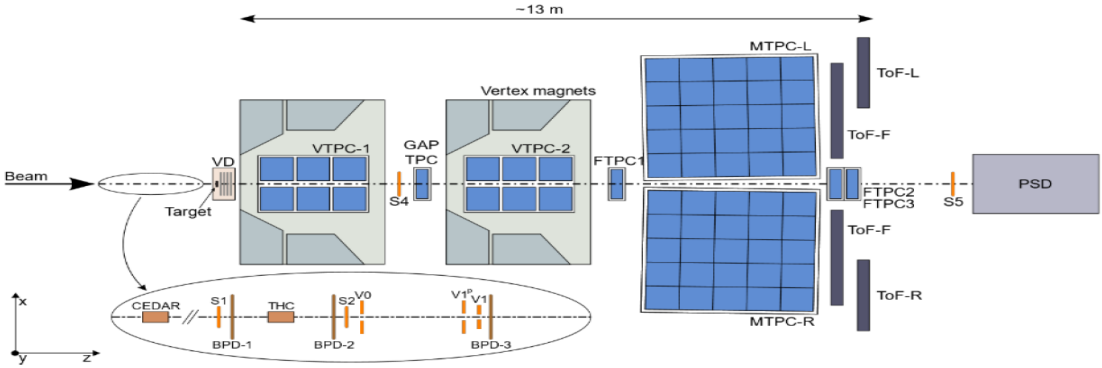
Warsaw University of Technology, Warsaw, Poland[1]
University of Silesia, Katowice, Poland[2]
Jan Kochanowski University in Kielce, Poland[3]

ML4Jets2023 6-10 November 2023.
DESY, Hamburg, Germany

Agenda

NA61/SHINE experiment

NA61/SHINE is a fixed-target experiment, which means that particles from the incoming beam hit the targets, and the producing particles are recorded by a detection system located behind the target.



The detection system of the NA61/SHINE experiment consists mainly of 4 large TPC chambers (2 Vertex-TPC chambers placed in a magnetic field and 2 Main-TPC chambers) and 4 smaller chambers (Gap-TPC and 3 Forward-TPC chambers).

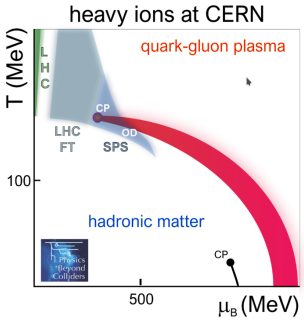
NA61/SHINE physics program

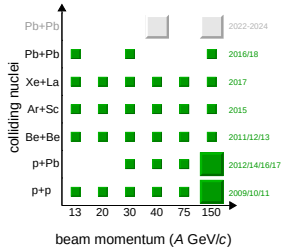
Strong interactions physics:

- Study of the properties of onset of deconfinement and onset of fireball.
- Properties of QCD matter (EoS).
- Exploration region of the critical point of strongly interacting matter.
- Direct measurement of heavy quarks (open charm) at SPS energies.

Cosmic ray and neutrino physics:

- Measurements of nuclear fragmentation cross section for cosmic ray physics.
- Measurement for neutrino programs at J-PARC and FERMILAB.





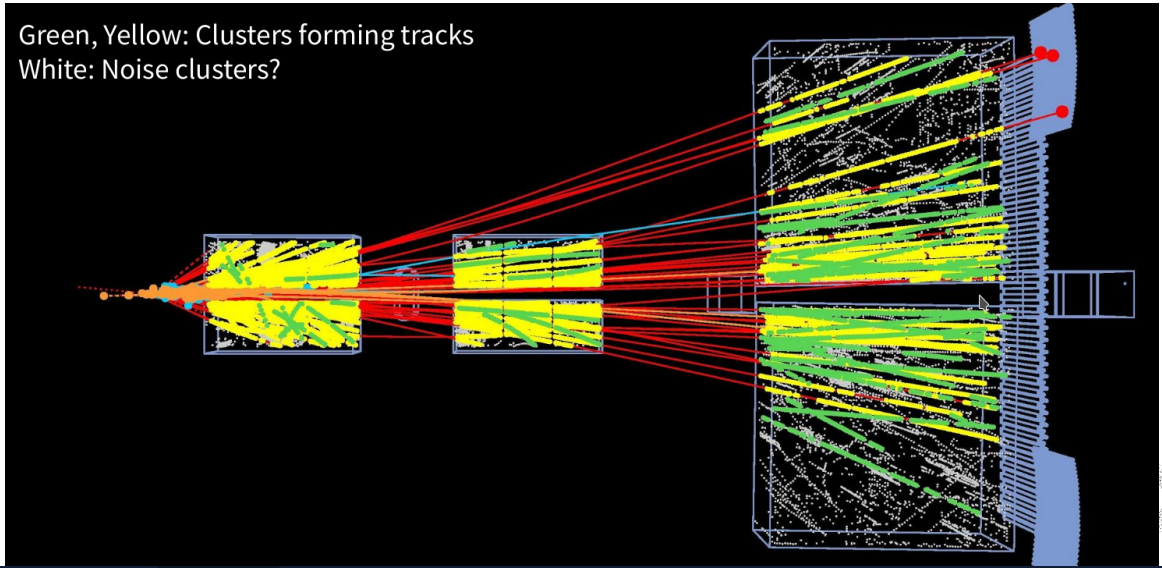
NA61/SHINE experiment update in 2022

- Changes in the detector configuration due to the need for a new physical program.
- Time-Projection-Chambers (TPCs) read-out electronics upgrade.
  - Smaller number of malfunctioning channels.
  - More homogenous pad-by-pad gains.
  - Lower noise level.
- New DAQ system.
  - Increasing the event flow rate from 80 Hz to 1.7 kHz.
  - Data taking with upgraded detector in 2022.
- New or upgraded sub-detectors.
  - Large Acceptance Vertex Detector (VD).
  - New silicon strip Beam Position Detectors (BPD).
  - New Time of flight detectors left and right (TOF-L, TOF-R).
  - New Projectile Spectator Detectors (PSD).
- November 2022: 58 mln of Pb+Pb collisions at 150A GeV/c was collected.

Need for noise filtering

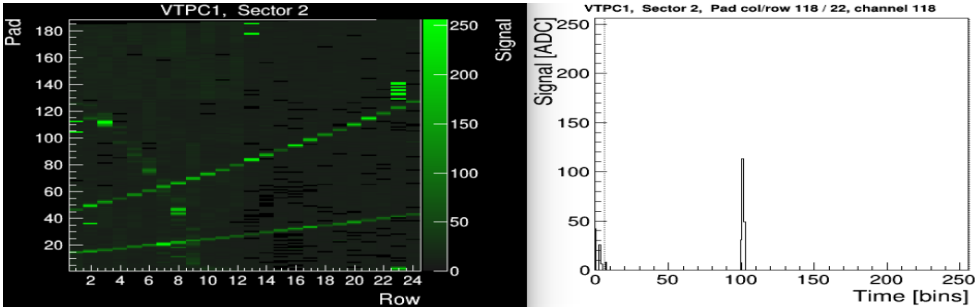
- An increased read-out speed leads to increased data transfer rates and a substantial volume of data.
- It is better not to store all the data due to the high cost and computer power required.
- A fast and effective online noise filtering solution is essential, especially for Pb+Pb interactions which are the largest data sets.
- Traditional methods based on particle track reconstruction are time-consuming and resource-intensive and can be replaced with machine learning methods.
- The goal is to separate noise clusters from the clusters that form the track (signal) before the reconstruction using Machine Learning techniques.
  - Support for online/offline reconstruction.
  - An alternative for online reconstruction.

An event in SHINE



Filtering methods

- Online reconstruction - reconstruct local tracks and remove the clusters which do not belong to any track.
- Dense NN (DNN) - use the neural network to find the noisy clusters - clusters properties as the input (charge, number of timeslices, number of padrows).
- Convolutional NN (CNN) - use the neural network to find the noisy clusters - raw data as the input.



Implementation

- Tools and technologies used.
  - nVidia GPU + drivers.
  - CUDA Toolkit installed.
  - cuDNN SDK (CUDA Deep Neural network library).
  - Tensorflow and Tesorflow C++.
  - Keras library.
- Description of the implementation.
  - The implementation process involves the development of SHINE modules within the ShineOffline framework, which leverage deep learning (DL) techniques for cluster filtering.
  - One of the key challenges faced was the integration of tools from the Tensorflow C++ into the ShineOffline based on ROOT C++ framework of the NA61 experiment.
  - To facilitate seamless integration and deployment within the TDAQ system, it incorporated TensorFlow C++.
  - For easier deployment within the TDAQ system, Docker images containing the ShineOffline software with TensorFlow C++ were prepared.
  - These Docker images can be used to create Docker containers for making ML predictions within the experiment's software framework.

ML software into the ShineOffline framework integration

- List of Shine-Offline modules towards input training data preparation:
  - NoiseClusterExcluderED module.
  - recEventLoop2 program.
  - Clusters program.
- The network training stage takes place in the Python environment with Tensorflow and Keras and Keras Tuner libraries using GPU:
  - tpc_ddn_helper.py script.
  - train_model.py script.
  - nn_tuner.py script.
- List of ML modules in Shine-Offline for data filtering and removing:
  - DLNoiseClusterFilterKS module.
  - DLNoiseClusterRemoverKS module.

How to use it

How to use built containers on TDAQ and ML with GPU machines:

1. Go to:
   ```
   cd $SOFTWARE_PATH shine-tf
   ```
   ```
   shine-dev-env-tf        ape-ubuntu-20-04     04ce0db8e953   6 weeks ago   3.68GB
   shine-dev-env           ape-ubuntu20-04      2656c816a711   6 weeks ago   2.75GB
   ```
2. Build the container:
   ```
   docker compose run shine-tf
   ```
3. In the container go to folder with ML modules:
   ```
   cd /shine/UMFiltering/DLNoiseClusterRemover2AJ
   ```
4. Source the shine environment and run the code:
   ```
   . shine-env.sh
   ```
   make run


```
Shine Offline Framework v1r22p0-development
based on NIM A 580 (2007) 1485.
home page: https://twiki.cern.ch/twiki/bin/view/MA61/SHINEOfflineHome
VCS revision: v1r21p0-432-g25741d0a7
bug reports to: https://its.cern.ch/jira/browse/SHINE
```

```
CONTAINER ID   IMAGE                                COMMAND       CREATED         STATUS                     PORTS     NAMES
c31eea3e6745   shine-dev-env-tf:ape-ubuntu-20-04    "/bin/bash"   38 hours ago    Exited (0) 38 hours ago              shinetf_ms-shine-tf-run-8a3de42403c5
```

Influence of noise reduction on reconstruction

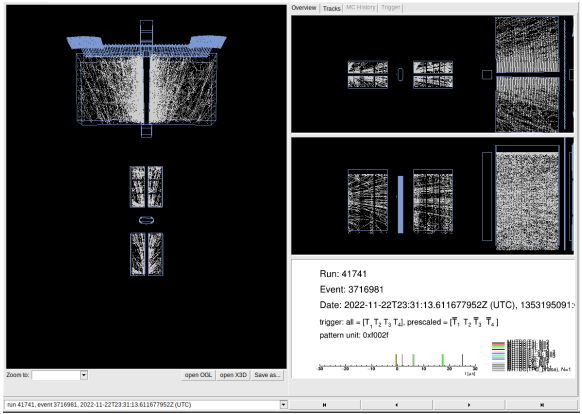


Figure: Clustered raw data.

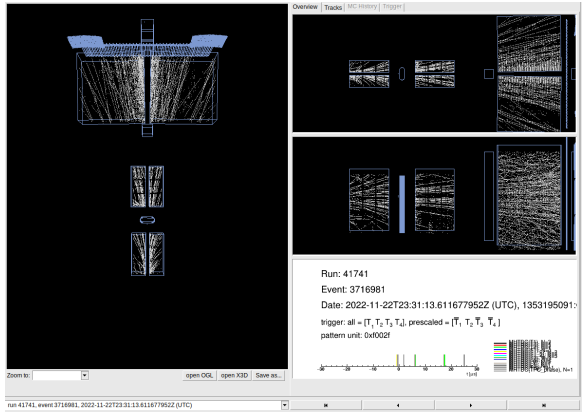
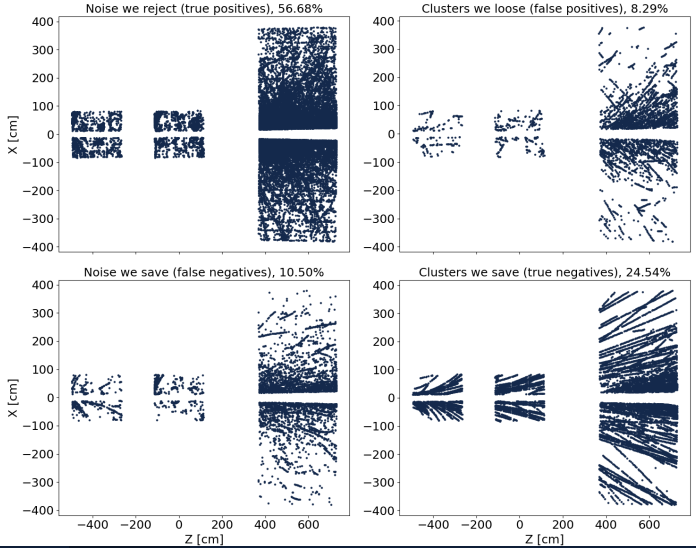


Figure: Reduced Data using CNN algorithms (threshold 0.5)

CNN confusion matrix (Pb+Pb collisions at 150 GeV/c)

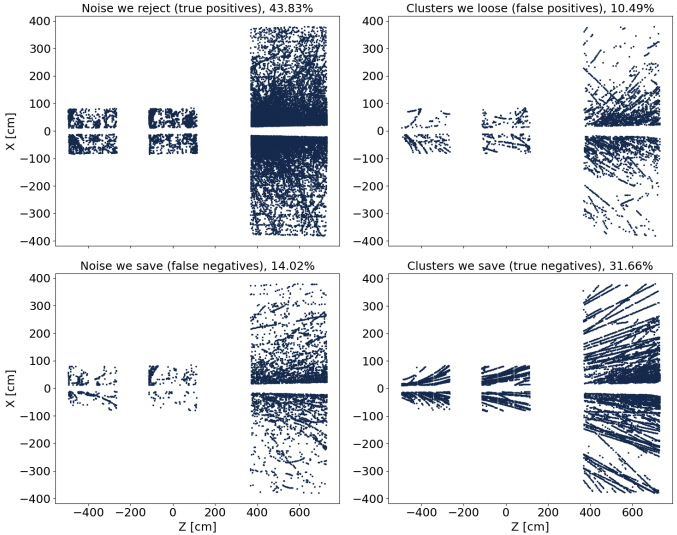


Noise to signal ratio is approximately from chunk file 041741x00894:

$$\frac{65\%}{35\%}$$

We reject: $56.7/65 = 87.2\%$ of the noise and keep: $24.5/35 = 70\%$ of the clusters (and $10.5/65 = 16\%$ of the noise)

DNN confusion matrix (Pb+Pb collisions at 150 GeV/c)



Noise to signal ratio for this event is:

$$\frac{57.8\%}{42.2\%}$$

We reject: $43.8/57.8 = 75.77\%$ of the noise and keep: $31.66/42.2 = 74.88\%$ of the clusters (and $14.2/57.8 = 24.5\%$ of the noise).

Metrics (Pb+Pb collisions at 150 GeV/c)

thresholds	accuracy	precision	recall	f1
0.5	0.79	0.80	0.78	0.79

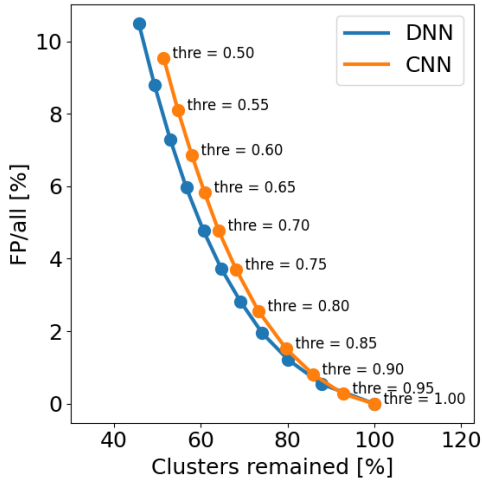
Table: CNN Metrics

	precision	recall	f1-score
0	0.69	0.75	0.72
1	0.81	0.76	0.78
accuracy			0.75
macro avg	0.75	0.75	0.75
weighted avg	0.76	0.75	0.76

Figure: DNN Metrics

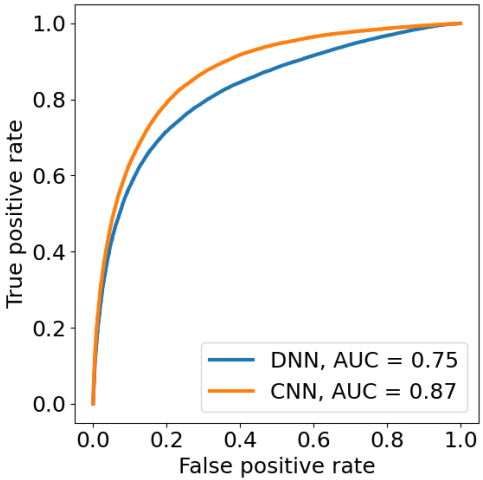
- **Precision** – the proportion of positive identifications that were actually correct
- **Recall** - proportion of actual positives identified correctly
- **F1** – harmonic mean of Precision and Recall

Parametric cluster classification (Pb+Pb collisions at 150 GeV/c)



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

Receiver Operating Characteristic (ROC) (Pb+Pb collisions at 150 GeV/c)



Receiver Operating Characteristic (ROC) analysis of CNN-filtered and DNN-filtered data.

Performance tests (Pb+Pb collisions at 150 GeV/c)

- GPU used: NVIDIA GeForce RTX 3080
- CPU used: Intel(R) Xeon(R) W-1250P CPU @ 4.10GHz

Trainig Set Prep Time	N of events	time [h]	ev/sec	sec/ev
DNN	836	0.05	4.75	0.21
CNN	200	0.48	0.12	8.42

Table: Training Set Preparation Time.

thr=0.5	N events	total time[h]	ev/sec	sec/ev	% of pred. module	ev/sec	sec/ev
DNN classification	836	3.44	0.07	14.81	0.25	0.27	3.66
DNN rejection					0.73	0.09	10.83
CNN classification	836	4,20	0.06	18.08	0.45	0.12	8.19
CNN rejection					0.53	0.10	9.58

Table: DNN and CNN prediction times (threshold = 0.5).

Test	N events	total time[h]	ev/sec	sec/ev
Full reco removal	200	1.00	0.06	18.06
CNN all	836	4.20	0.06	18.08
CNN classification	836	1.90	0.12	8.19

Table: Reconstruction duration after CNN.

Conclusion

- Machine learning software was adapted for the TPC clusters classification.
- Two neural network models: dense and convolutional networks (DNN, CNN) have been implemented and optimized.
- Integration with standard NA61/SHINE software (rewrote the CNN and DNN algorithms from Python to C++ using Tensorflow C++).
- Modules for labeling signal and noise clusters and for noise rejection have been implemented.
- Performance time, parametric cluster classification, and Receiver Operating Characteristic (ROC) analysis for both CNN and DNN were performed.
- The analysis shows that CNN has better accuracy when compared to DNN, but classification in CNN takes longer compared to DNN.
- Ready-to-use ML modules in the NA61/SHINE software in the production TDAQ system (containerization using Docker was applied).

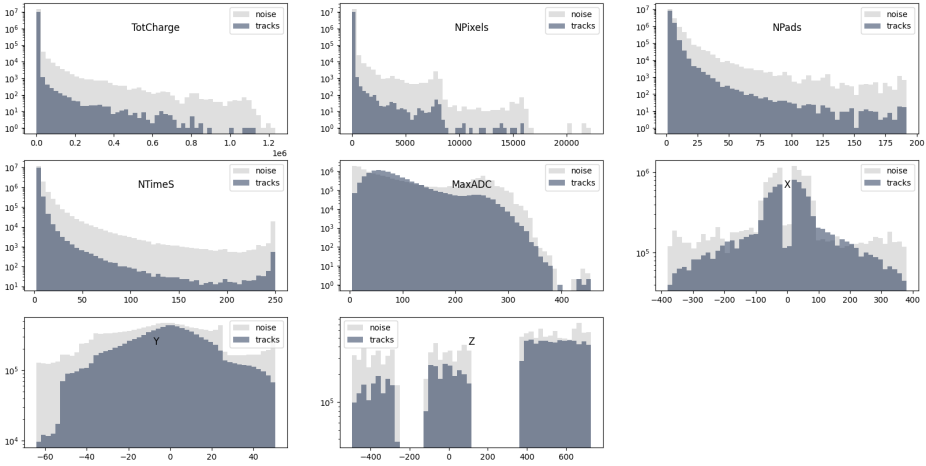
Acknowledgments

We extend our thanks to the NA61/SHINE collaboration for the opportunity to conduct this research. This work was supported by the WUT IDUB.

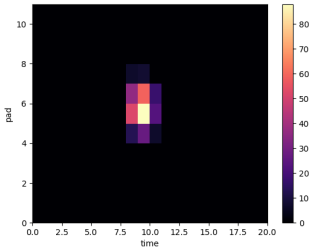
Thank You!

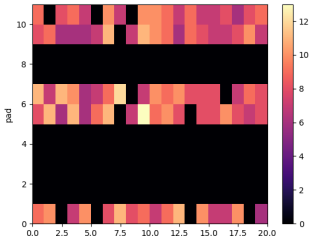
Supplementary material!

DNN input data: reconstructed properties of the clusters



CNN input data: 2D-Maps (raw cluster data)





- Data must be prepared for the analysis training for neural networks.
- Reconstruct local tracks
- Divide clusters into 2 groups: good clusters (belonging to track) and noisy clusters.
- For each cluster find pad ID and timestamp of max charge deposit
- Include signal from ± 5 neighbouring pads and ± 9 timestamps training.

DNN network's architecture

- The best dense network model has the following structure found using Keras Tuner, HyperBand method.

Layer	Size	Activation
Dense	126	ReLu
Dense	120	ReLu
Dense	52	ReLu
ReLu	1	Sigmoid

Table: DNN architecture of the network

CNN network's architecture

- Average pooling after each convolutional layer
- 30% dropout after each hidden dense layer
- Number of parameters: 5 000 $\rightarrow$ to avoid overfitting and shorten training and testing time
- Hyperparameter optimization – sklearn.GridSerachCV()

Layer	Size	Activation
Conv2D	32	ReLu
Conv2D	64	ReLu
Dense	256	ReLu
Dense	64	ReLu
Dense	128	ReLu
Dense	1	Sigmoid

Table: CNN architecture of the network

Performance tests (Pb+Pb collisions at 150 GeV/c)

raw to root	N of events	time[h]	ev/sec	sec/ev
DNN	836	0.58	0.40	2.50
CNN	200	0.73	0.08	13.18

Table: Reconstruction time that must be performed to prepare the data for training for DNN and CNN.

```
clusters  clusters.cc  Makefile  prepTrainingSet.sh  shine-env.sh
[root@shine-tf] TrainingSetPreparer #> . prepTrainingSet.sh
Writing data to a file: /data/clustered_local_tracked_test_run-041741x00894_vbuf2_2022-11-23_00-31-19.root.csv
Read 836 number of events
Execution time was 232 seconds.
Writing data to a file: /data/clustered_local_tracked_train_run-041743x00994_vbuf4_2022-11-23_07-36-16.root.csv
Read 133 number of events
Execution time was 31 seconds.
[root@shine-tf] TrainingSetPreparer #>
```

Figure: Reconstruction time that must be performed to prepare the data for training for DNN.