Development of data analysis applications for the WarsawTPC experiment – report

Maciej Bajor
Przemysław Szyc
Szymon Sławiński

Supervisor: dr hab. Artur Kalinowski

January 30, 2024

1 Introduction

Data analysis in the WarsawTPC [1] experiment, which studies the phenomenon of photodisintegration of oxygen and carbon atoms, has so far been carried out using programs from the TPCReco [2] repository, written in C++. An important element of this software, key when fine-tuning analysis tools, is event simulation using the Monte Carlo method.

In order to improve the quality of automatic analysis of experimental data, a set of Python programs have been created in 2023, enabling the reconstruction and analysis of events using machine learning.

2 Aims of the team project

Goals of the project focused on improving track reconstruction using machine learning. For this purpose, it was necessary to prepare appropriate input data from the simulations that would correspond to the data from the experiment - hence the need to perform comparisons as well as corrections in the simulation code to obtain the best possible compliance with the experimental data.

Tasks in the area of machine learning included optimizing the input data format from the point of view of throughput, i.e. accelerating data reading from `.ROOT` [3] files, training a model that recognizes track end points, testing the model's performance and comparing it with the current algorithm on simulated and real data. In the future, the model should be integrated with the TPCReco environment using the C++ API from the TensorFlow package [4].

3 Repositories and workspaces of the project

Changes to the code were being made in two repositories on GitHub [5] - a branch of the main TPCReco [6] repository, as well as in the files of the MachineLearning [7] repository, responsible for training and validation of machine learning models. Approved changes to the code were merged with branches preceded by a ZPS prefix.

In addition to the above, the team used a Notion [8] teamspace for the purposes of creating and dividing tasks, submitting partial reports to the project supervisor, and writing down instructions for certain activities.

Communication within the team took place via chat in Google Workspace and at meetings with the supervisor, organized every two weeks. The file exchange took place using a shared folder on Google Drive.

Computing-intensive tasks were initially performed in Google Colab [9] and locally, then on a faculty computer without a graphics co-processor, and in the end access to ICM's HPC [10] machines has been granted.

4 Monte Carlo simulations

Activities in the field of event simulation focused on providing the necessary training and validation sets for machine learning programs, as well as comparing track alignment with real data.

A simple addition has been made to the `TPCDigitizerRandom` code, allowing for the determination of the blur of tracks with effective diffusion in a flat distribution; the minimum and maximum sigma values can be provided in the Monte Carlo configuration file as follows:

```
"TPCDigitizerRandom": {
      "sigmaXYmin": 0.75,
      "sigmaXYmax": 1.5,
      "sigmaZmin": 0.75,
      "sigmaZmax": 1.5,
      "NSamplesPerHit": 100,
      "MeVToChargeScale": 100000
    }
```

Comparisons of the simulations with real data were also made, using fits from GUI. With a fixed vertex, tracks were generated according to the fit parameters and beam energy. The comparisons made indicate discrepancies in special cases, mainly when the reaction products moved along one of the detector axes (an example event is presented in Figure 1). For the purposes of making comparisons, corrections were also made to the code, including the ability to set axis limits in the *plf.plotEvent* function from the MachineLearning repository, and changes have been prepared for the simulation modules `ReactionTwoProng` and `EventGenerator` from the TPCReco [6] repository, which still require consultation with the author of the original code.

With respect to data generation, GUI use, and event reconstruction, instructions have been created in Notion [11] as part of documentation improvements to make it easier to become familiar with the tools of TPCReco software.

5 Machine Learning

The position of particle tracks can be reconstructed in the target format, i.e. in XYZ coordinates, and for each projection separately, in the UVWT coordinate system [12].

In the case of the XYZ target format, the model, whose input data are arrays with dimensions (256, 512, 3) - (strip number, time interval number, projection number) - representing readings from the detectors, reconstructs the positions of three points in the Cartesian coordinate system, i.e. the output space consists of three 3-dimensional vectors.

Reconstruction in the UVWT format means that for each subspace (UT, WT, VT) the reconstruction is performed independently. The input is the array (256, 512, 1), and the output are the coordinates of the projections of points on a given subspace (two 3-element vectors). This reconstruction also requires the use of a function that maps vectors from the UT, VT, WT space to the XYZ space after the reconstruction.

Both of these approaches were developed during the project:

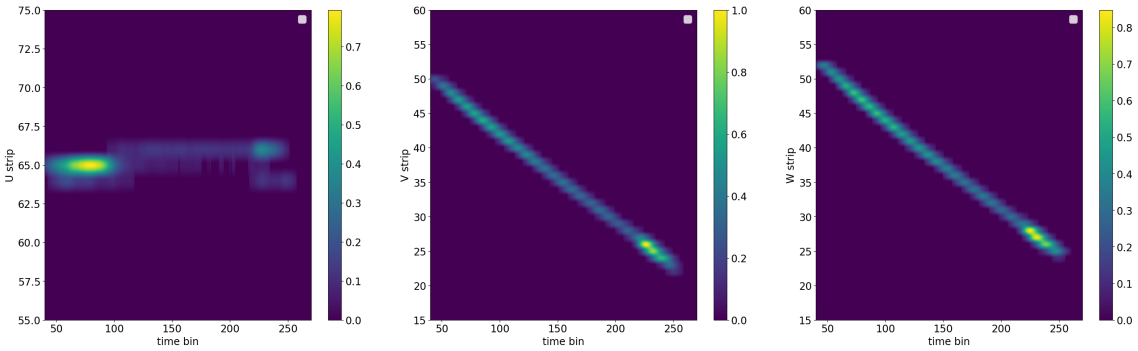
5.1 Reconstruction in XYZ coordinates

In the context of this method, the following tasks have been done:

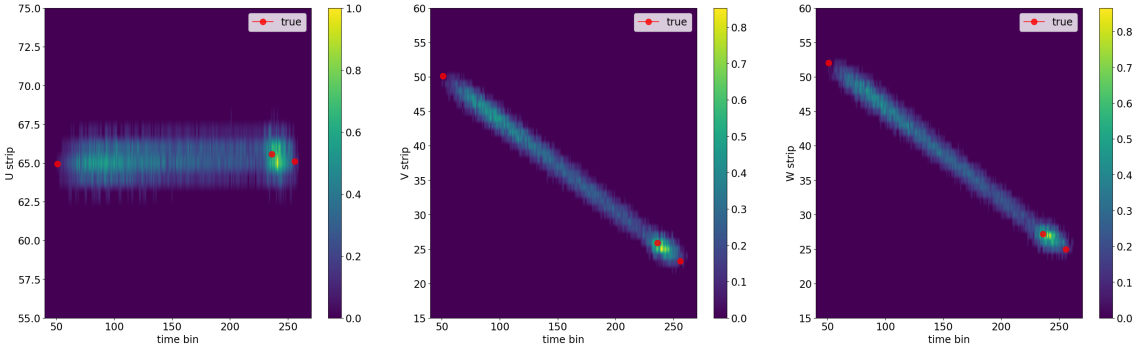
1. Enabled data conversion from .root format to .tfrecord format;
2. A Jupyter Notebook [13] has been created in which it is possible to train the model in the Google Colab environment.

Data conversion takes place in the notebook `ROOT_to_TFRecord.ipynb`, the function *process_and_save* is responsible for the conversion. The output data format (target in XYZ or UVWT variables) is selected by providing an appropriate argument to the *process_and_save* function.

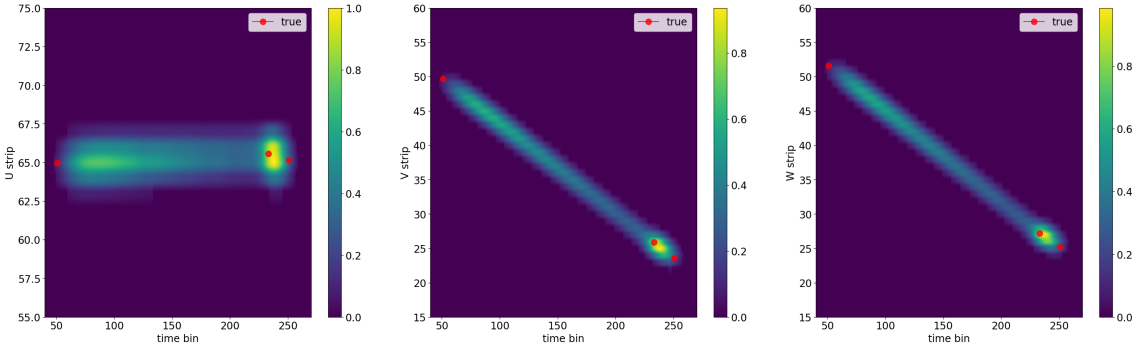
Working in the Google Colab environment was made possible by adding the *ZPS_colab-friendly* branch and creating versions of notebooks integrated with this environment and Google Drive.



(a) Event No. 1014 from experimental data portion.



(b) Event simulated using `TPCDigitizerRandom` with the introduced change (effective diffusion of 1.5 mm).



(c) Event simulated using `TPCDigitizerSRC` (effective diffusion of 1.5 mm).

Figure 1: Comparison of an event read from experimental data for beam energy $E = 11.5$ MeV with events simulated based on the GUI fit (red dots are the vertex and endpoints of tracks determined from the fit).

5.2 Reconstruction in UVWT coordinates

Tasks done to develop this method:

1. The ability to convert the target variable to UVWT coordinates has been added to the code for changing the data format from .root to .tfrecord;
2. ResNet-50 [14] models and a convolutional model (shown in Figure 2) have been prepared for training in this format;
3. Functions for data visualization have been integrated with these models.

The notebook `model_uvwt.ipynb` contains an example of using the data visualization function (from the module `plotting_functions.py`) with the model reconstructing based on projections, along with the training results.

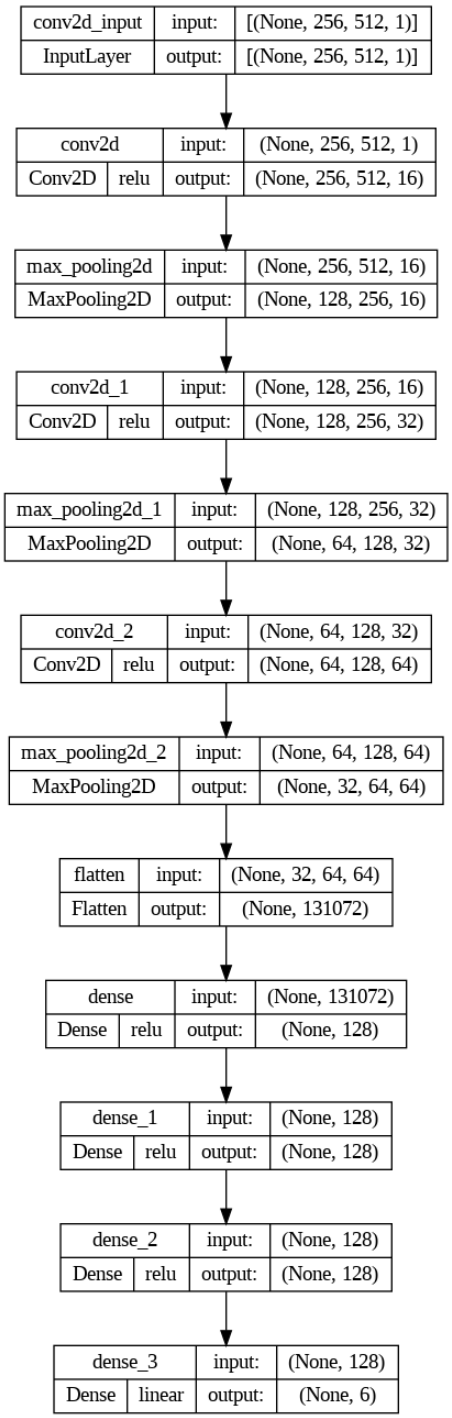


Figure 2: Diagram of the convolutional model used.

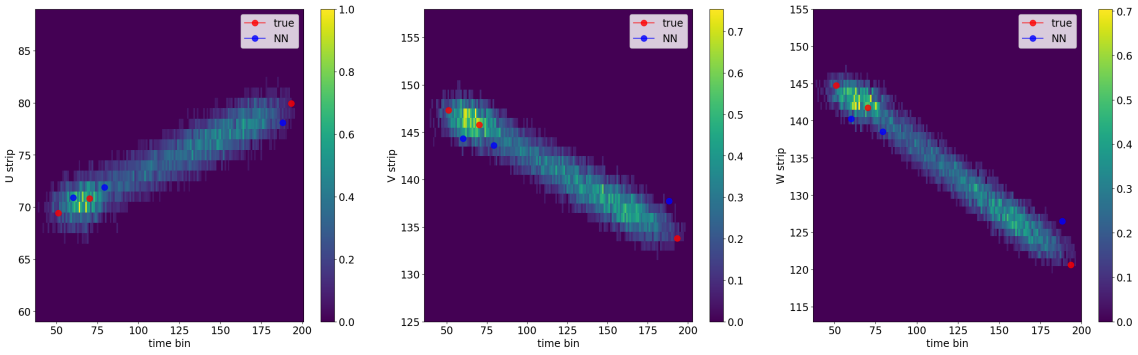


Figure 3: Reconstruction of the positions of points obtained in simulations (red) by the model (blue) for an example event from the test set.

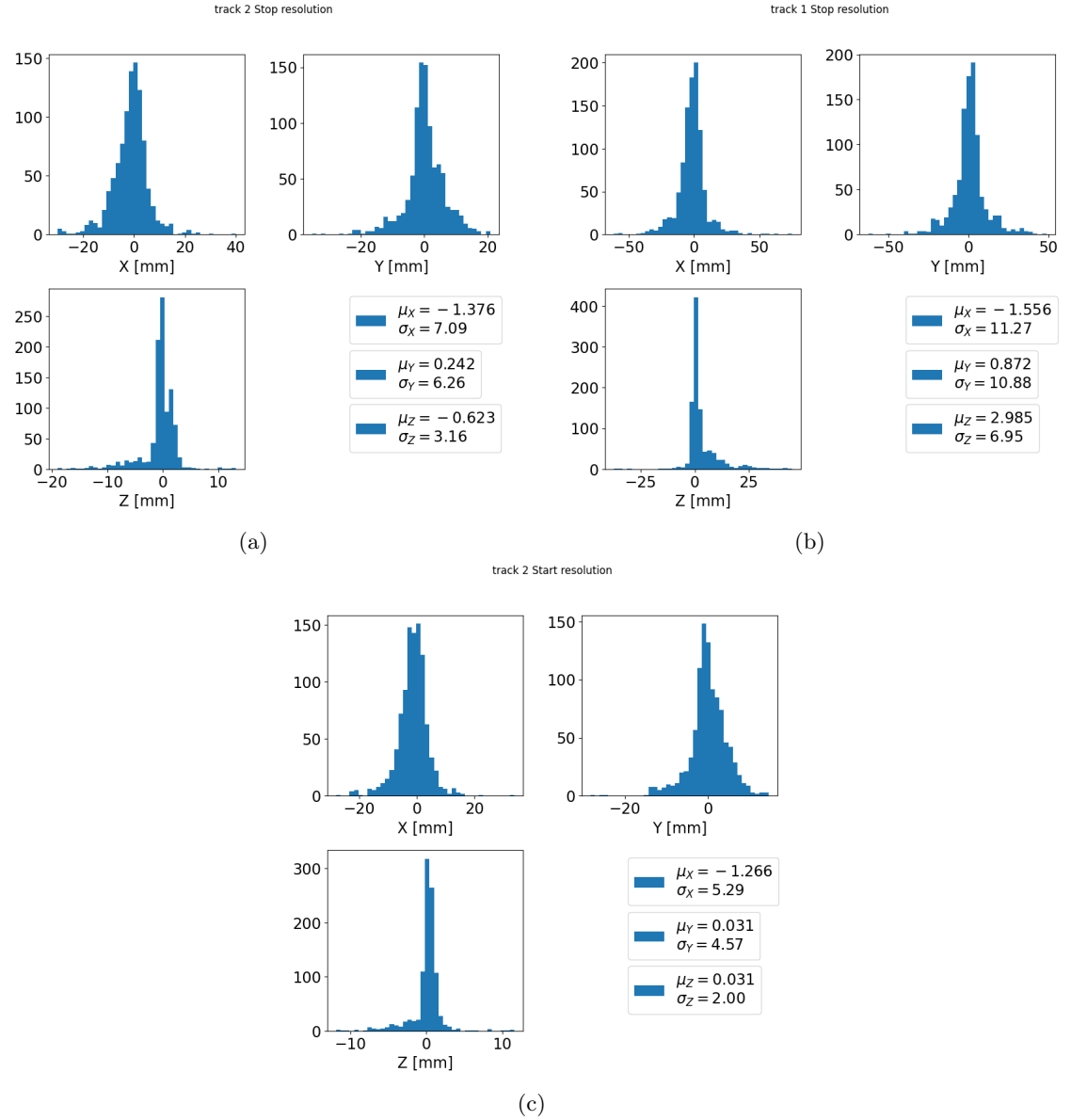


Figure 4: Resolution plots of point coordinates in the reconstruction performed by the convolutional model for examples from the test set.

In the `resnet50.ipynb` notebook there is an implementation of the ResNet-50 model, which, however, has not been trained due to hardware limitations.

5.3 Conversion of `.root` files to `.tfrecord`

To speed up data loading by the model, a program was written to convert `.root` files to `.tfrecord`.

The program loads charge maps from the `.root` file. Then, using the `multiprocessing` module, several maps are simultaneously converted to binary data and saved to separate `.tfrecord` files. The use of multiprocessing allows to significantly speed up the conversion process. An additional advantage is that loading data in the `.tfrecord` format from several files is faster than from a single one [15].

As a simple test, the time of loading 20,000 events from the `.root` and `.tfrecord` files was compared. Loading data from the `.root` file took 10 min 7 s, and from the `.tfrecord` files – 1 min 9 s, which is almost 9 times faster. Conversion from `.root` to `.tfrecord` took 8 min 27 s.

The code and demonstration of using the `.root` to `.tfrecord` conversion are available in the notebook `ROOT_to_TFRecord.ipynb`. It is possible to save outputs in XYZ or UVWT coordinates. The notebook also demonstrates loading data from the `.tfrecord` file.

5.4 Further actions

Suggested next steps for machine learning include:

1. Training the ResNet-50 model in both data formats and comparing the models' performance;
2. Improving the performance of existing models by changing the architecture and using a different dataset, in particular for reconstruction in UVWT coordinates;
3. Testing the generalization of the model on various types of data, especially experimental data.

6 Conversion of `.graw` files to `.root`

As part of maintenance work on the TPCReco repository, the program that converts `.graw` files to `EventTPC.root` has been restored. The program has been adapted to work with a configuration system using `.json` files, which is used by the rest of the programs in the repository.

6.1 Test of `grawToEventTPC`

Added a `grawToEventTPC` test which:

1. tests the validity of the function that creates the file name `.root`
2. converts 1 event from test file `.graw` to `.root`
3. checks whether the `.root` file opens
4. checks whether the number of events is correct
5. compares charge maps loaded from `.graw` and `.root`
6. checks whether the `.root` file closes
7. deletes the `.root` file

6.2 Next steps

Another task to be performed to improve the development of the TPCReco application may be to implement GitHub Actions [16]. GitHub Actions allow the code owners to automatically perform certain actions in response to events happening in the repository. For example, it could automatically compile the application and run tests in response to a new pull request.

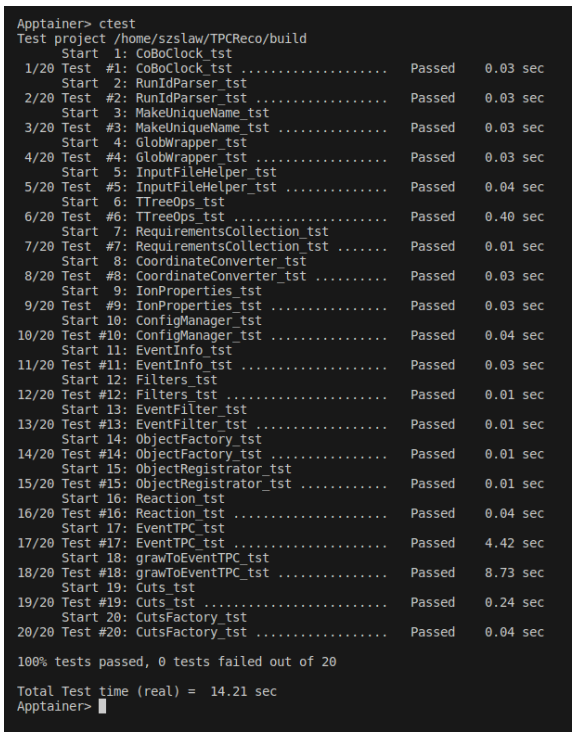


Figure 5: A `ctest` command result – all tests passed.

7 Summary

The team project was carried out in the winter semester of the 2023/24 academic year at the Faculty of Physics of the University of Warsaw. Within the scope of the goals agreed with the project supervisor and taking into account the size of the team, it has improved applications for analyzing data from the WarsawTPC experiment, both in the area of preparing simulated and real data, as well as improving the reconstruction process using machine learning. A know-how base has also been built, which will allow for more efficient introduction of new participants to the project.

Bibliography

[1] Wojciech Dominik. *Nuclear Astrophysics Studied With TPCs Operating in Gamma-Beams: Warsaw Active Target TPC*. URL: `https://indico.duke.edu/event/1/contributions/33/attachments/28/39/nuclear_photonics_2023_WD.pdf`. Sept. 2023.

[2] *Track reconstruction for TPC data with 2D projections readout*. URL: `https://github.com/akalinow/TPCReco` (visited on 01/29/2024).

[3] *ROOT files in ROOT analysis framework*. URL: `https://root.cern/manual/root_files/` (visited on 01/29/2024).

[4] *TensorFlow C++ API Reference*. URL: `https://www.tensorflow.org/api_docs/cc` (visited on 01/29/2024).

[5] *GitHub platform*. URL: `https://github.com/` (visited on 01/29/2024).

[6] *TPCReco fork*. URL: `https://github.com/mwbaj/TPCReco` (visited on 01/29/2024).

[7] *MachineLearning fork, WAWTPC folder*. URL: `https://github.com/mwbaj/MachineLearning/tree/ZPS%5C_2023%5C_winter/WAWTPC` (visited on 01/29/2024).

[8] *Notion teamspace*. URL: `https://akalinow.notion.site/Teamspace-Home-3ffdcdf4b5c44a3687be98a8979a6249?pvs=4` (visited on 01/29/2024).

[9] *Google Colab*. URL: `https://colab.research.google.com/notebooks/intro.ipynb` (visited on 01/29/2024).

[10] *ICM computer systems*. URL: `https://kdm.icm.edu.pl/Zasoby/komputery_w_icm.en/` (visited on 01/28/2024).

[11] *User Documentation*. URL: https://akalinow.notion.site/User-documentation-c5f6fa3d4957476d9240b77b6213572e (visited on 01/29/2024).

[12] Mikołaj Ćwiok. *Nuclear reactions of astrophysical interest with gamma-ray beams [detector description]*. URL: http://indico.fuw.edu.pl/getFile.py/access?contribId=2&sessionId=0&resId=0&materialId=slides&confId=49. Nov. 2016.

[13] *WAWTPC_ML.ipynb*. URL: https://github.com/mwbaj/MachineLearning/blob/ZPS_colab-friendly/WAWTPC/WAWTPC_ML.ipynb (visited on 01/29/2024).

[14] Kaiming He et al. "Deep residual learning for image recognition." In: *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 770–778.

[15] *TFRecord and tf.train.Example*. URL: https://www.tensorflow.org/tutorials/load_data/tfrecord (visited on 01/27/2024).

[16] *GitHub Actions*. URL: https://github.com/features/actions (visited on 01/29/2024).