Rozwój aplikacji do analizy danych w eksperymencie WarsawTPC – raport

Maciej Bajor
Przemysław Szyc
Szymon Sławiński

Opiekun: dr hab. Artur Kalinowski

30 stycznia 2024

1 Wstęp

Analiza danych w eksperymencie WarsawTPC [1], zajmującym się badaniem zjawiska fotodezintegracji atomów tlenu i węgla, odbywała się do tej pory przy pomocy programów z repozytorium TPCReco [2], napisanych w języku C++. Istotnym elementem tego oprogramowania, kluczowym przy dostrajaniu narzędzi do analizy, są symulacje zdarzeń metodą Monte Carlo.

W celu poprawy jakości automatycznej analizy danych z eksperymentu stworzono w 2023 roku zestaw programów napisanych w języku Python, umożliwiających rekonstrukcję i analizę zdarzeń z wykorzystaniem uczenia maszynowego.

2 Cel projektu zespołowego

Założenia projektu skupiały się na poprawie działania rekonstrukcji torów z użyciem uczenia maszynowego. Do tego celu potrzebne było przygotowanie odpowiednich danych wejściowych z symulacji, które odpowiadałyby danym z eksperymentu - stąd zaszła konieczność wykonania porównań, a także poprawek w kodzie do symulacji, aby uzyskać jak najlepszą zgodność z danymi doświadczalnymi.

Zadania w obszarze uczenia maszynowego obejmowały optymalizację formatu danych wejściowych z punktu widzenia przepustowości, a więc przyspieszenie odczytywania danych z plików w formacie `.ROOT` [3], trening modelu rozpoznającego punkty krańcowe torów, test wydajności modelu i porównanie z obecnym algorytmem na danych symulowanych i rzeczywistych. W dalszej przyszłości powinna nastąpić integracja modelu ze środowiskiem TPCReco z użyciem API do C++ z pakietu TensorFlow [4].

3 Repozytoria i obszary robocze projektu

Zmiany w kodzie wprowadzane były w dwóch repozytoriach w serwisie GitHub [5] - odgałęzieniu głównego repozytorium TPCReco [6], oraz w plikach repozytorium MachineLearning [7], odpowiedzialnych za trening i sprawdzanie dokładności modeli uczenia maszynowego. Zaakceptowane zmiany w kodzie trafiały do gałęzi poprzedzonych przedrostkiem ZPS.

Oprócz powyższych, zespół korzystał z przestrzeni w Notion [8] na potrzeby związane z tworzeniem i podziałem zadań, przekazywaniem raportów cząstkowych opiekunowi projektu, oraz spisywaniem instrukcji do niektórych czynności.

Komunikacja w zespole odbywała się na czacie w Google Workspace oraz na spotkaniach z opiekunem, organizowanych co dwa tygodnie. Wymiana plików miała miejsce za pomocą folderu współdzielonego na Dysku Google.

Zadania wymagające mocy obliczeniowej wykonywano początkowo w Google Colab [9] i lokalnie, następnie na komputerze wydziałowym bez koprocesora graficznego, a pod koniec uzyskano dostęp do maszyn HPC ICM [10].

4 Symulacje Monte Carlo

Działania w zakresie symulacji zdarzeń skupiały się na zapewnieniu potrzebnych zbiorów treningowych i walidacyjnych do uczenia maszynowego, a także na porównaniu zgodności przebiegu torów z rzeczywistymi danymi. Wprowadzono do kodu `TPCDigitizerRandom` prostą poprawkę, pozwalającą na ustalanie rozmycia torów w rozkładzie płaskim efektywnej dyfuzji; minimalne i maksymalne wartości sigm można podać w pliku konfiguracyjnym Monte Carlo w następujący sposób:

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

Wykonano także porównania symulacji z danymi rzeczywistymi, z wykorzystaniem dopasowań z GUI. Przy ustalonym wierzchołku generowano tory zgodnie z parametrami dopasowań i energią wiązki. Wykonane porównania wskazują na rozbieżności w szczególnych przypadkach, głównie gdy produkty reakcji poruszały się wzdłuż którejś z osi detektora (przykładowe zdarzenie znajduje się na Rysunku 1). Na potrzeby wykonania porównań również wprowadzono poprawki w kodzie, między innymi zaimplementowano możliwość ustalania granic osi w funkcji *plf.plotEvent* z repozytorium MachineLearning [7], a także przygotowano do wprowadzenia zmiany w modułach symulacji `ReactionTwoProng` i `EventGenerator` z repozytorium TPCReco [6], które wymagają jeszcze konsultacji z autorem oryginalnego kodu.

W odniesieniu do generowania danych, korzystania z GUI i rekonstrukcji zdarzeń, stworzono instrukcje w Notion [11] w ramach ulepszania dokumentacji, aby łatwiej było zaznajomić się z narzędziami pakietu TPCReco.

5 Uczenie Maszynowe

Rekonstrukcji położenia torów cząstek można dokonywać w formacie docelowym, czyli we współrzędnych XYZ, oraz dla każdej projekcji oddzielnie, w układzie odniesienia UVWT [12].

W przypadku formatu docelowego XYZ model, którego danymi wejściowymi są tablice o wymiarach (256, 512, 3) - (numer paska, numer przedzialu czasowego, numer rzutu) - reprezentujące odczyty z detektorów, dokonuje rekonstrukcji położeń trzech punktów w kartezjańskim układzie współrzędnych, tzn. przestrzeń wyjść składa się z trzech wektorów 3-wymiarowych.

Rekonstrukcja w formacie UVWT polega na tym, że dla każdej podprzestrzeni (UT, WT, VT) rekonstrukcja wykonywana jest niezależnie. Wektorem wejściowym są tablice (256, 512, 1), a wyjściem współrzędne rzutów punktów na daną podprzestrzeń (dwa wektory 3-elementowe). Rekonstrukcja ta wymaga też użycia funkcji mapującej wektory z przestrzeni UT, VT, WT do przestrzeni XYZ po dokonaniu rekonstrukcji.

Podczas projektu oba te podejścia były rozwijane:

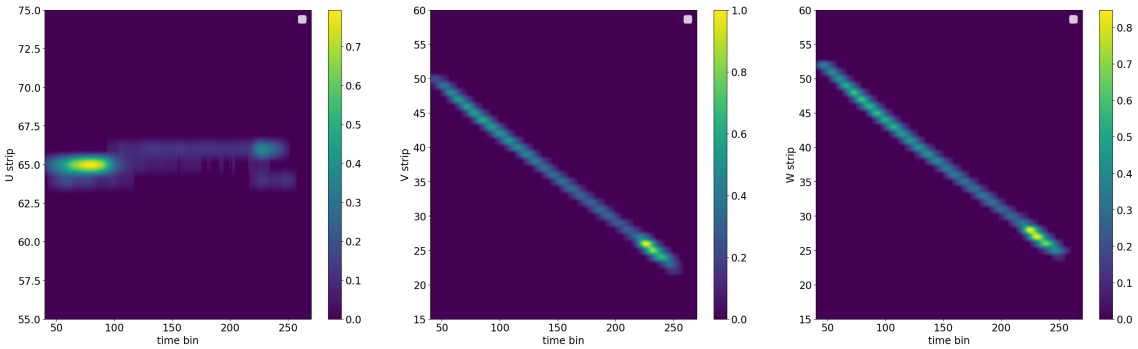
5.1 Rekonstrukcja we współrzędnych XYZ

W kontekście tej metody wykonano następujące zadania:

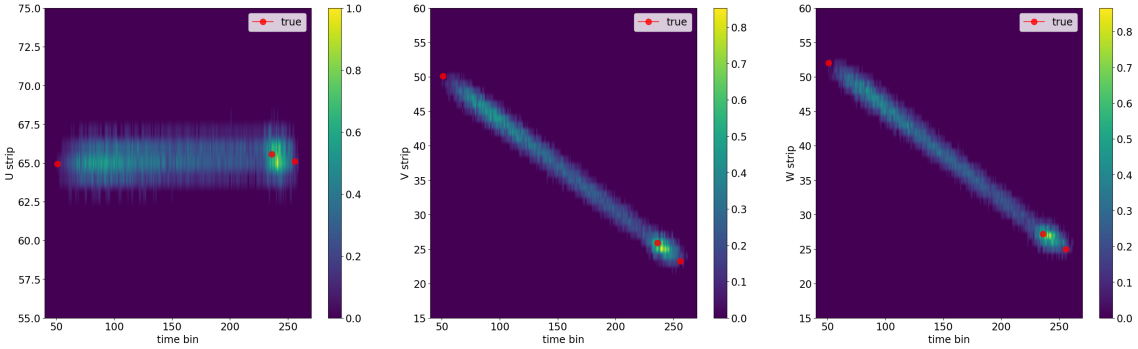
1. Umożliwiono konwersję danych z formatu .root do formatu .tfrecord;
2. Stworzono notatnik [13], w którym możliwym jest trenowanie modelu w środowisku Google Colab.

Konwersja danych odbywa się w notatniku `ROOT_to_TFRecord.ipynb`, za konwersję odpowiedzialna jest funkcja *process_and_save*. Format danych wyjściowch (target w zmiennych XYZ czy UVWT) wybierany jest poprzez podanie odopwiedniego argumentu do funkcji *process_and_save*.

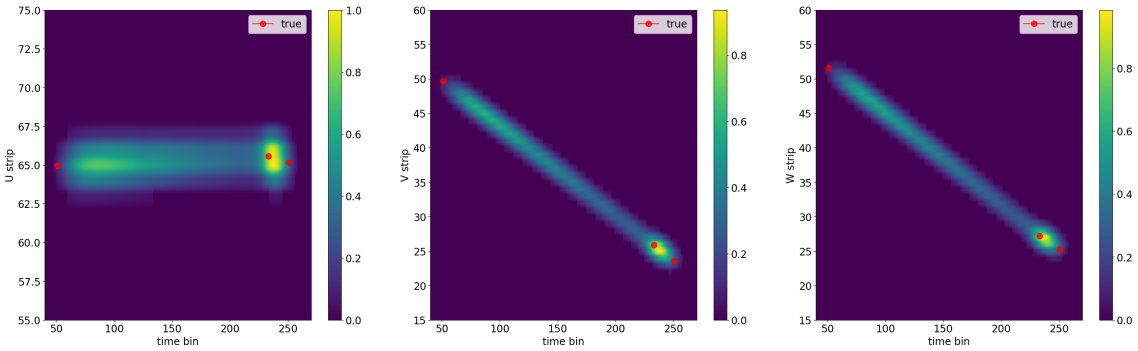
Praca w środowiku Google Colab została umożliwiona poprzez dodanie gałęzi *ZPS_colab-friendly* i stworzenie wersji notatników zintegrowanych z tym środowiskiem oraz Dyskiem Google.



(a) Zdarzenie nr 1014 z porcji danych eksperymentalnych.



(b) Zdarzenie symulowane z wykorzystaniem `TPCDigitizerRandom` z wprowadzoną poprawką (efektywna dyfuzja 1,5 mm).



(c) Zdarzenie symulowane z użyciem `TPCDigitizerSRC` (efektywna dyfuzja 1,5 mm).

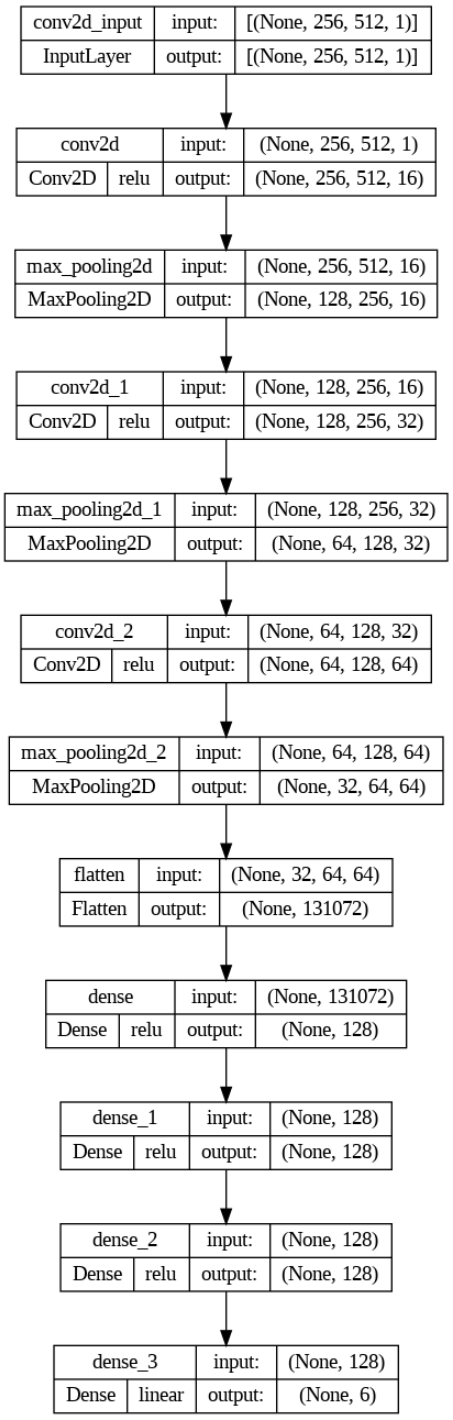
Rysunek 1: Porównanie zdarzenia odczytanego z danych eksperymentalnych dla energii wiązki $E = 11,5$ MeV ze zdarzeniami symulowanymi na bazie dopasowania z GUI (czerwone punkty to położenia wierzchołka i krańców torów wyznaczone na bazie dopasowania).

5.2 Rekonstrukcja w współrzędnych UVWT

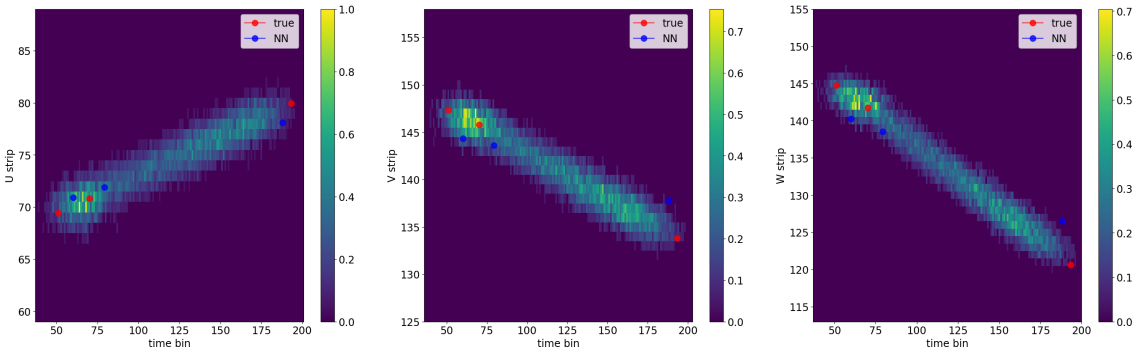
Wykonane zadania w celu rozwijania tej metody:

1. Do algorytmu zmiany formatu danych z .root do .tfrecord dodano możliwość konwersji zmiennej target do współrzędnych UVWT;
2. Przygotowano modele ResNet-50 [14] oraz początkowy model konwolucyjny (schemat na Rysunku 2) do treningu w tym formacie;
3. Zintegrowano funkcje do wizualizacji danych z działaniem tych modeli.

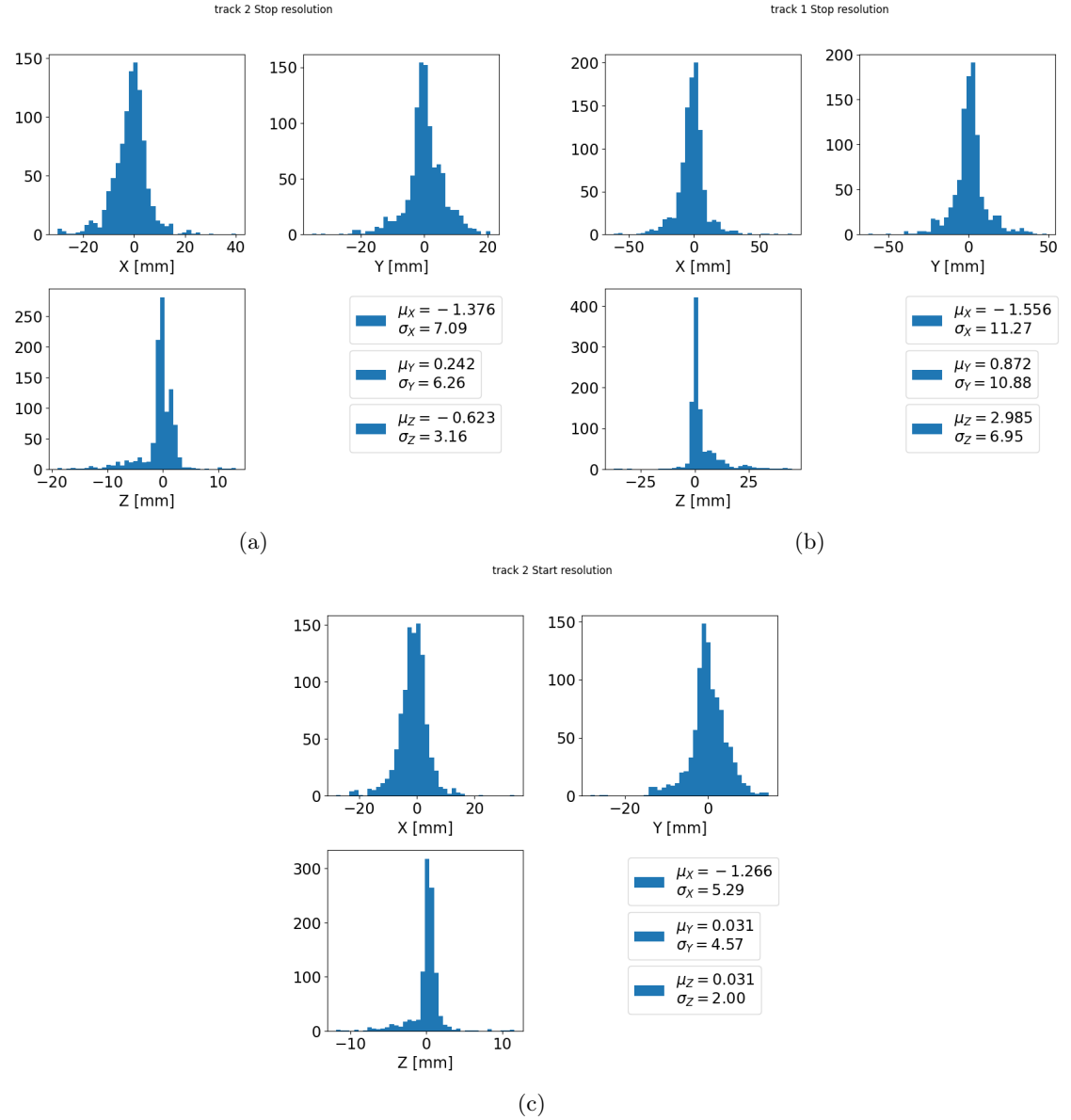
W notatniku `model_uvwt.ipynb` zawarty jest przykład użytkowania funkcji do wizualizacji danych (z modułu `plotting_functions.py`) z modelem rekonstrukcji po projekcjach, wraz z rezultatami treningu.



Rysunek 2: Schemat użytego modelu konwolucyjnego.



Rysunek 3: Rekonstrukcja położeń punktów uzyskanych w symulacjach (kolor czerwony) przez model (kolor niebieski) dla przykładowego zdarzenia ze zbioru testowego.



Rysunek 4: Wykresy rozdzielczości współrzędnych punktów w dokonywanej przez model konwolucyjny rekonstrukcji dla przykładów ze zbioru testowego.

W `resnet50.ipynb` znajduje się implementacja modelu ResNet-50, który jednak nie został wytrenowany z racji ograniczeń sprzętowych.

5.3 Konwersja plików `.root` na `.tfrecord`

W celu przyśpieszenia wczytywania danych przez model napisano program konwertujący pliki `.root` na `.tfrecord`.

Program wczytuje mapy ładunku z pliku `.root`. Następnie, wykorzystując moduł `multiprocessing`, kilka map jednocześnie jest konwertowanych na dane binarne i zapisywanych do oddzielnych plików `.tfrecord`. Zastosowanie multiprocessingu pozwala znacznie przyśpieszyć proces konwersji. Dodatkową zaletą jest to, że wczytywanie danych w formacie `.tfrecord` z kilku plików jest szybsze niż z jednego [15].

W ramach prostego testu porównano czas wczytania 20 000 zdarzeń z pliku `.root` i `.tfrecord`. Wczytywanie danych z pliku `.root` zajęło 10 min 7 s, a z plików `.tfrecord` 1 min 9 s, co daje prawie 9-krotne przyśpieszenie. Konwersja z `.root` na `.tfrecord` zajęła 8 min 27 s.

Kod oraz demonstracja użycia konwersji `.root` na `.tfrecord` dostępny jest w notatniku `ROOT_to_TFRecord.ipynb`. Istnieje możliwość zapisania wyjść we współrzędnych XYZ lub UVWT. W notatniku zademonstrowane jest też wczytywanie danych z pliku `.tfrecord`.

5.4 Dalsze działania

Sugerowane kolejne działania dotyczące uczenia maszynowego to np:

1. Wytrenowanie modelu ResNet-50 w obu formatach danych i porównanie działania modeli;
2. Poprawa wydajności dotychczasowych modeli poprzez zmianę architektury oraz użycie innego zbioru danych, w szczególności dla rekonstrukcji we współrzędnych UVWT;
3. Badanie generalizacji modelu do różnych rodzajów danych, w szczególności danych pochodzących z eksperymentu.

6 Konwersja plików `.graw` na `.root`

W ramach prac konserwacyjnych nad repozytorium TPCReco przywrócono do działania program konwertujący pliki `.graw` na `EventTPC.root`. Program przystosowano do działania z systemem konfiguracji używającym plików `.json`, który stosowany jest przez resztę programów w repozytorium.

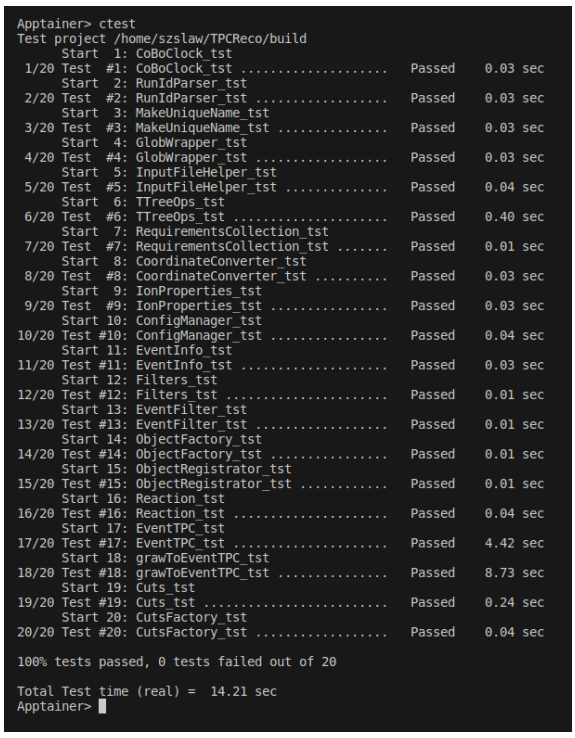
6.1 Test `grawToEventTPC`

Dodano test `grawToEventTPC` który:

1. testuje poprawność funkcji tworzącej nazwę pliku `.root`
2. konwertuje 1 zdarzenie z testowego pliku `.graw` na `.root`
3. sprawdza czy plik `.root` się otwiera
4. sprawdza czy ilość zdarzeń jest poprawna
5. porównuje mapy ładunków załadowane z pliku `.graw` i `.root`
6. sprawdza czy plik `.root` się zamyka
7. usuwa plik `.root`

6.2 Dalsze kroki

Kolejnym zadaniem do wykonania w celu usprawnienia rozwijania aplikacji TPCReco może być zastosowanie GitHub Actions [16]. GitHub Actions pozwala na automatyczne wykonanie pewnych czynności w reakcji na zdarzenia dziejące się z repozytorium. Można na przykład automatycznie kompilować aplikację i uruchamiać testy w odpowiedzi na nowy pull request.



Rysunek 5: Rezultat komendy `ctest` – wszystkie testy z wynikiem pozytywnym.

7 Podsumowanie

Projekt zespołowy realizowany był w semestrze zimowym roku akademickiego 2023/24 na Wydziale Fizyki Uniwersytetu Warszawskiego. W zakresie celów uzgodnionych z opiekunem projektu, mając na uwadze liczebność zespołu, udało się rozwinąć aplikacje do analizy danych z eksperymentu WarsawTPC zarówno w obszarze przygotowania danych symulowanych i rzeczywistych, jak również usprawnień procesu rekonstrukcji z wykorzystaniem uczenia maszynowego. Zbudowano także bazę wiedzy, która pozwoli na sprawniejsze wprowadzenie nowych uczestników do projektu.

Bibliografia

[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`. Wrz. 2023.

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

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

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

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

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

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

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

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

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

[11] *User Documentation*. URL: https://akalinow.notion.site/User-documentation-c5f6fa3d4957476d9240b77b6213572e (term. wiz. 29. 01. 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. List. 2016.

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

[14] Kaiming He i in. „Deep residual learning for image recognition." W: *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition* (2016), s. 770–778.

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

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