Contribution ID: **194** Type: **Poster**

Integration of machine learning-trained models into JUNO's offline software

Wednesday, 26 October 2022 11:00 (30 minutes)

The Jiangmen Underground Neutrino Observatory (JUNO) is under construction in South China and will start data taking in 2023. It has a central detector with a 20-kt liquid scintillator, equipped with 17,612 20-inch PMTs (photo-multiplier tubes) and 25,600 3-inch PMTs. The requirement on energy resolution of 3\%@1MeV makes the offline data processing challenging, so several machine learning based methods have been developed for reconstruction, particle identification, simulation etc. These methods are implemented with machine learning libraries in Python, however, the offline software is based on a C++ framework called SNiPER. Therefore, how to integrate them and run the inference in offline software is important.

In this contribution, integration of machine learning-trained models into JUNO's offline software will be presented. Three methods are explored: using SNiPER's Python binding to share data between C++ and Python; using native C/C++ APIs of the machine learning libraries, such as TensorFlow and PyTorch; using ONNX runtime. Even though SNiPER is implemented in C++, it provides Python binding via Boost Python. In recent updates of SNiPER, a special data buffer is implemented to share data between C++ and Python, which makes it possible to run machine learning methods in following way: a C++ algorithm reads event data and converts them to \texttt{numpy} arrays; a Python algorithm then accesses these \texttt{numpy} arrays and invokes machine learning libraries in Python; finally, the C++ algorithm puts the results into event data. For the native C/C++ APIs of machine learning libraries and ONNX runtime, a C++ algorithm is used to convert the event data to the corresponding formats and invoke the C/C++ APIs. The deployments of the three methods are also studied: using SNiPER's Python binding is the most flexible method for users, as users could install any Python libraries using \texttt{pip} by themselves; using native C/C++ APIs requires the users to use the same versions in JUNO official software release; using ONNX runtime only requires users to convert their own models to ONNX format. By comparing the three methods, ONNX is recommended for most of users in JUNO. For developing and testing of machine learning-models in offline software, developers could choose the other two methods.

Significance

As a JUNO L3 manager for software release, I try to deploy the C/C++ API of the native machine learning libraries into JUNO offline in several years ago. But the requirements from developers are different, such as one uses TF 1.x, while another one uses TF 2.x. So that drives our core software group to find some more generic solutions. As mentioned in the abstract, SNiPER provides a data buffer, which could be accessed from both C++ and Python side. This feature lets us do the data conversion in C++ and do the inference in Python. Recently, we also notice that the ONNX is popular, such as Geant4 includes an example using ONNX. So we also apply it in our software.

References

1. An example to show the data interoperability in C++ and Python: https://github.com/JUNO-collaboration/offline-example-pyalg

Experiment context, if any

JUNO

Primary authors: Mr ZOU, Jiaheng; LIN, Tao (Chinese Academy of Sciences (CN)); Dr LI, Teng (Shandong University, CN); Dr LI, Weidong (Chinese Academy of Sciences (CN)); HUANG, Xingtao (Shandong University)

Presenter: LIN, Tao (Chinese Academy of Sciences (CN))

Session Classification: Poster session with coffee break

Track Classification: Track 1: Computing Technology for Physics Research