Welcome!

Contents

Overview

This Jupyter Book contains the final report for the 2022 RICH AI Capstone Project completed by Nico Van den Hooff, Rakesh Pandey, Mukund Iyer, and Shiva Jena as part of the University of British Columbia Master of Data Science program. The project was completed in partnership with TRIUMF, Canada's particle accelerator centre and one of the world's leading subatomic physics research centres.

Executive summary

The NA62 experiment at CERN (European organization for nuclear research) studies the rate of the ultra-rate meson decay into a pion to verify the Standard Model in physics. The aim of this project is to develop a binary classification model based on advanced Machine Learning (ML) to distinguish pion decays from muon decays using the output of the Ring Imaging Cherenkov (RICH). The challenge lies in concurrently increasing the pion efficiency and mion efficiency and surpassing the performance of the MLE algorithm. The data used to build the machine learning models had 2 million examples, controlled for momentum and converted to point cloud form by the addition of a time dimension. Two deep learning models were applied: PointNet and Dynamic Graph CNN (DGCNN). Both were built using the point clouds of hits, particle momentum and the ring radius computed using the MLE. The best performing PointNet model used all these features, with a time delta of 0.2 ns and 16 epochs of training. Likewise the best performing DGCNN used all the features, k = 8 nearest neighbors and a time delta of 0.3. The overall best performing model was PointNet as it has the highest AUC ROC, exceeds the pion efficiency from the MLE estimate for all momentum bins, and maintains a low muon efficiency for momentums beyond 34 GeV/c. Meanwhile, the DGCNN is able to maintain a similar pion efficiency but fails to maintain an adequate muon efficiency to surpass the MLE estimate. The final data product is a machine learning pipeline that takes in the experiment data in a HDP5 format, pre-processes it, prepares training data, trains classifier models on the training data and evaluates model results on test data through modular scripts.

Table of contents

Analysis and results

Appendix

- Supplementary Jupyter notebooks

References

- References

By Nico Van den Hooff, Rakesh Pandey, Mukund Iyer, Shiva Jena

Introduction

The NA62 experiment at CERN (European organization for nuclear research) studies the rate of the ultra-rate meson decay into a pion to verify the Standard Model in physics. However, a muon and positron may be produced in the decay as noise events. The latter can be identified experimentally, however it is difficult to distinguish between the pion and muon.

"*RICH AI*", initiated by TRIUMF (Canada's national particle accelerator center), is aimed at improving Particle Identification (PID) performance of the Ring Imaging Cherenkov (RICH) by developing a binary classification model based on advanced Machine Learning (ML). The classification will be based on the fundamental difference in the ring size generated by either particle at a specific momentum value.

The main challenge in this project is to increase pion efficiency, that is, the fraction of the matched detected pions out of the total number of pion producing events. Concurrently the muon efficiency needs to be maximized, which translates to minimizing the number of muons that are misclassified as pions out of the total muon events. These metrics will be compared to the output of the current method used for classification, the Maximum Likelihood Estimation, for events with momentum in the range of 15-45 GeV/c.

Ultimately, the output of the project is a repository containing fully modularized code for the classification. The scientists are able to select the dataset and hyperparameters, run the classification in the backend, and view the results along with the evaluation metrics as the output.

By Nico Van den Hooff, Rakesh Pandey, Mukund Iyer, Shiva Jena

Data Generation

Contents

In order to generate the NA62 data, several experiment “runs” are performed. For each run, the experiment configuration is fixed and the following steps are performed:

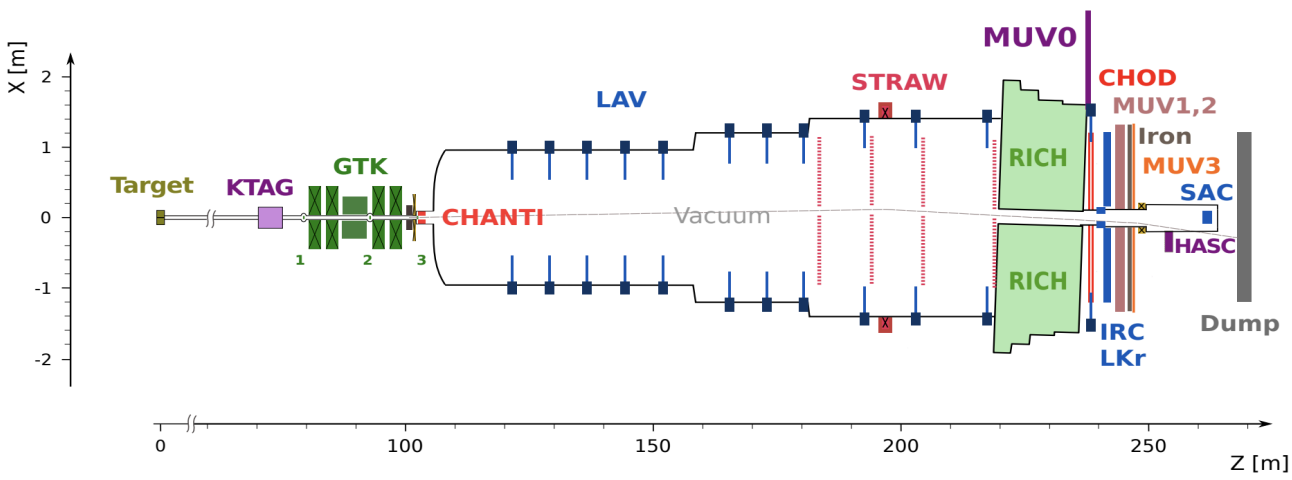

Fig. 1 Cross section of the setup of the NA62 experiment

1. A beam rich in kaon particles is delivered in “bursts” every four or five seconds into the detector. The set up as shown in Fig. 1 [Gil *et al.*, 2017].
2. During a burst, several particle decays occur. Each particle decay has an individual “event” ID associated with it.

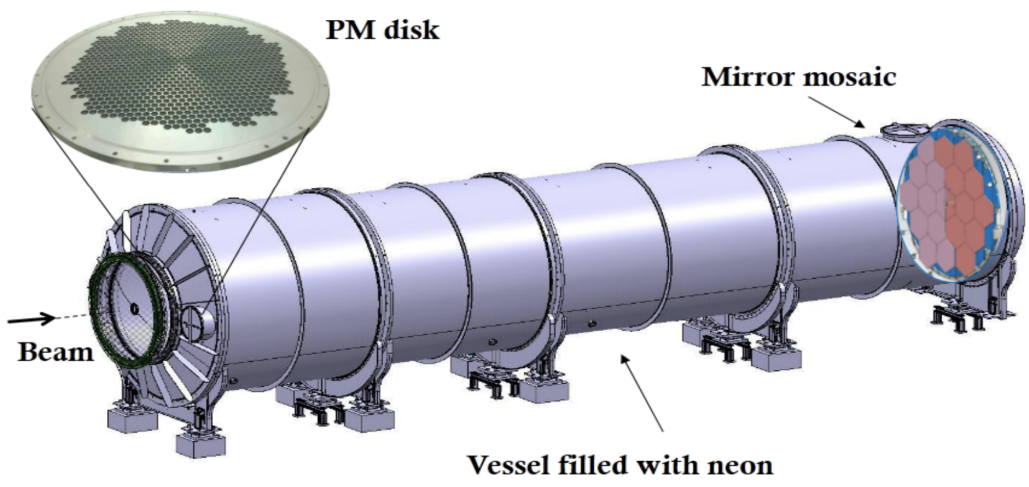

Fig. 2 Ring-imaging Cherenkov detector

1. The product of the decay is accelerated through a chamber of neon gas in the RICH detector and a cone of light is emitted. The RICH detector is shown in Fig. 2 [Anzivino *et al.*, 2020].
2. The cone of light is reflected by a mosaic of mirrors onto an array of photomultiplier tubes (“PMT”). In an ideal situation, the cone of light forms a “ring” on the PMT array.
3. Each individual PMT in the array records whether or not it was hit by light, as well as the time of arrival for each hit of light.
4. The hodoscope counters (CHOD) detector shown in Figure 2 records the time that the particle decay occurs.

Based on the experimental setup detailed above, two sets of features associated are derived for each event. The first set corresponds to the subatomic particle motion: the particle momentum and time spent in the detector (CHOD time). The second set of features are derived from the light emitted by the subatomic particle motion. Each photon detected by a PMT tube is recorded as a hit on the PMT grid with X and Y coordinates relative to this grid and the time of the hit. As an abstraction to the hit scatter, the ring radius and center that result from the MLE fit that TRIUMF currently employ are also included. All in all, there are a total of 5 features for each event.

Data volume

The data was generated as part of the 2018 NA62 experiments performed at CERN. There are a total of 11 million labeled decay events, each containing the features detailed above. However, there is a large class imbalance in the data set. Only 10% of the examples are of pions, the class of interest.

Preprocessing

Subsampling w/ respect to momentum bins

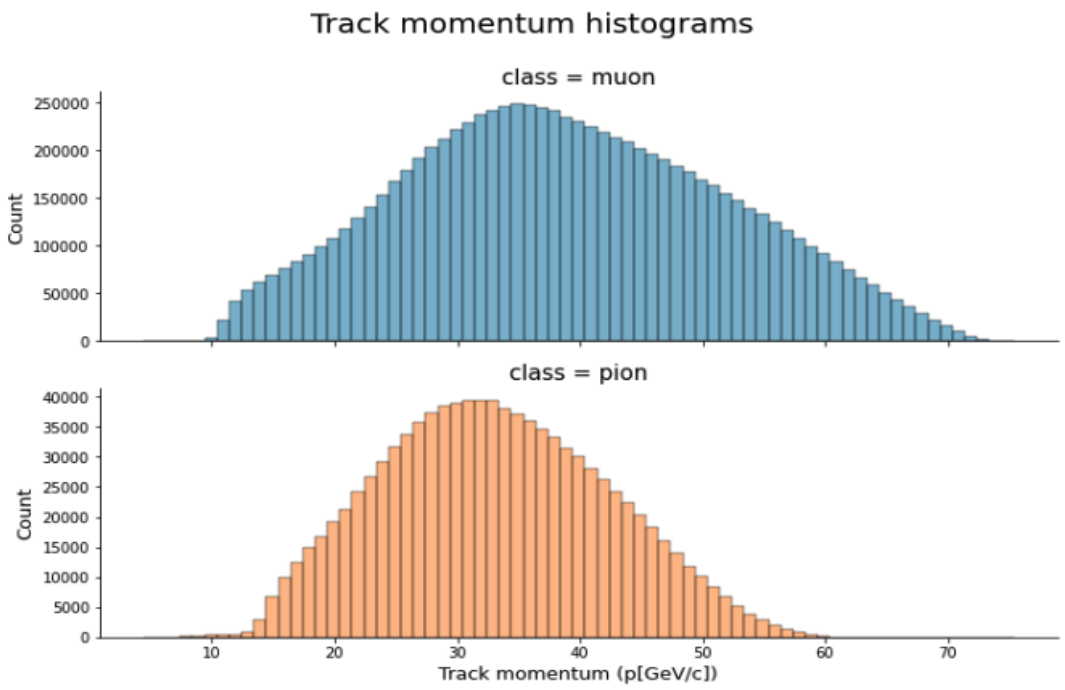

Fig. 3 Distribution of momentum for all samples by class.

The class imbalance will be detrimental to achieving a high pion efficiency as the pion is the minority class. Undersampling the muons to match the number of pions was a feasible solution to address this issue due to the large dataset. However, random sampling of muons examples cannot be used for this data. This is due to the systematic difference in the distribution of momentums between the two particles and is purely an artifact of the experimental setup as seen in Fig. 3. This will bias the output as the objective of this project is to carry out the classification strictly based on differences in the ring size between the particles. The solution was to split the data into three equally sized bins by momentum in the range of 15-45 GeV/c, count the number of pions in each bin, and sample an equal number of muons within that bin. The resulting synthetic dataset contained 2 million examples. There were enough examples to feed into the machine learning models, and momentum as a feature was controlled.

Debiasing w/ respect to ring center bias

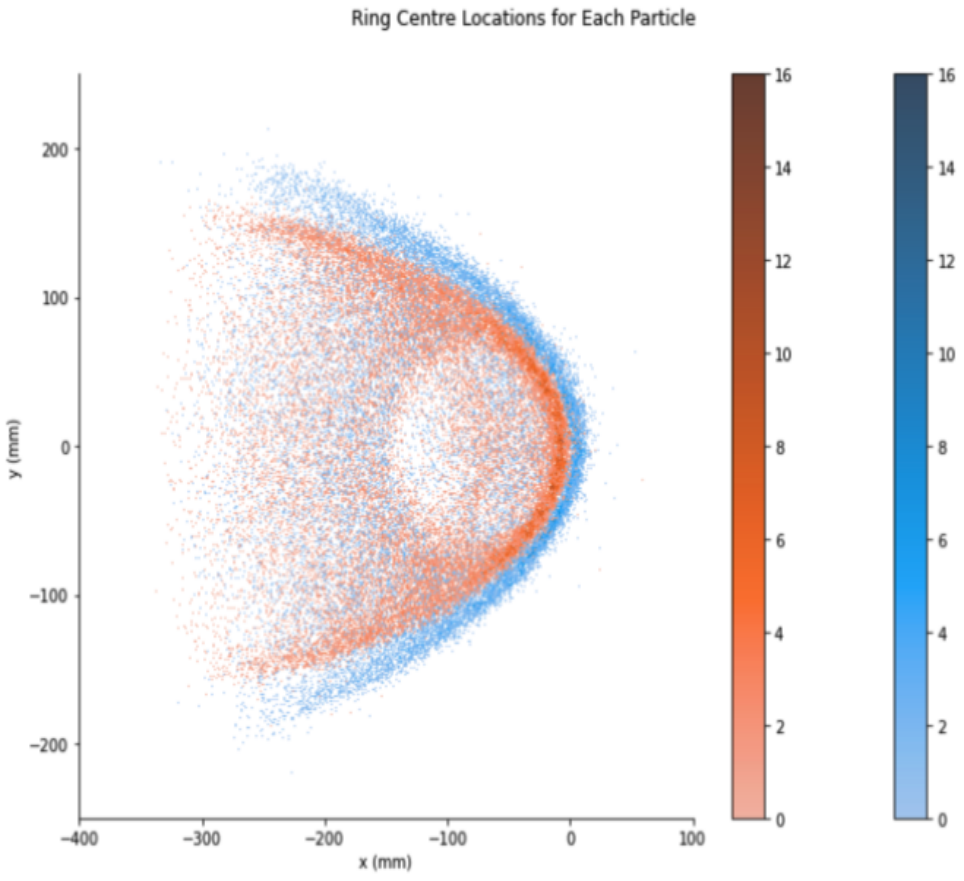

Fig. 4 Distribution of ring center calcualted using the MLE algorithm for all samples by class.

Similarly, Fig. 4 details the difference in the ring centers computed using the MLE between the two classes. As this feature is an abstraction of the scatter of X and Y coordinates, it can be inferred that the bias exists in the raw hits information which will be fed into the deep learning models. These models identify this spatial difference on the standardized PMT grid, and therefore bias the classification. Demeaning each of the hits data using the global X and Y positions of the ring centers, irrespective of class, will remove this bias.

Point cloud generation

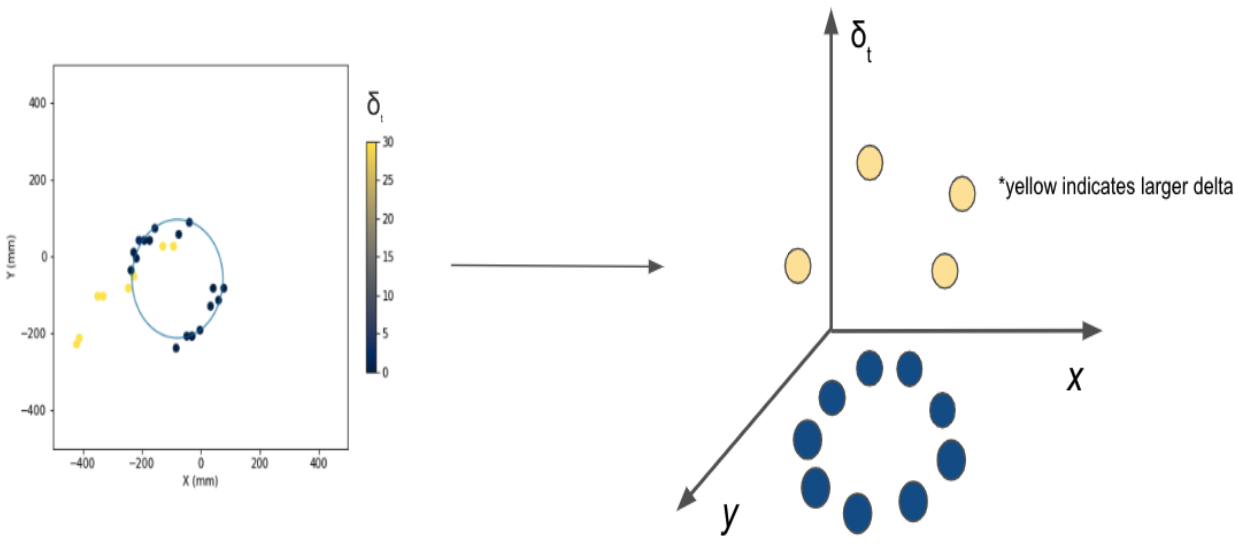

Fig. 5 The generation of the point cloud from the 2D scatter through the addtion of a time dimension.

Naturally, the photon hits information for each event is a system of X and Y coordinates that exists in a 2D plane. As the number of hits for an event usually ranges from 15-30, and the PMT grid is of size 1952, the information is very sparse when treated as a standard image. Hence the photon hits information was converted to a point cloud by adding a third dimension of time: the absolute value of the difference between the photon hit time and the particle travel. This is an eloquent solution as noise hits will have a large value in this new dimension and therefore be separated from the ring produced by the genuine motion of the particle. This is detailed in Fig. 5.

By Nico Van den Hooff, Rakesh Pandey, Mukund Iyer, Shiva Jena

Data Science Methods

Contents

Baseline model: Gradient boosted trees

Why GBT are a good baseline model?

Gradient Boosted Decision Trees (GDBT) use ensemble of decision trees sequentially minimising a loss function and hence, are popular due their efficiency, accuracy and ability to avoid overfitting. Besides, the libraries associated offer flexibility in terms of parameters such as decision tree algorithm, loss function, regularization, GPU related parameters etc. which make them a popular first choice as baseline models.

There are several algorithm based GBDTs available on open source platforms such as Lightgbm, Catboost, Xgboost, Adaboost etc. They are mostly available in form of individual libraries with native implementations and with sklearn in some cases.

Why we used this model

As the number of features were low in our case, the decision tree algorithms were not expected to vary considerably. Therefore, Xgboost (XGBClassifier) with sklearn API was chosen as our baseline GBDT for benchmarking purposes primarily due to its support in form of parameters enabling GPU accelaration for faster training.

Features that were used for GBT

The following features were used for Xgboost:

- ring_radius: radius of the the circle fitted by the MLE algorithm (provided by TRIUMF)
- track_momentum: momentum data as provided along with data (provided by TRIUMF)
- total_hits_filtered: an engineered feature on total number of hits per event/entry, filtered with a fixed value of delta (chod_time - hit_time)

Pros/Cons

Pros:

- simple and intuitive, as the features used are physics informed properties of the particles (classes)
- efficient in terms of low training time Cons:
- does not capture the position data of the hits as it directly captures the MLE radius
- takes a fixed value of delta (chod_time - hit_time) without any scope to learn how to eliminate noise in the data
- low pion efficiency after limiting the muon efficiency

Results

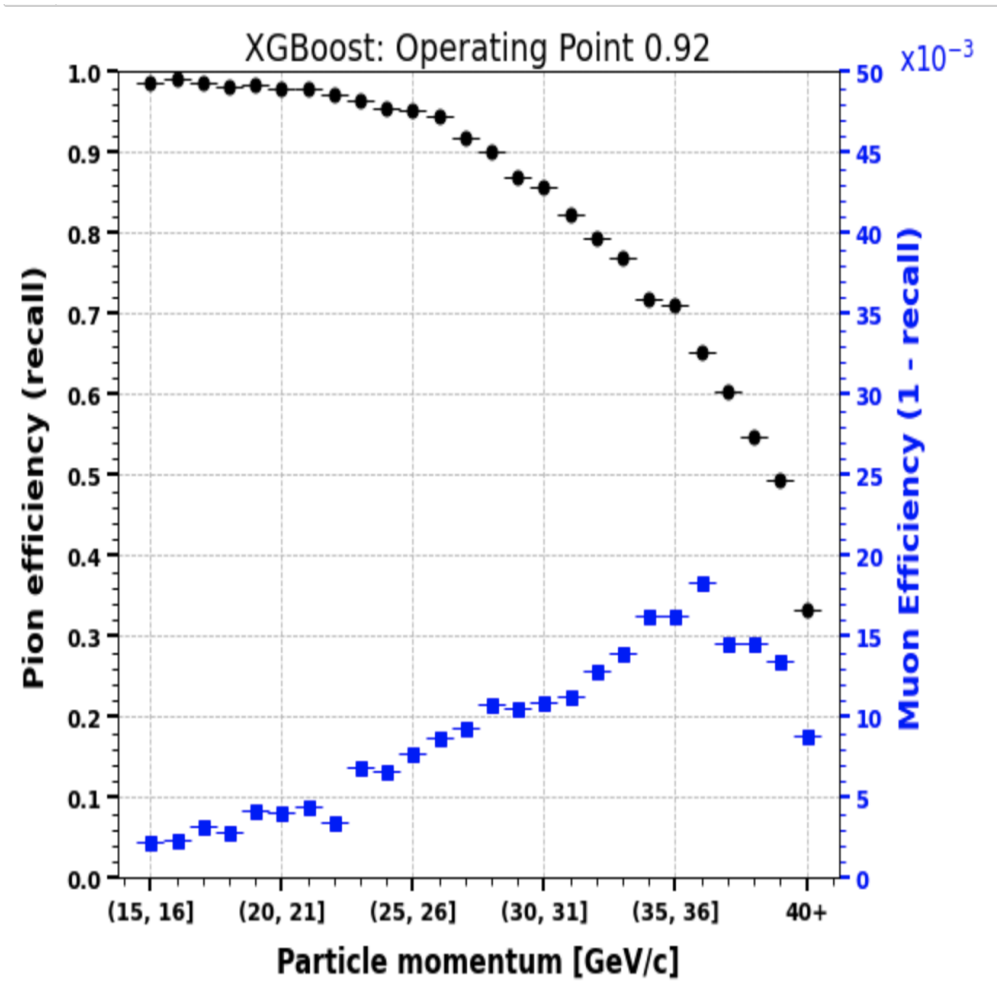

Fig. 6 xgboost results

As observed above, the pion efficiency drops sharply with increase in momentum beyond 35 GeV/C. Besides, muon efficiency is poor at the chosen operating point. Further, different xgboost models were trained and tested on different momentum bins. A Global xgboost model trained over 15-45 GeV/c momentum bin as well as local xgboost models were trained and evaluated. It was observed that the models performed poorly in higher momentum bins as shown below:

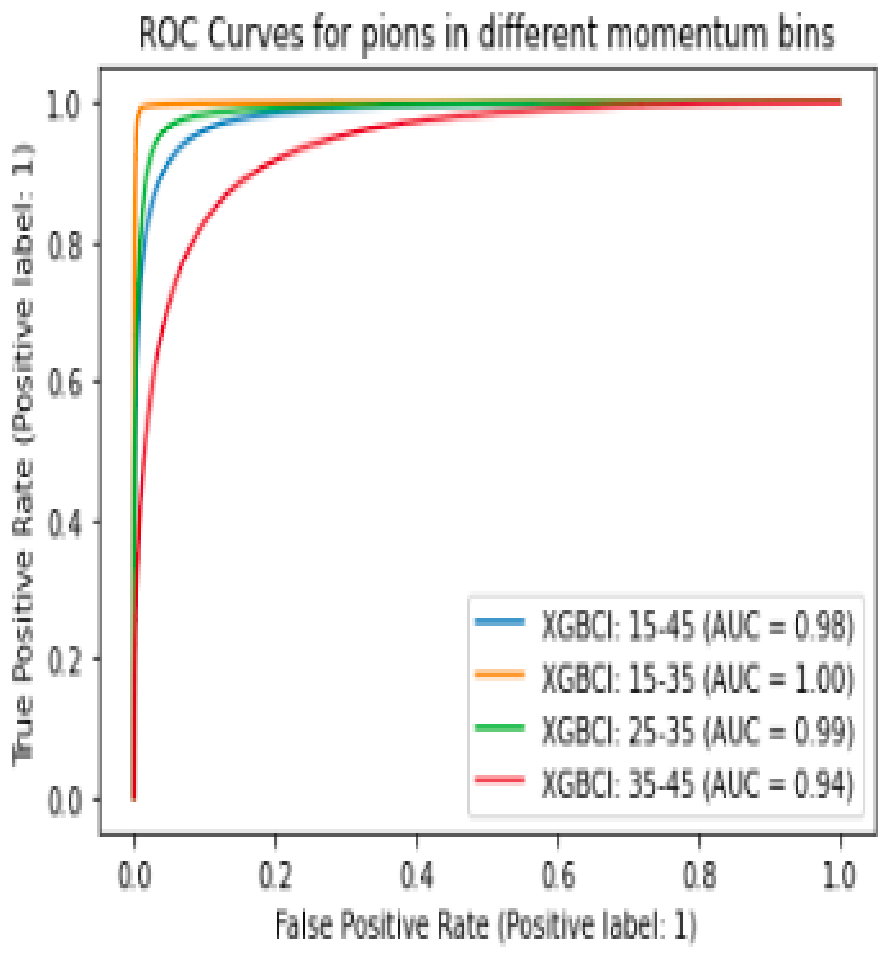

Fig. 7 ROC curves of xgboost models on different momentum bins

The following ROC curves plot establishes that the models were actually leveraging discriminating power of input features and not biased by distributional issues in data.

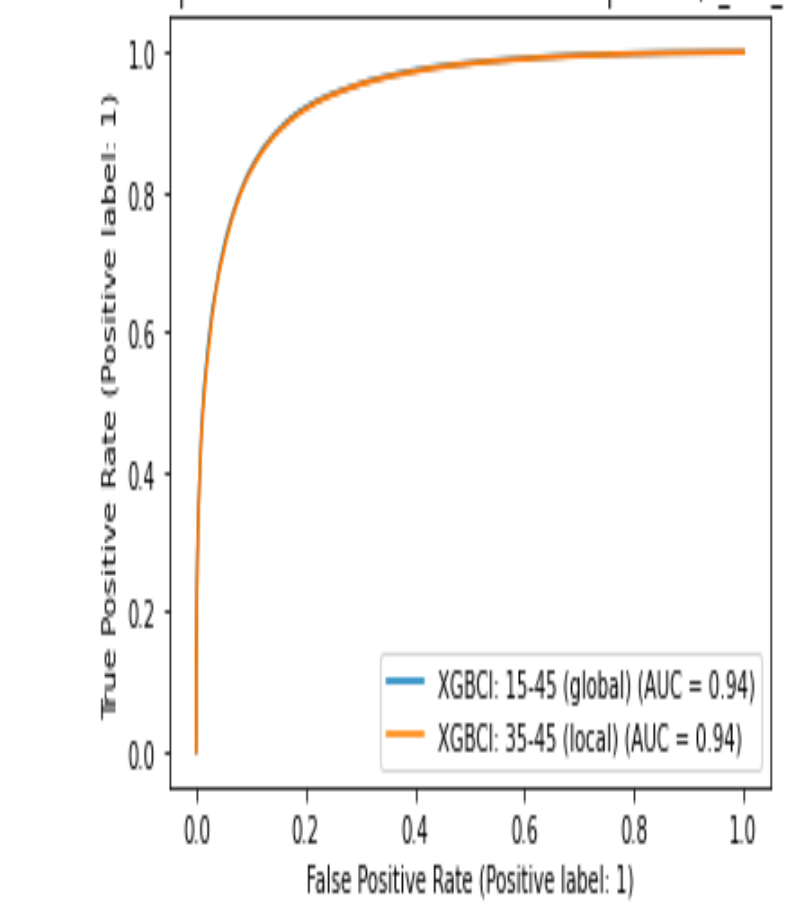

Fig. 8 ROC curves of globally and locally trained xgboost models on 35-45 GeV/c momentum bin

Justification for moving onto deep learning

- Xgboost GBDT model did not use position data of the hits. Instead, it used the already engineered feature - ring_radius from the analytical MLE method leaving no scope for improving the results.
- Therefore, to improve results, more accurate models were required which could extract features directly and more precisely from the hits position data.

Thus, deep learning models were which leverage feature extraction from position data in form of point clouds comprised the further steps in modeling approach beyond baseline GBDT model.

Deep learning

Selecting models

In selecting our deep learning models, we searched for models that were specifically designed to work well with point cloud coordinate data. Based on this research, we identified two models with architectures that appeared to fit our problem scope well. The first model that we identified was called PointNet [Qi *et al.*, 2017], which was developed by researchers at Stanford University. The second model that we identified was called Dynamic Graph CNN [Wang *et al.*, 2019], which was developed by researchers at Massachusetts Institute of Technology, UC Berkely, and Imperial College London. We also note that traditional convolutional neural networks would not work well for our data due to its sparsity.

Tuning models

We tried out two separate feature combinations when tuning both PointNet and the Dynamic Graph CNN models. The first feature combination was a model that took the hits point cloud, momentum, and radius data as input to the neural networks. The second feature combination was a simpler model that did not take the radius as input, but still received the hits point cloud and momentum data.

In terms of hyperparameters, the common hyperparameters that were tuned were the:

- Time delta between the hit time and the CHOD time, for which values of 0.20ns to 0.50ns were used
- Learning rate, for which a constant learning rate and a learning rate scheduler were used
- Number of epochs, for which a maximum of 24 epochs were used

The Dynamic Graph CNN had an additional hyperparameter, which was the value k for the K-nearest neighbours graph that is dynamically generated by the models architecture. We tried values between 8 and 20 nearest neighbors.

PointNet

Model benefits

- PointNet achieves a strong pion efficiency accross all momentum bins, while maintaining similar muon efficiency to prior NA62 performance
- PointNet uses a symmetric function (max pooling) to make it robust to any change in the order of the coordinates that make up the input point cloud data
- PointNet uses a Spatial Transformer Network [Jaderberg *et al.*, 2015] to make it robust to any spatial variability within the input point cloud data

Model shortcomings

- PointNet requires the longest training time of all the models we tested (~24 hours on three GPUs)
- By it's design, PointNet is not able to capture local information within the point cloud

Best performing model

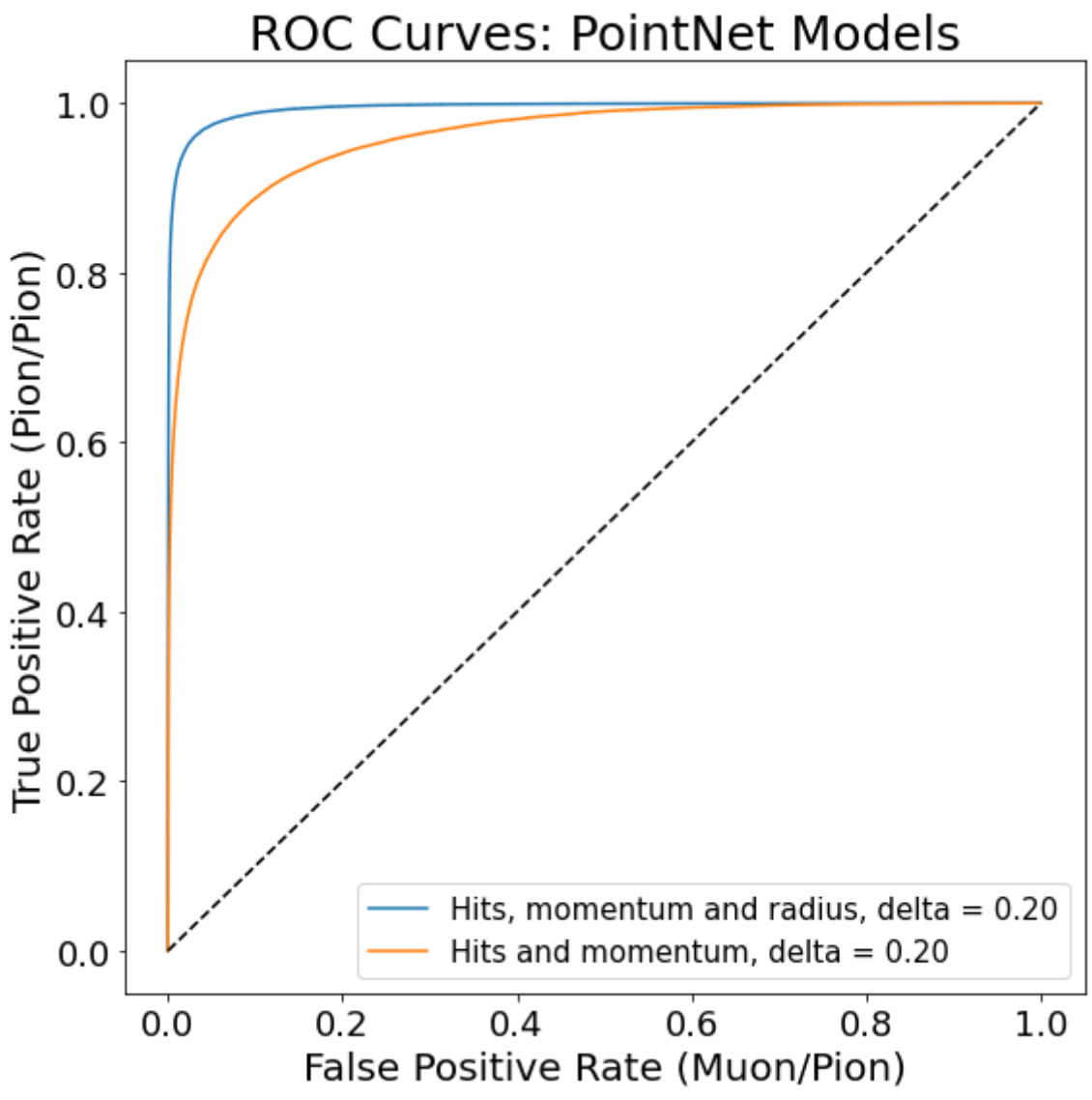

Fig. 9 Pointnet ROC Curves

As can be seen in the receiver operating characteristic curves ("ROC") in Fig. 9, our best PointNet model consisted of the following:

- All features: hits point cloud, momentum, and radius
- Time delta: 0.20ns

- Learning rate: Linear annealing scheduler. Initial learning rate of 0.0003 that increases to 0.003 in the first half of training, before decreasing back to the original learning rate in the second half of training
- Epochs: 16 epochs of training

We selected an operating point of 0.93 on the ROC curve, as this allowed us to achieve a strong pion efficiency while maintaining good muon efficiency relative to the NA62 results.

Dynamic Graph CNN

Model benefits

- Dynamic Graph CNN is designed to capture local information within the point cloud data
- Dynamic Graph CNN training time is almost two times faster than PointNet

Model shortcomings

- Although Dynamic Graph CNN can maintain a similar pion efficiency to PointNet, it stuggles with muon efficiency.

Best performing model

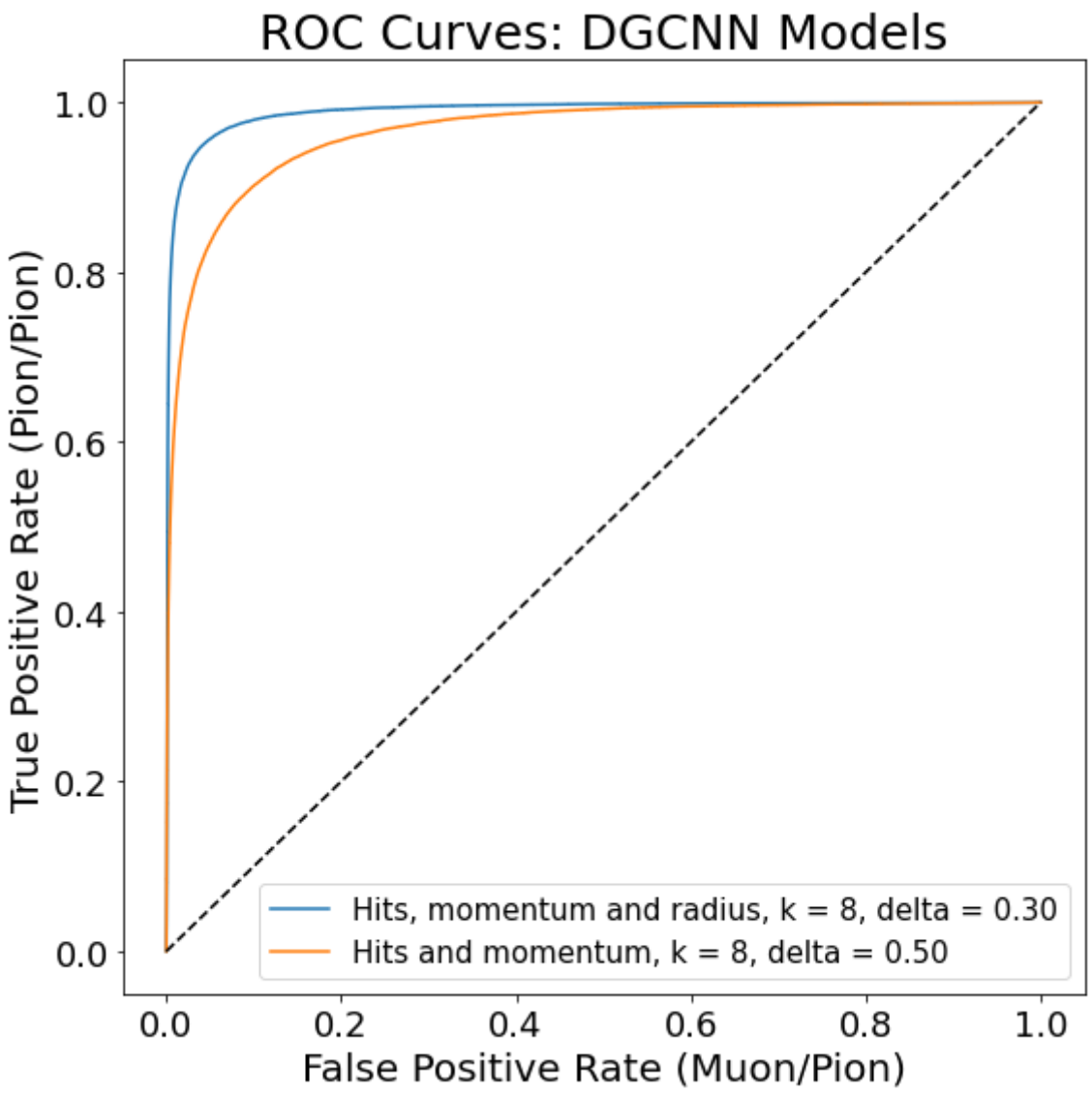

Fig. 10 Dynamic Graph CNN ROC Curves

As can be seen in the receiver operating characteristic curves ("ROC") in Fig. 10, our best Dynamic Graph CNN model consisted of the following:

- All features: hits point cloud, momentum, and radius
- Time delta: 0.30ns
- Learning rate: Linear annealing scheduler. Initial learning rate of 0.0003 that increases to 0.003 in the first half of training, before decreasing back to the original learning rate in the second half of training.
- Epochs: 8 epochs of training

We selected an operating point of 0.96 on the ROC curve, as this allowed us to achieve a strong pion efficiency. We were unable to select an operating point that achieved both a good pion efficiency and a good muon efficiency.

Overall model results

Selecting the overall best model

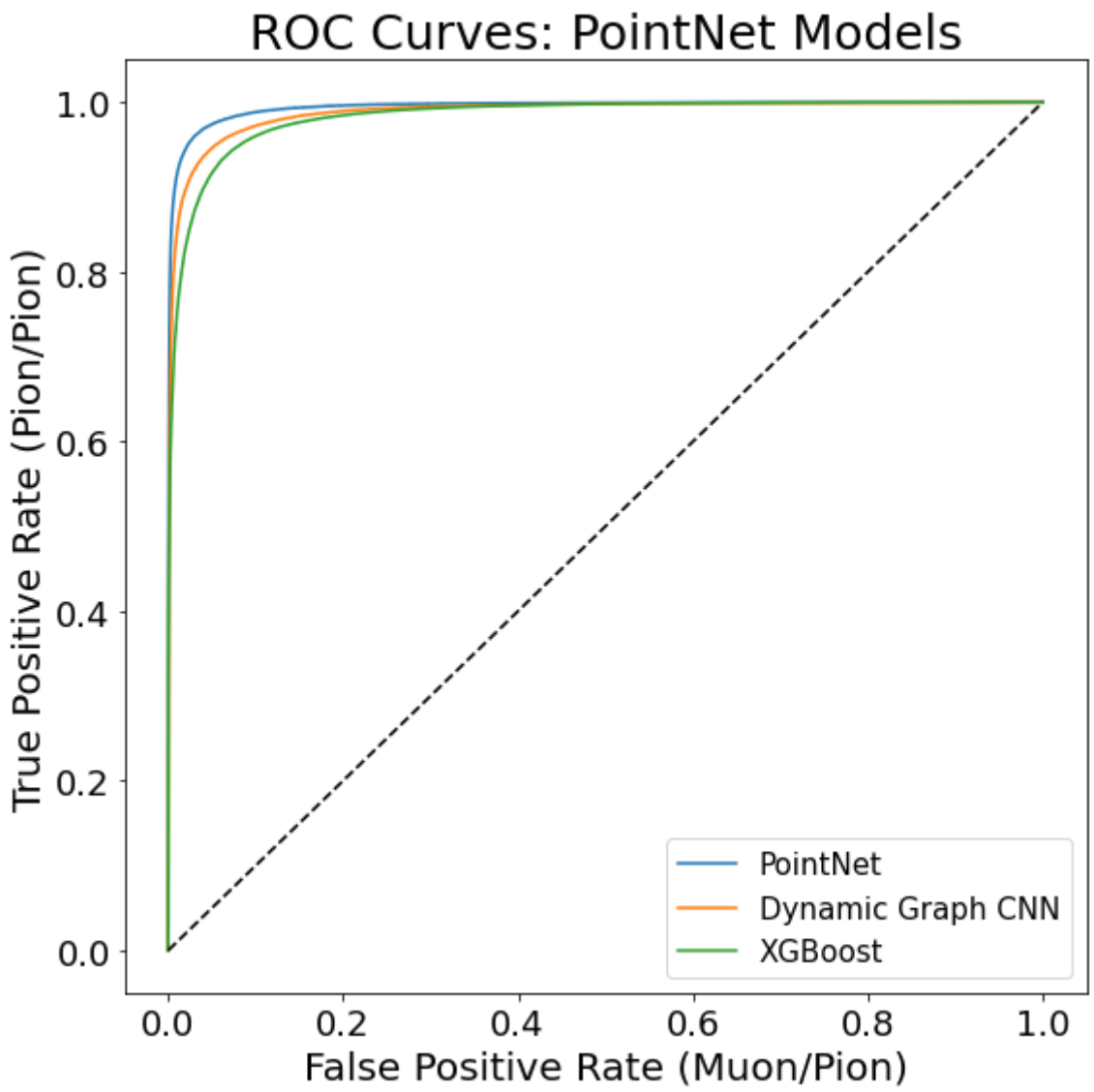

Fig. 11 ROC Curves: All models

In selecting our overall best model, we used the ROC curve. As can be seen in the Fig. 11, our overall best model was PointNet, following by Dynamic Graph CNN. Both deep learning models were able to surpass our baseline XGBoost model.

Model efficiencies

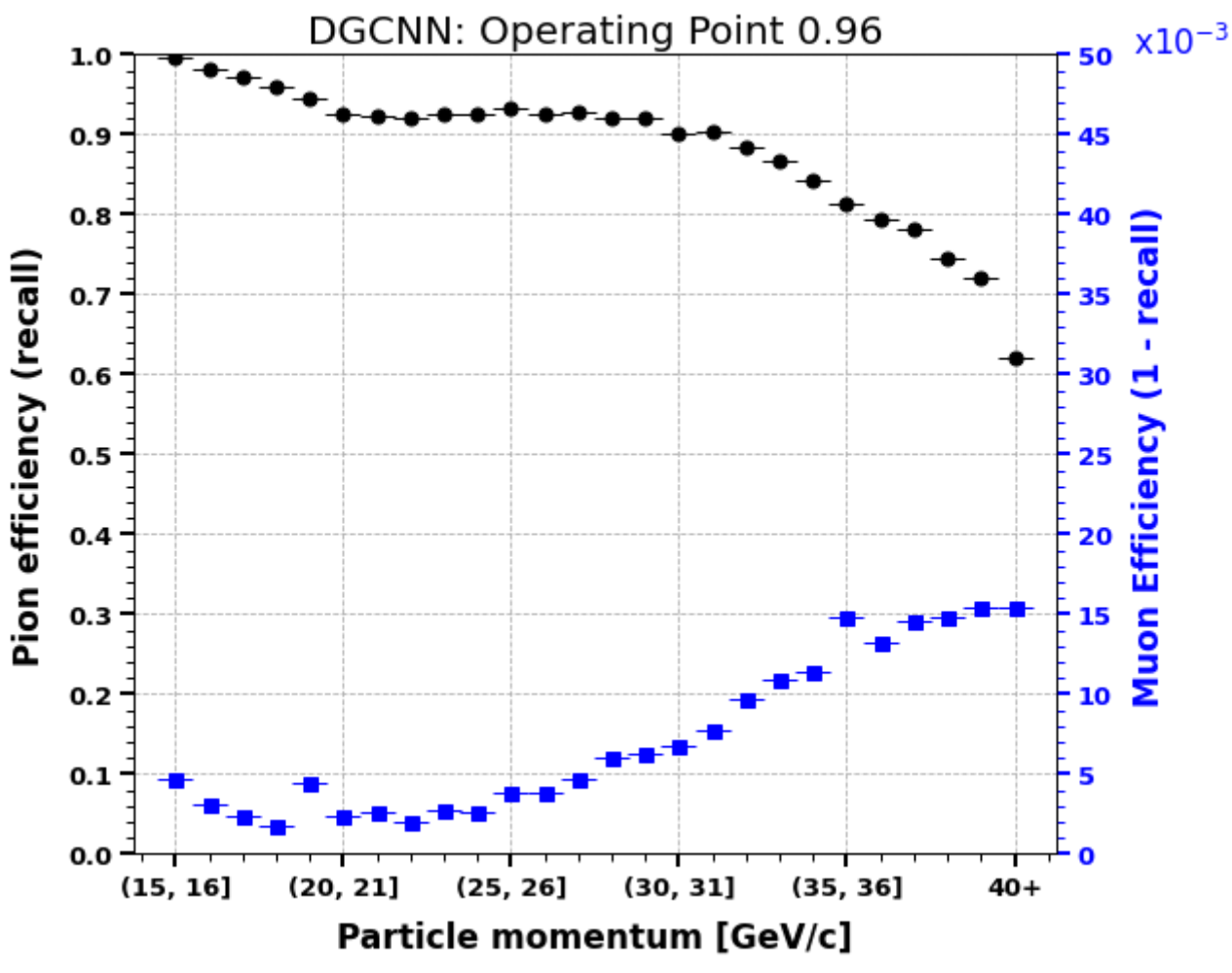

Fig. 12 Dynamic Graph CNN Model Efficiencies

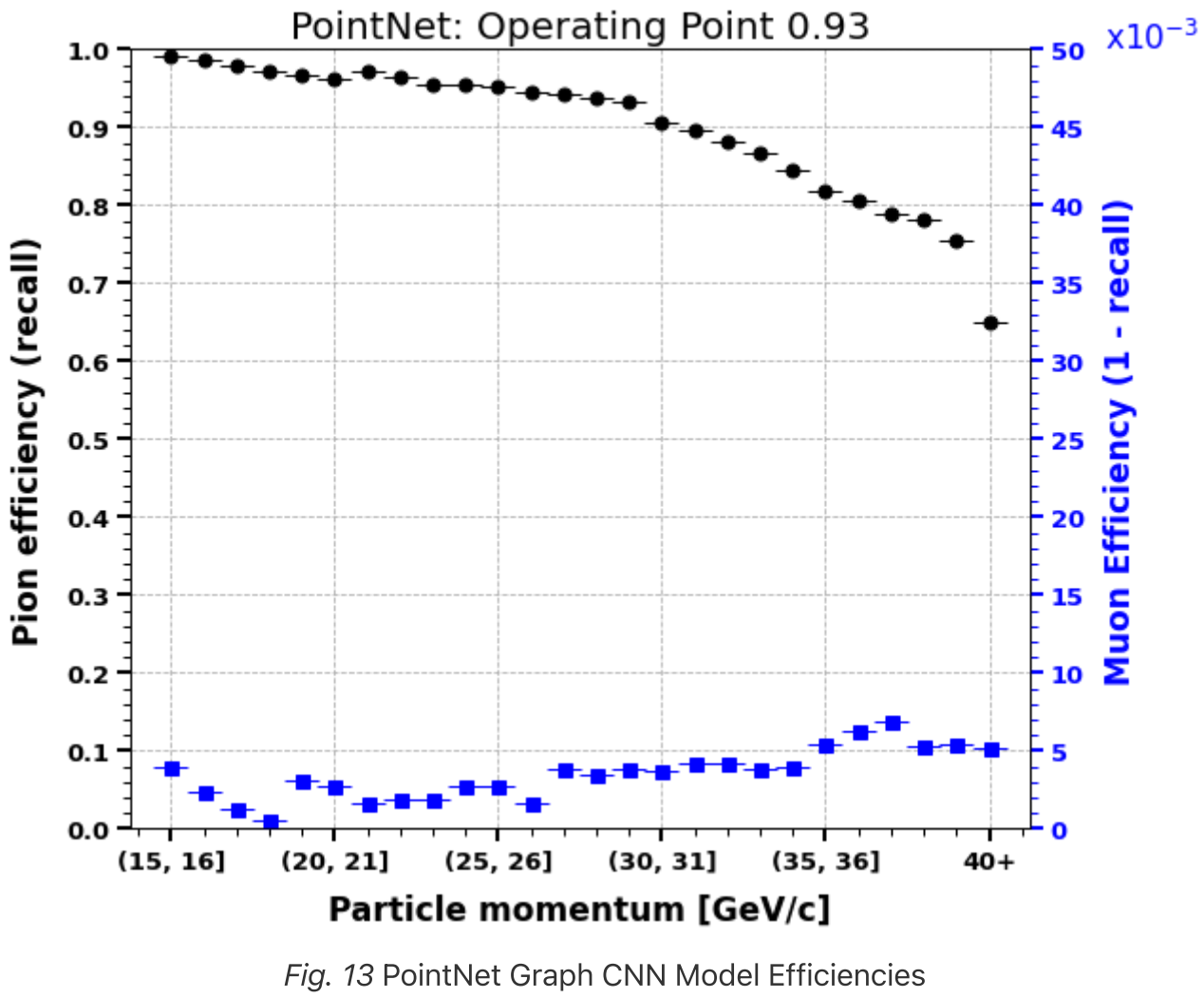

Fig. 13 PointNet Graph CNN Model Efficiencies

In analyzing our final model performance in terms of pion and muon efficiency in Fig. 12 and Fig. 13, we note the following:

- Both the PointNet and Dynamic Graph CNN models were able to surpass the prior NA62 pion efficiency performance across all momentum bins.
- The Dynamic Graph CNN model was unable to achieve a similar muon efficiency to NA62 in any momentum bin.
- The PointNet model was able to surpass NA62 muon efficiency in momentum bins > 34 GeV/c.

By Nico Van den Hooff, Rakesh Pandey, Mukund Iyer, Shiva Jena

Data Product

Contents

Product overview

Our data product is a machine learning pipeline that takes in the experiment data in HDF5 format, pre-processes them, prepares training data, trains classifier models on the training data and evaluates model results on test data through modular scripts. An overview of the product along with key library dependencies is shown in the image below:

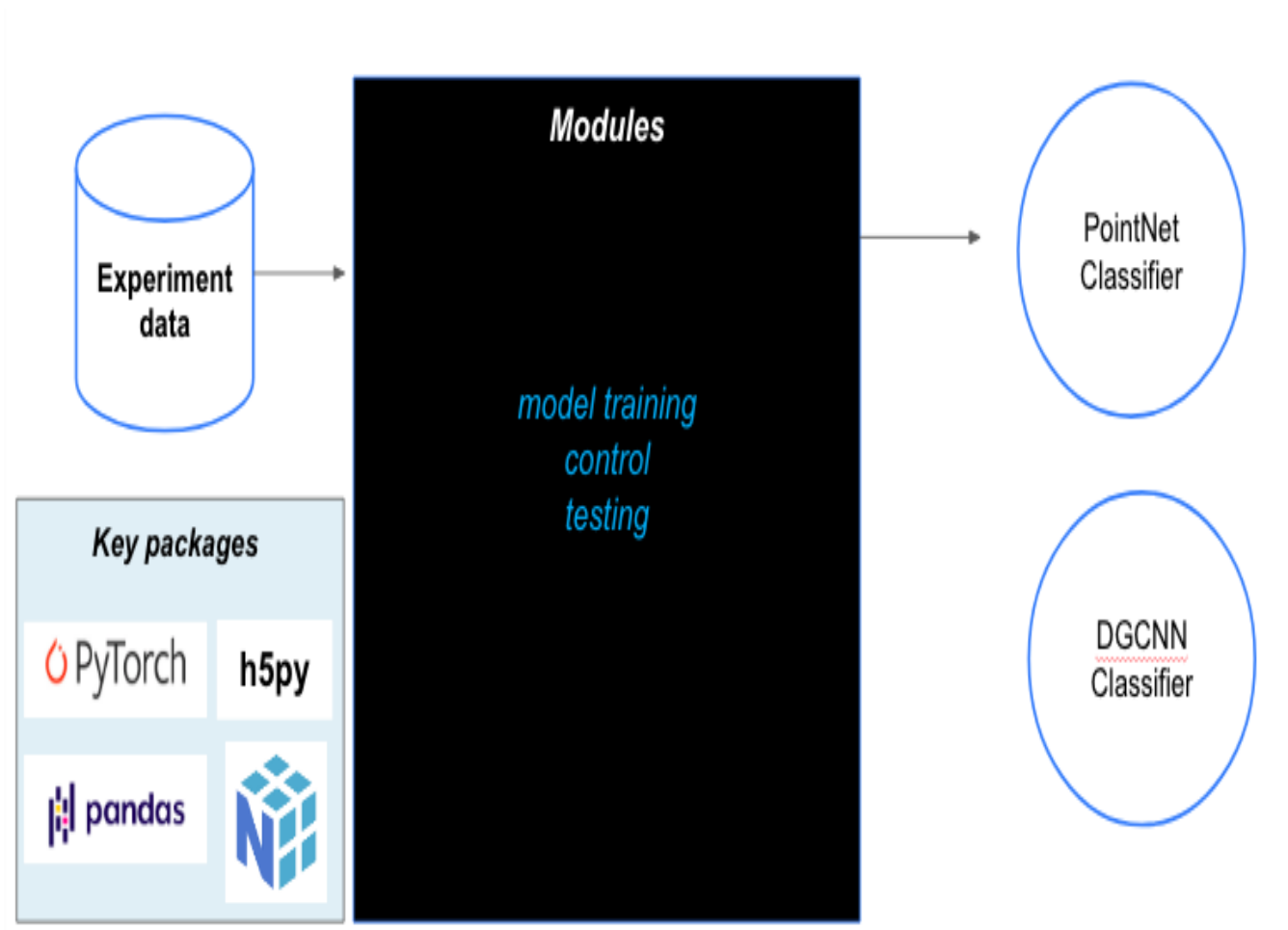

Fig. 14 Product Overview

Modules and usage of the product

The product has one script to run all the models and generate results. All the codes are built into modules. The modules and a snapshot of the associated scripts or sub-modules are shown in the following sections.

`config`

The product has all the controls built in the `config` module. It has the control parameters wherein the user can control model parameters/ hyperparameters such as train/test/saved model paths, dataloader configurations such as batch size, number of workers, etc., number of epochs for training, device id, and many more.

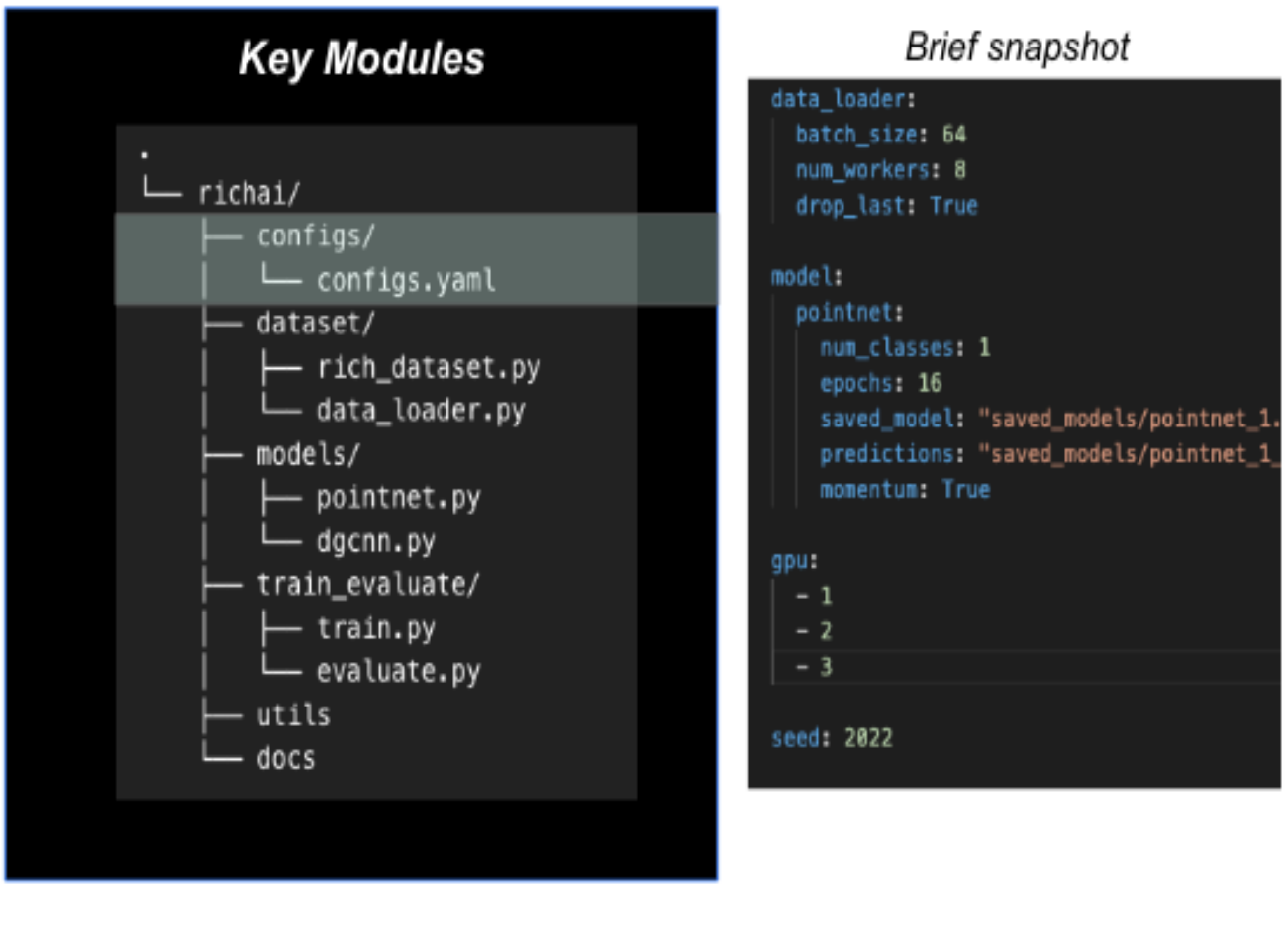

Fig. 15 config module

`dataset`

`dataset` has two sub-modules:

- `rich_dataset.py` processes raw HDF5 format and extracts events, hits and position data.
- `dataloader.py` loads data in batches as feed into the models

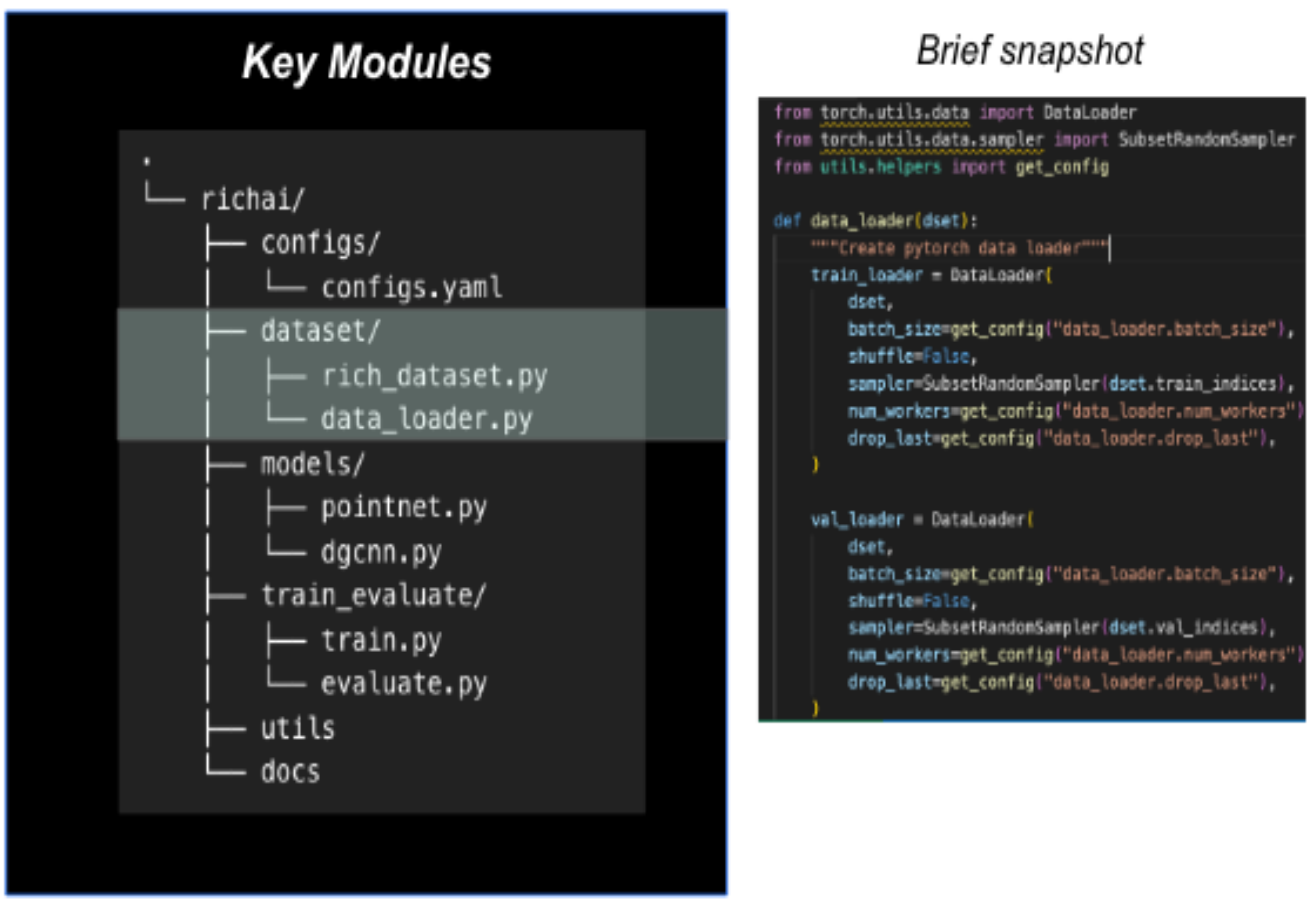

Fig. 16 dataset module

`models`

`models` has the architecture built in for the two deep learning models PointNet and DGCNN.

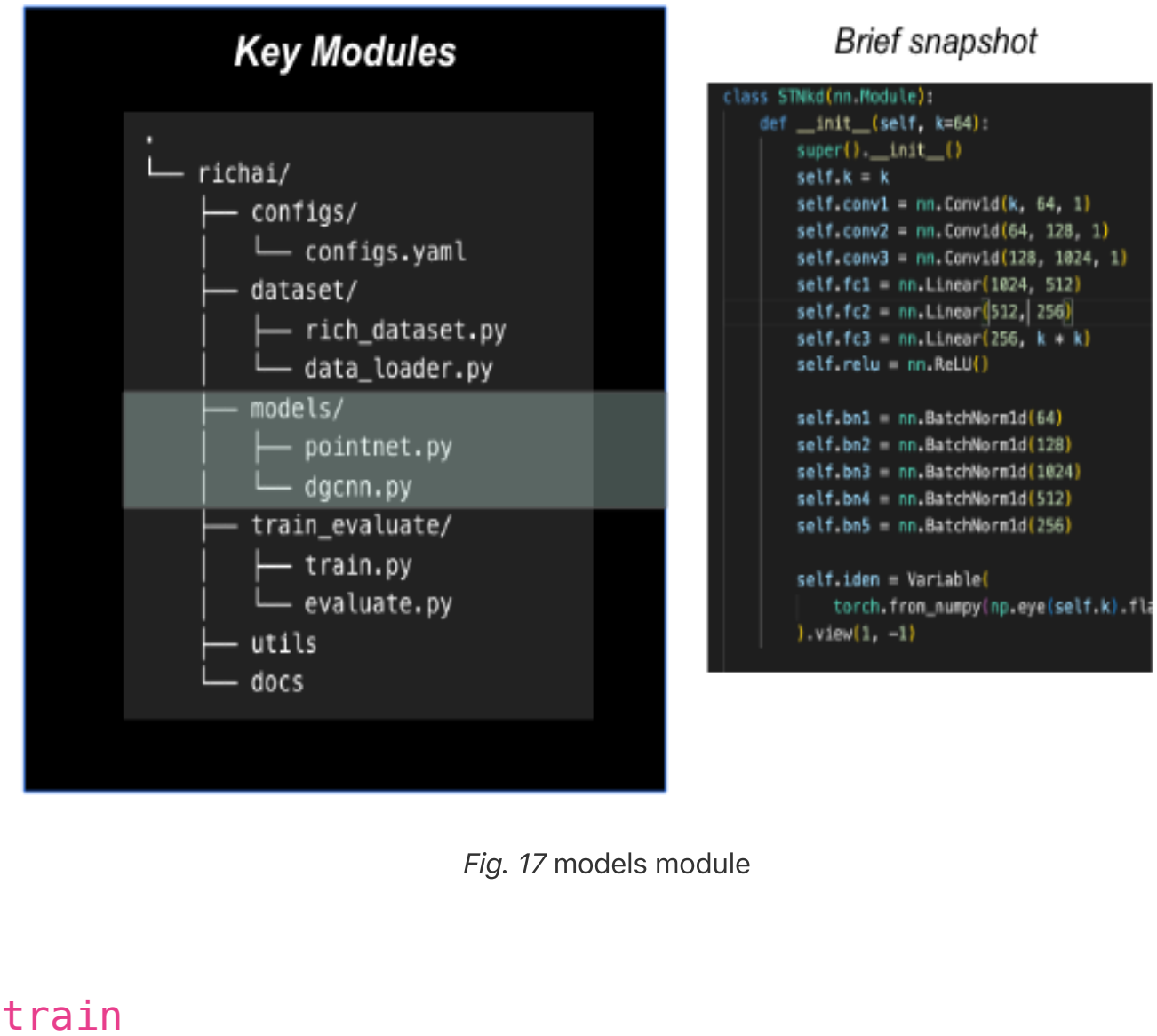

Fig. 17 models module

`train`

`train` has modules for training and evaluating the model results.

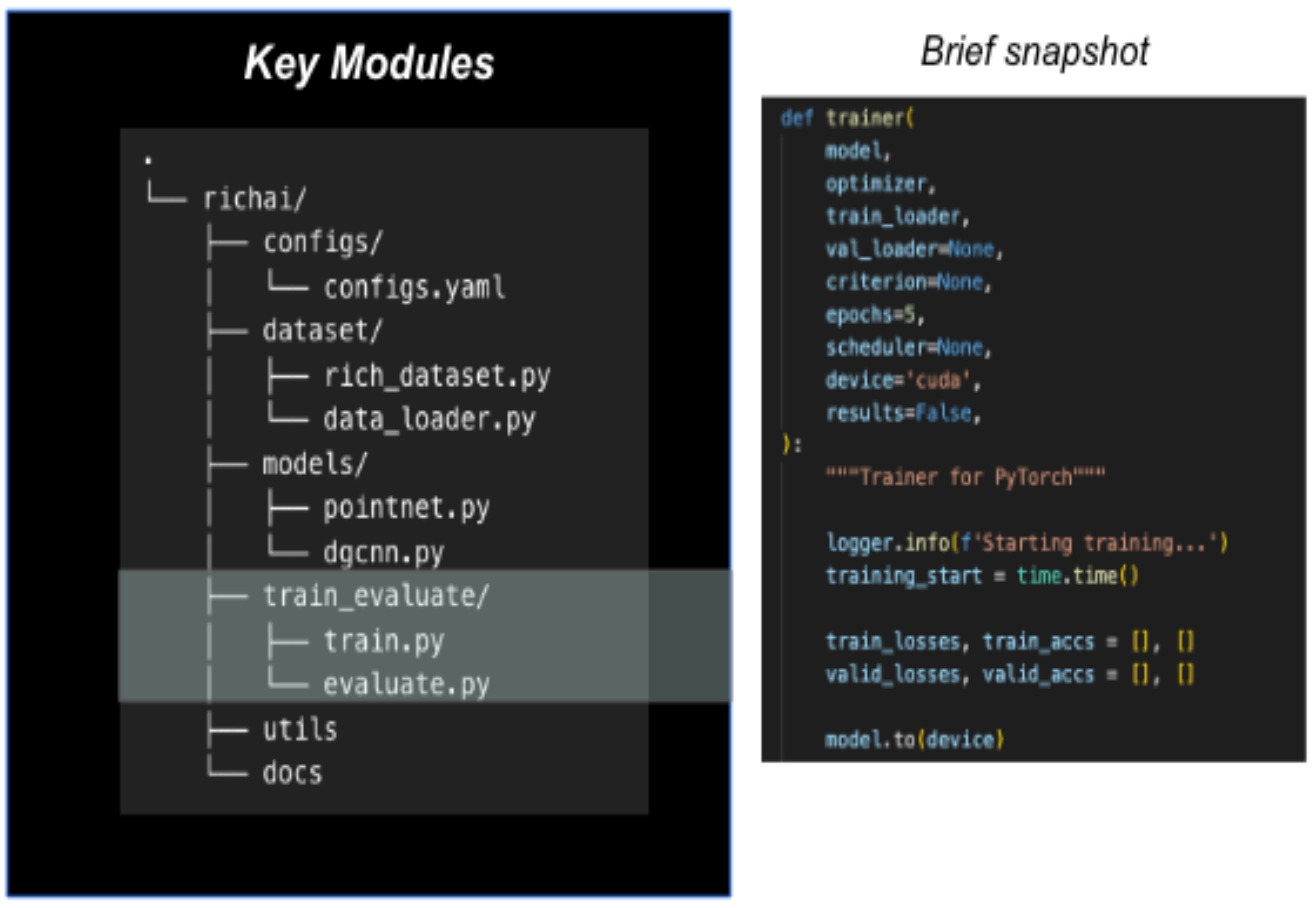

Fig. 18 train module

Others

`utils` has helper functions to generate some plots for usage by user. `docs` has proposal and final reports.

Pros and Cons

Pros:

- PointNet and DGCNN have high pion efficiency at higher operating points
- Easy to configure using the config module
- Modular codes allow flexibility in tweaking the architecture in future, if needed.

Cons:

- Models require long training time and high computing power with multiple GPUs due to their inherent architecture.

Hence, further optimization may be tried by hyperparameter tuning and/or experimenting with the architectures of the two models. Due to time constraints and limitations of producing a working product, further improvements could not be built in. Although there may not be huge scope for improvement in these model results in terms of pion and muon efficiences, hyperparameter tuning or experimenting with the model architectures may be tried as future course of action in improving the product performance.

By Nico Van den Hooff, Rakesh Pandey, Mukund Iyer, Shiva Jena

Conclusions and recommendations

The RICH AI project aimed to predict the particle in concern as pion or muon with given input features extracted from experiment data. This pertained to a binary classification problem. The goal was to create better and more accurate models using deep learning based classification models in comparison to the existing analytical methods.

The data product designed in this capstone project by building two deep learning based classifiers addresses this problem by accurate extracting features from the data. The product successfully addresses the classification problem and achieves the goals with much more accurate results than the analytical method in terms of the evaluation metrics. The models may perform better with more training data while due consideration to the momentum distributions.

The above models were an attempt to implement point clouds based deep learning models at a level pertaining mostly to the vanilla architectures. Going forward, more customizations and hyperparameter tunings as well as complex models such as PointNet++ could be tried to experiment further and see if the results improve.

By Nico Van den Hooff, Rakesh Pandey, Mukund Iyer, Shiva Jena

References

ABB+20 G Anzivino, M Barbanera, A Bizzeti, F Brizioli, F Bucci, A Cassese, P Cenci, R Ciaranfi, V Duk, J Engelfried, and others. Light detection system and time resolution of the na62 rich. *Journal of Instrumentation*, 15(10):P10025, 2020.

GAM+17 E. Cortina Gil, E. Martin Albarran, E. Minucci, G. Nüssle, S. Padolski, P. Petrov, N. Szilasi, B. Velghe, G. Georgiev, V. Kozhuharov, L. Litov, T. Husek, K. Kampf, M. Zamkovsky, R. Aliberti, K. H. Geib, G. Khoriauli, K. Kleinknecht, J. Kunze, D. Lomidze, R. Marchevski, L. Peruzzo, M. Vormstein, R. Wanke, A. Winhart, M. Bolognesi, V. Carassiti, S. Chiozzi, A. Cotta Ramusino, A. Gianoli, R. Malaguti, P. Dalpiaz, M. Fiorini, E. Gamberini, I. Neri, A. Norton, F. Petrucci, M. Statera, H. Wahl, F. Bucci, R. Ciaranfi, M. Lenti, F. Maletta, R. Volpe, A. Bizzeti, A. Cassese, E. Iacopini, A. Antonelli, E. Capitolo, C. Capoccia, A. Cecchetti, G. Corradi, V. Fascianelli, F. Gonnella, G. Lamanna, R. Lenci, G. Mannocchi, S. Martellotti, M. Moulson, C. Paglia, M. Raggi, V. Russo, M. Santoni, T. Spadaro, D. Tagnani, S. Valeri, T. Vassilieva, F. Cassese, L. Roscilli, F. Ambrosino, T. Capussela, D. Di Filippo, P. Massarotti, M. Mirra, M. Napolitano, G. Saracino, M. Barbanera, P. Cenci, B. Checcucci, V. Duk, L. Farnesini, E. Gersabeck, M. Lupi, A. Papi, M. Pepe, M. Piccini, G. Scolieri, D. Aisa, G. Anzivino, M. Bizzarri, C. Campeggi, E. Imbergamo, A. Piluso, C. Santoni, L. Berretta, S. Bianucci, A. Burato, C. Cerri, R. Fantechi, S. Galeotti, G. Magazzu\textquotesingle , M. Minuti, A. Orsini, G. Petragnani, L. Pontisso, F. Raffaelli, F. Spinella, G. Collazuol, I. Mannelli, C. Avanzini, F. Costantini, L. Di Lella, N. Doble, M. Giorgi, S. Giudici, E. Pedreschi, R. Piandani, G. Pierazzini, J. Pinzino, M. Sozzi, L. Zaccarelli, A. Biagioni, E. Leonardi, A. Lonardo, P. Valente, P. Vicini, G. D\textquotesingle Agostini, R. Ammendola, V. Bonaiuto, N. De Simone, L. Federici, A. Fucci, G. Paoluzzi, A. Salamon, G. Salina, F. Sargeni, C. Biino, G. Dellacasa, S. Garbolino, F. Marchetto, S. Martoiu, G. Mazza, A. Rivetti, R. Arcidiacono, B. Bloch-Devaux, M. Boretto, L. Iacobuzio, E. Menichetti, D. Soldi, J. Engelfried, N. Estrada-Tristan, A. M. Bragadireanu, O. E. Hutanu, N. Azorskiy, V. Elsha, T. Enik, V. Falaleev, L. Glonti, Y. Gusakov, S. Kakurin, V. Kekelidze, S. Kilchakovskaya, E. Kislov, A. Kolesnikov, D. Madigozhin, M. Misheva, S. Movchan, I. Polenkevich, Y. Potrebenikov, V. Samsonov, S. Shkarovskiy, S. Sotnikov, L. Tarasova, M. Zaytseva, A. Zinchenko, V. Bolotov, S. Fedotov, E. Gushin, A. Khotjantsev, A. Khudyakov, A. Kleimenova, Yu. Kudenko, A. Shaikhiev, A. Gorin, S. Kholodenko, V. Kurshetsov, V. Obraztsov, A. Ostankov, V. Rykalin, V. Semenov, V. Sugonyaev, O. Yushchenko, L. Bician, T. Blazek, V. Cerny, M. Koval, R. Lietava, G. Aglieri Rinella, J. Arroyo Garcia, S. Balev, M. Battistin, J. Bendotti, F. Bergsma, S. Bonacini, F. Butin, A. Ceccucci, P. Chiggiato, H. Danielsson, J. Degrange, N. Dixon, B. Döbrich, P. Farthouat, L. Gatignon, P. Golonka, S. Girod, A. Goncalves Martins De Oliveira, R. Guida, F. Hahn, E. Harrouch, M. Hatch, P. Jarron, O. Jamet, B. Jenninger, J. Kaplon, A. Kluge, G. Lehmann-Miotto, P. Lichard, G. Maire, A. Mapelli, J. Morant, M. Morel, J. Noël, M. Noy, V. Palladino, A. Pardons, F. Perez-Gomez, L. Perktold, M. Perrin-Terrin, P. Petagna, K. Poltorak, P. Riedler, G. Romagnoli, G. Ruggiero, T. Rutter, J. Rouet, V. Ryjov, A. Saputi, T. Schneider, G. Stefanini, C. Theis, S. Tiuraniemi, F. Vareia Rodriguez, S. Venditti, M. Vergain, H. Vincke, P. Wertelaers, M. B. Brunetti, S. Edwards, E. Goudzovski, B. Hallgren, M. Krivda, C. Lazzeroni, N. Lurkin, D. Munday, F. Newson, C. Parkinson, S. Pyatt, A. Romano, X. Serghi, A. Sergi, R. Staley, A. Sturgess, H. Heath, R. Page, B. Angelucci, D. Britton, D. Protopopescu, I. Skillicorn, P. Cooke, J. B. Dainton, J. R. Fry, L. Fulton, D. Hutchcroft, E. Jones, T. Jones, K. Massri, E. Maurice, K. McCormick, P. Sutcliffe, B. Wrona, A. Conovaloff, P. Cooper, D. Coward, P. Rubin, and R. Winston. The beam and detector of the NA62 experiment at CERN. *Journal of Instrumentation*, 12(05):P05025–P05025, may 2017. URL: https://doi.org/10.1088%2F1748-0221%2F12%2F05%2Fp05025, doi:10.1088/1748-0221/12/05/p05025.

JSZ+15 Max Jaderberg, Karen Simonyan, Andrew Zisserman, and others. Spatial transformer networks. *Advances in neural information processing systems*, 2015.

QSMG17 Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 652–660. 2017.

WSL+19 Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019.

By Nico Van den Hooff, Rakesh Pandey, Mukund Iyer, Shiva Jena