CaloRICH AI

Discrimination of Muon and Pion Decay in NA62

Crystal Geng, Daniel Merigo, Kelvin Wong, Peng Zhang

June 28, 2023

Contents

1 Executive Summary

The project focuses on the NA62 project conducted by CERN. The NA62 requires a high level of identification performance to distinguish between muons and pions during a rare meson decay in order to verify the Standard Model of Physics. This identification is performed using the Cherenkov Radiation phenomenon, specifically the photon ring radius constructed from the array of photodetectors in the RICH detector.

From our results, the XGBoost seemed to be the best model to perform this ring-fitting. The results, unfortunately, did not yield a better performance than the current state-of-the-art algorithm employed by the NA62 team, which uses a maximum likelihood estimation (MLE) model.

2 Introduction

The CaloRICH AI project is proposed by TRIUMF, Canada's national particle accelerator center, to improve the Ring-Imaging CHerenkov detector (RICH detector) particle identification performance. Our work this year is an extension of the RICH AI project [den Hooff N et al., 2022], attempted by an MDS capstone team of similar goals last year.

The project dataset comes from the NA62 experiment at CERN, the European Organization for Nuclear Research, which involves a particle physics experiment designed to study rare decays of charged kaons. The capstone project aims to build a model to accurately classify a particle as a pion produced in a decay event. We aim to train models using the data from a subset of the 2021A NA62 experiments, which contains around 2.4 million decay events labeled by calorimeter [Gil et al., 2017].

Based on the research and advice by the partner, we chose XGBoost as the baseline Machine Learning model, and applied a modified version of PointNet as well as a simple Multi-Layer Perceptron neural network.

2.1 Challenges

Our challenge is to improve the capability to determine the specific ring radius of the Cherenkov Radiation cone in the RICH detector, by avoiding the use of the momentum detection and performing a regression analysis instead of a classification based on the "hits" data only.

3 Data Science Methods

3.1 Data

The dataset we work with is provided by TRIUMF. The data structure can be read from the Raw Data section on the project website.

3.2 Regressor

3.2.1 XGBoost Regressor

3.2.1.1 Why This Model? For our baseline model, we have attempted to train XGBRegressor, RandomForestRegressor, LightGBM and CatBoostRegressor. Out of all the models we have trained, we selected XGBRegressor (Chen and Guestrin 2016) as our baseline model because of its impressive performance and fast execution compared to other baseline models we experimented with. We employed XGBRegressor to train our data set, which exclusively consisted of muons. In addition to the previously mentioned engineered features, we included `total_in_time_hits` as one of the features in our model. The target variable `y_train` used for training the model corresponds to the calculated ring radius (`ring_radius_calc`), which is the theoretical ring radius determined based on the track momentum and mass of the particles.

3.2.1.2 Performance Using the default hyperparameters of `n_estimators = 100` and `max_depth = 6`, the XGBRegressor achieved an average training R^2 score of 0.96 and an average test R^2 score of 0.94 during 5-fold cross-validation. We attempted hyperparameter optimization using `RandomizedSearch`, but it did not yield significant improvements in performance. Additionally, the best parameter value found by `RandomizedSearch` was `n_estimators = 500`, which required substantial computational cost. Therefore, we decided to stick with the default hyperparameters.

3.2.1.3 Residual Analysis Although the XGBRegressor showed excellent performance, we noticed a bias near the higher end of the radius spectrum. The scatter plot of residuals shown in Fig. 1, which represents the difference between the calculated ring radius and the predicted ring radius, reveals a slight bias towards the higher end of the calculated ring radius. More data points are concentrated above the 0 line, indicating that the model tends to underestimate the radius for higher values. In light of this residuals pattern, further in-depth analyses were conducted to investigate the underlying cause of this phenomenon.

To examine the correlation between the predicted ring radius and the theoretical ring radius, we computed the difference between the theoretical ring radius and its overall maximum for each data point. This is illustrated by the red line in Fig. 2. If a data point lies above the red line, it suggests that the predicted ring radius is smaller than the theoretical ring radius. Conversely, if a data point lies below the red line, it indicates that the predicted ring radius is larger than the theoretical ring radius. Upon examining the more detailed plot on the right side of Fig. 2, it becomes apparent that the data points below the red line are

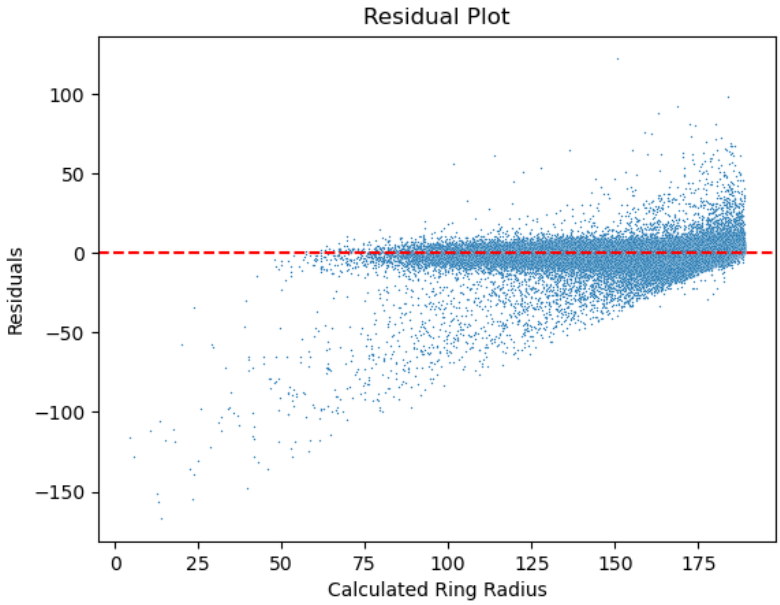

Figure 1: Residual plot residuals vs. calculated ring radius for XGBRegressor

considerably scarce. Additionally, the data points appear to be "pushed" upwards by the presence of the red line. This observation suggests that the predictions are constrained by the upper limit of the theoretical ring radius. As a result, the predictions are unable to exceed the maximum value of the theoretical ring radius.

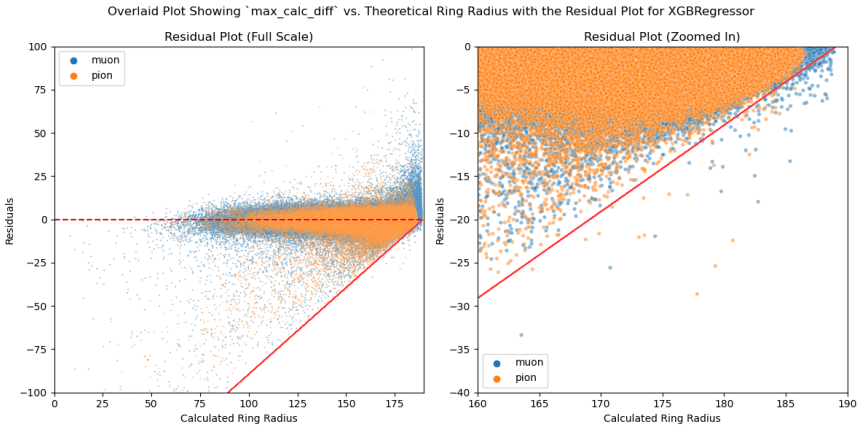

Figure 2: Overlaid residual plot showing the difference between the theoretical ring radius and its overall maximum for each data point (shown as red line)

3.2.1.4 Conclusion The XGBRegressor model demonstrates favourable performance, as evidenced by high training and test R^2 scores, with the majority of residuals concentrated around 0. However, there exists an underestimation issue specifically associated with higher radius values. This underestimation is believed to be limited by the upper limit of the theoretical ring radius, a hypothesis supported by our residual analysis.

In attempts to address this problem, we explored various techniques such as feature selection, L1 and L2 regularization, and hyperparameter optimization. Unfortunately, these approaches did not effectively

mitigate the bias. Notably, we observed the same residual bias when employing other tree-based models such as RandomForestRegressor, LightGBMRegressor, and CatBoostRegressor.

Given the persistent bias, we became intrigued by the possibility of utilizing more sophisticated models like neural networks, hoping they may alleviate the issue.

3.3 Neural Networks

3.3.1 MLP (Multilayer Perceptron)

3.3.1.1 Why This Model? A multilayer perceptron (MLP) is a classic fully connected feedforward neural network. The model is expected to capture more complex linear and non-linear relationships between input features (regressors) and the target. MLP is easy to implement in the PyTorch ("PyTorch Documentation" n.d.) framework. It provides us with the extensive flexibility to custom design deep learning structures, define particular loss functions, or add more functionalities to our modeling.

3.3.1.2 Features and Model Structure The MLP model has taken all 18 engineered features as the inputs, the theoretical ring radius as the target, to predict the ring radius for each event.

After a number of experiments, we have selected a MLP model with 4 hidden layers. The hidden size is gradually decreased from 1024 to 12, and the ReLU activation function has been used in each hidden layer.

3.3.1.3 Training Process and Results To avoid possible bias from two particles, the model has been trained on randomly split 75% of muon events, then tested on the rest of muon events and all pion events. Based on the PyTorch framework, we have developed the training and validation data loaders, the custom defined trainer, used standard MSELoss function and Adam optimizer (Kingma and Ba 2017) to train the MLP model.

After the training completed, we tested the regression performance on the test data. Between the predicted ring radius and the theoretical values, the R^2 score was 0.934 and the MAE (Mean Absolute Error) was 1.92. Compared to the result of the XGBRegressor approach (R^2 was 0.94), the accuracy of the MLP model was slightly lower.

We have also performed residual analysis for the MLP model. As shown in Fig. 5, the similar patterns were observed compared to the residual plot from the XGBRegressor. The result confirms the bias analysis in the previous section.

3.3.1.4 Conclusion From the result comparison between XGBRegressor and the MLP model, we can see the deep learning architecture (MLP) did not improve the regression performance.

One of the advantages of deep learning methods is to rely on the complex networks to perform feature engineering and feature extraction. In our case, we feed the MLP model with explicitly engineered features rather than the raw data (coordinates of hits for each event). It is possible that the method we have used in this section did not reach the full potential of the deep learning.

Another advantage is that we can do quantile regression in this sort of architecture. The results for that is published in the report website.

3.3.2 PointNet

3.3.2.1 Architecture Developed in Stanford, PointNet is a simple yet effective architecture that directly consumes 3-dimensional point clouds and can perform diverse operations such as classification, part segmentation and even semantic parsing (Qi et al. 2017). In order to apply this architecture to the regression problem we had to do some modifications, including: - Adding a third dimension to the hit locations using Gaussian noise in the range (0, 0.5) - Substituting the output layer with a linear fully connected layer - Changing the output size to 1

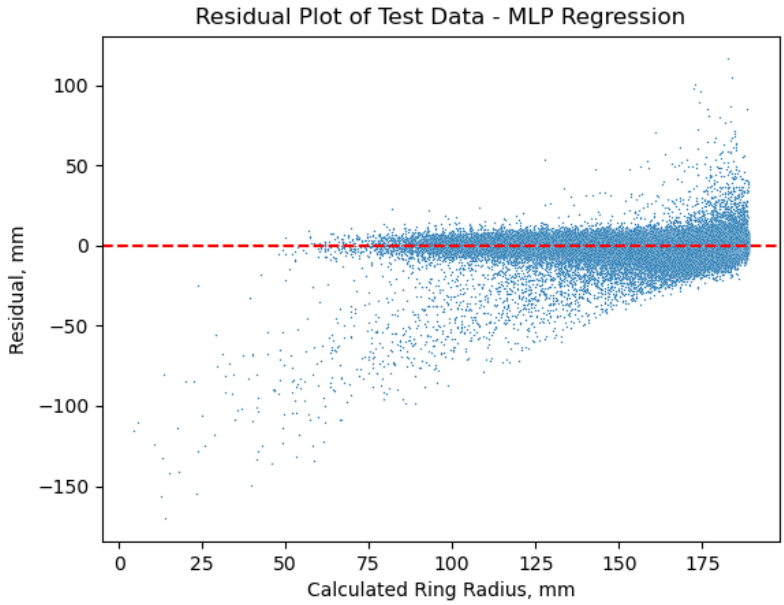

Figure 3: Residual plot residuals vs. calculated ring radius for MLP model

We developed 2 distinct architectures, based on the base PointNet architecture, in PyTorch to compare and test simultaneously. We used the MSELoss Loss function and Adam Optimizer. Each architecture also utilized ReLU as the activation function

3.3.2.2 Benefits and Limitations PointNet is an architecture developed to perform classifications, translating the architecture to a regression problem meant discarding the semantic segmentation portion and adding a 3rd dimension at first. This added complexity to the data preparation and in the end could not capture local information or immediately determine the particle for a given event.

Training time was also a limitation as, even with pre-transformed parquet files, the training time averaged 15 minutes per epoch on a high-end customer GPU.

3.3.2.3 Results As shown in Fig 7, the performance of PointNet is deficient, not being able to capture the trends in the model. The predicted results have a different behaviour than the ground truth, shown here as a black line. These results, combined with the long training time, motivated us to decide on XGBoost as the best modelling technique for this regression problem.

3.4 Best Performing Model

After evaluating the performance of three models: XGBRegressor, MLP, and PointNet, it appears that XGBRegressor currently outperforms the others. According to the results, both XGBRegressor and MLP demonstrate similar best scores. However, the performance of the PointNet architecture has not been satisfactory thus far, and it also requires an exceptionally long training time.

Model	Training Time	Best Test R^2
XGBRegressor	~ 3 minutes	0.94
MLP	~ 8 minutes	0.94
PointNet	~ 23.5 hours	0.20

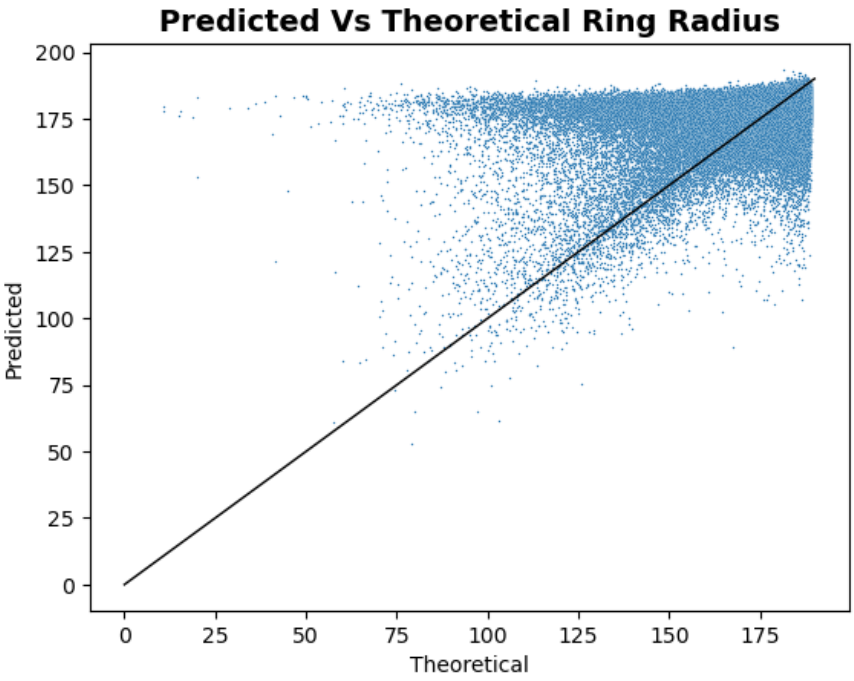

Figure 4: Residual plot predicted vs. calculated ring radius for PointNet

When comparing the performance of XGBRegressor and MLP, it is difficult to determine the better performing model solely based on their performance scores. However, taking into account the difference in training time, with XGBRegressor being more than 2 times faster, and considering the additional complexity associated with MLP modeling, it is reasonable to conclude that XGBRegressor is a more efficient and simpler choice in terms of both time consumption and computational power. Overall, we determine XGBRegressor to be the best performing model with our training data.

4 Data Product and Results

The data product of the *CaloRICH AI* project is a concise, documented and reproducible machine learning pipeline. The pipeline includes the data preprocessing, model training, model evaluation and analyses required to achieve the results presented in this report in a manner such that the TRIUMF team can verify the findings and potentially incorporate the pipeline into their own ML pipelines.

4.1 Structure of the Data Product

Our pipepline includes three components: the data preprocessing, the model training and the model performance anlayses.

4.1.1 Data Preprocessing

The data preprocessing pipeline generates parquet files from the original HDF5 file format that prepare the data for training and testing. This pipeline also calculates and adds the **engineered features** we created for the XGBoost and MLP models.

4.1.2 Model Training and Analyses

We also provide a simple set of commands based on a makefile that trains each of the selected models and performs the analyses of the model's outcome for the following three models:

1. XGBoost Regressor
2. Multilayer Perceptron neural network
3. PointNet

Details on how to execute the pipeline and run the makefiles can be found in the README.md of the project's GitHub repository.

4.1.3 Additional Notebooks

There are a number of supplementary notebooks in the notebooks folder, separated into experiments and analyses. The files are documented by the corresponding README.md in each of the subfolders.

5 Conclusions and Recommendations

5.1 Conclusions

In this capstone project we have applied machine learning regression methods to predict the ring radius from variable number of "hits" (excited photodetectors in the RICH detector) for each event of decay. Three models, XGBRegressor, MLP, and PointNet, have been adopted to perform the ring fitting. Both XGBRegressor and MLP were using engineered features generated from hit positions as inputs, while PointNet was directly trained on the raw data of hit coordinates. The result of regression evaluation metrics shown XGBRegressor had the best performance compared to the other two deep learning approaches. Considering the simplicity of model structure and implementation, timing, and future maintenance, we recommend XGBRegressor as the best option to perform the regression task.

Although the result of PointNet did not yield an expected performance, we have identified the probable reason, a vanishing gradient, a thus have devised solutions to this problem. Due to the time consuming process of training a new model with the necessary modifications we are keeping these conclusions for the moment and will update if a solution is found and has proven to yield better results than XGBoost.

5.2 Recommendations

For the future works, we would suggest the following areas:

1. In addition to the three regular regression models, we have also started a quantile regression based on the MLP architecture. Due to time constraints, more research and analysis could be continued to explore the possibility of quantile regression to facilitate the differentiation of pions and muons in different track momentum ranges.
2. Since the ultimate goal of the NA62 project is to distinguish pions and muons (Gil et al. 2017), the resolution of fitted ring radius for the two particles in each track momentum range would be important. We need to consider both accuracy and precision of the predictions in order to separate two particles effectively in the future studies.
3. We also suggest exploring the use of a Convolutional Neural Network (CNN) in future experiments as they have proven reliability in image pattern recognition. If the RICH photodetector is treated as a pixel array there could be the opportunity to obtain a promising result.

6 References

Chen, Tianqi, and Carlos Guestrin. 2016. "XGBoost." In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. https://doi.org/10.1145/2939672.2939785.

Gil, E. Cortina, E. Martin Albarran, E. Minucci, G. Nüssle, S. Padolski, P. Petrov, N. Szilasi, et al. 2017. "The Beam and Detector of the NA62 Experiment at CERN." *Journal of Instrumentation* 12 (05): P05025–25. https://doi.org/10.1088/1748-0221/12/05/p05025.

Kingma, Diederik P., and Jimmy Ba. 2017. "Adam: A Method for Stochastic Optimization." https://arxiv.org/abs/1412.6980.

"PyTorch Documentation." n.d. The Linux Foundation. Accessed June 24, 2023. https://pytorch.org/docs/stable/index.html#.

Qi, Charles R, Hao Su, Kaichun Mo, and Leonidas J Guibas. 2017. "Pointnet: Deep Learning on Point Sets for 3d Classification and Segmentation." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 652–60.