Fall 2021

CS 550 - Advanced Operating Systems

- PROJECT REPORT -

Exploration of machine learning algorithms applied to the categorization of proton collisions in particle accelerators

Pablo Lostao - A20474488

Jose Ignacio Pozuelo - A20497402

Exploration of machine learning algorithms applied to the categorization of proton collisions in particle accelerators

Pablo Lostao
Illinois Institute of Technology
Illinois, United States
plostao@hawk.iit.edu

Jose Ignacio Pozuelo
Illinois Institute of Technology
Illinois, United States
jpozuelosaizdebustam@hawk.iit.edu

I. Introduction

Machine learning is a form of problem solving that is gaining more and more ground in every conceivable field, from song recommendations to landing rockets on Mars. It is therefore not surprising that solutions are beginning to be sought to facilitate scientific research using machine learning models, deep learning, neural networks...

One of the fields in which this type of technique is being applied is particle physics, more specifically in particle accelerators such as the one located at CERN, Geneva [1].

A particle accelerator is a device that uses electromagnetic fields to accelerate charged particles to high speeds, and thus make them collide with other particles. Specifically, in the LHC (*Large Hadron Collider)*, beams of protons collide at a rate of 1 billion collisions per second [2]. In the places where these collisions occur, detectors are placed that use electronic hardware to select potentially interesting proton collisions for further analysis.

These detectors, which constitute a distributed system, use software that chooses whether or not to select a collision in the required time [5], which is a mere microsecond.

What has been explained above works well so far, but with the new technologies available and the need to go deeper into the subatomic world, new approaches must be considered to obtain better results.

The main reason why new solutions have to be considered is because in 2027 [1] the LHC will upgrade its current state (one billion proton-proton collisions per second) in such a way that it will increase the number of collisions per second [4] by a factor of 5 to 7, ultimately resulting in a total amount of accumulated data an order of magnitude higher than what is possible with the current collider.

Due to this increase in collisions coupled with the high number of tasks being performed and the limited number of processors, machine learning solutions have to be explored to minimize latency and maximize the accuracy of the tasks to be performed [3].

II. Project Contributions

Contributions to the project have been as follows:

Pablo Lostao has been in charge of the development of the code necessary to obtain the final results as well as the writing of the results part of this report.

Jose Ignacio Pozuelo has been in charge of the initial development of the code for the previous analysis, as well as the writing of the more theoretical part of the report.

The documentation, work management and preparation of the presentation has been a collaborative effort.

III. Theoretical Basis

In order to have a better understanding of the project and its real applications, we will first explain a series of theoretical concepts in a simple manner.

A. Large Hadron Collinder (LHC)

The *Large Hadron Collider* (LHC) is the world's largest and most powerful particle accelerator. The LHC consists of a 27 km ring of superconducting magnets with a number of accelerating structures to boost the energy of the particles along the way [2].

Inside the accelerator, two high-energy particle beams travel at close to the speed of light before they are made to collide, the beams travel in opposite directions in separate beam pipes. The particles are so tiny that the task of making them collide is akin to firing two needles 10 km apart with such precision that they meet halfway.

All the controls for the accelerator, its services and technical infrastructure are housed under one roof at the CERN Control Centre. From here, the beams inside the LHC are made to collide at four locations around the accelerator ring, corresponding to the positions of four particle detectors – ATLAS, CMS, ALICE and LHCb.

When two protons are made to collide in the spots where these detectors are located, they disintegrate into different particles. After the disintegration, a distributed system

composed by a high number of sensors extracts the maximum possible parameters to be able to identify the particles that emerged from the collisions.

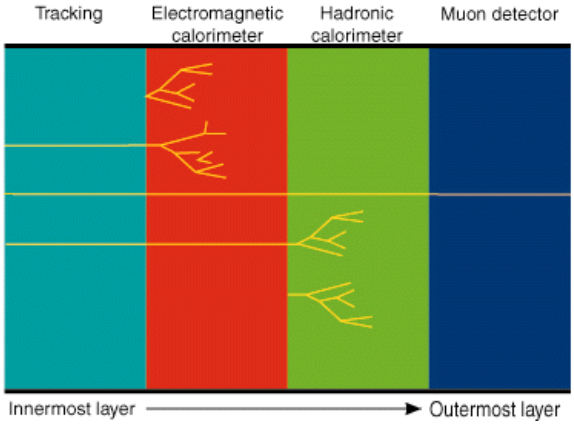
The process of identifying the exact particles that emerged from a collision is very expensive and taking into account that in the next few years the LHC will increase the number of collisions per second by a factor of 5 to 7, new solutions must be explored to manage them correctly. This is where the idea of using machine learning algorithms to identify the particles emerged from the collisions comes from.

The ultimate mission of the particle accelerator is to analyze these newly generated particles in the hope of finding something out of the ordinary that will open the door to "new physics".

B. How particle detectors works?

Particles produced in collisions normally travel in straight lines, but in the presence of a magnetic field their paths become curved. Electromagnets around particle detectors generate magnetic fields to exploit this effect. Physicists can calculate the momentum of a particle, a clue to its identity, from the curvature of its path: particles with high momentum travel in almost straight lines, whereas those with very low momentum move forward in tight spirals inside the detector.

Modern particle detectors consist of layers of subdetectors, each designed to look for particular properties, or specific types of particles.



Tracking devices reveal the path of a particle; *calorimeters* stop, absorb and measure a particle's energy; and *particle-identification detectors* use a range of techniques to pin down a particle's identity [10].

IV. Project Description

Once the basic operation of a particle accelerator has been explained, we will proceed with the description of the project to be carried out.

For this project we are going to apply machine learning algorithms to try to predict the particles that arise from the collision of 2 protons. With collisions expected to increase by several orders of magnitude in the next few years, it is necessary to use such techniques to try to minimize latency and maximize accuracy.

In order to make the task easier let's assume an ideal scenario in which 6 types of particles can come out of a collision: Electron, Kaon, Muon, Pion and Proton, additionally we will consider the Ghost particle used for erroneous detections in the track system.

To carry out the project we will use a dataset from the LHCb detector published by CERN itself. With this data we will train the model so that when it receives certain parameters from the detectors it will be able to know with high precision which particle it is.

A. LHCb – Dataset (training.csv)

As mentioned in the theoretical explanation, the detectors consist of a series of layers each containing a number of measurement systems distributed throughout the detector. Once the corresponding measurements have been made and centralized, we obtain the values presented in this dataset.

This file contains 49 parameters recorded by the detector to classify each particle and 1,200,000 rows each representing a particle detected from a proton collision [7]. Due to processing time problems, we are not able to use the entire dataset, so we will use only 100,000 lines (with the amount of data used, the execution time of the code was approximately 10 hours). It is worth noting that the more data the model has to train the better the result, so we would expect an increase in accuracy if we use more lines.

Once the data we are going to use is loaded, we will do a cross validation with a ratio of 80-20.

B. Model training

To train the model we are going to use the *AdaBoostClassifier* available in the *sklearn* library [8], this classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases.

In order to check the efficiency of our model we are going to test the accuracy in particle detection by modifying the two most relevant parameters, which are:

- *n_estimators* → The maximum number of estimators at which boosting is terminated. In case of perfect fit, the learning procedure is stopped early.
- *max_depth* → The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure.

The function to be executed will be as follows:

def AdaBoostClass(n_estimators, max_depth)

Which will return the following:

accuracy → percentage of particles correctly classified
t → execution time
predict_Class → list with predicted particles
prob_Class → probabilities of classification of each particle

The aim is to find the combination of *n_estimators* and *max_depth* that achieves the best accuracy without excessively long execution time.

V. Experimental Results

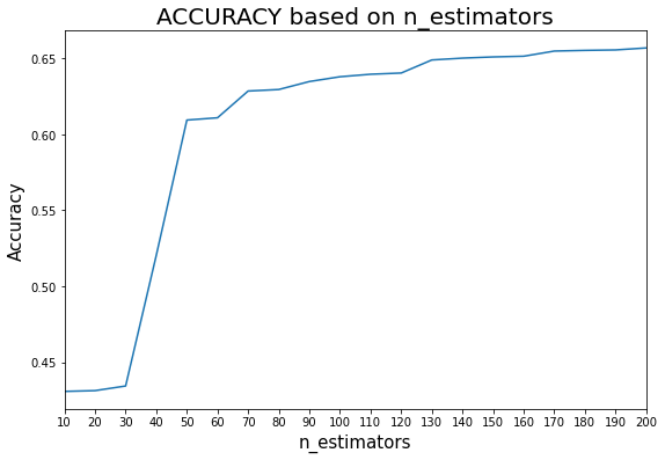
The experimental part is divided into 3 parts. First, we will analyze the accuracy of the model by modifying the parameters separately. The purpose of this is to see what the accuracy curve we obtain is like and compare it with the execution time to obtain the optimal value ranges for both parameters.

We will then modify both parameters at the same time with the optimal ranges obtained in the previous sections to achieve the highest possible accuracy. In this part we will use *GridSearchCV* to perform an exhaustive search on the specified parameter values.

All the code used can be found in the references via a link to GitHub [9].

A. Analysis n_estimators

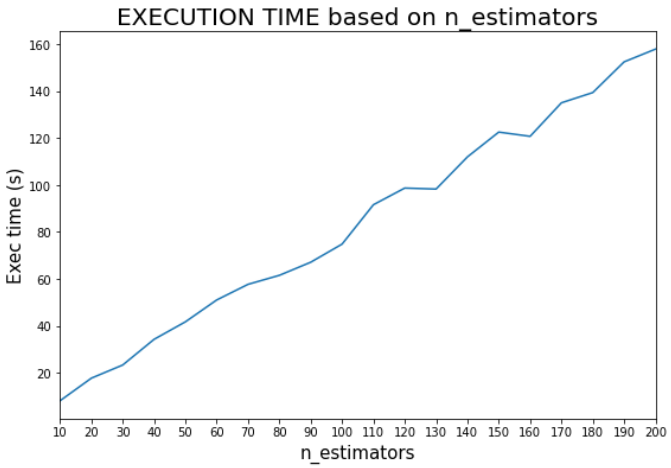
For this analysis we will train the model with a fixed *max_depth* value of 1 and test *n_estimators* values from 10 to 200. The results obtained are shown in the following graph:



Looking at the graph we can see that with *n_estimators* = 200 we obtain the best result with an accuracy of 0.657, looking at the range of values used we see 200 is the maximum value we have used. From this we can conclude that the higher the number of estimators, the higher the accuracy.

We can also observe that from *n_estimators* = 50 the results are relatively similar to those obtained with *n_estimators* = 200. So that, at least for the moment, the range 50-200 could be the optimal range.

The following is the graph of the execution times for each value used.

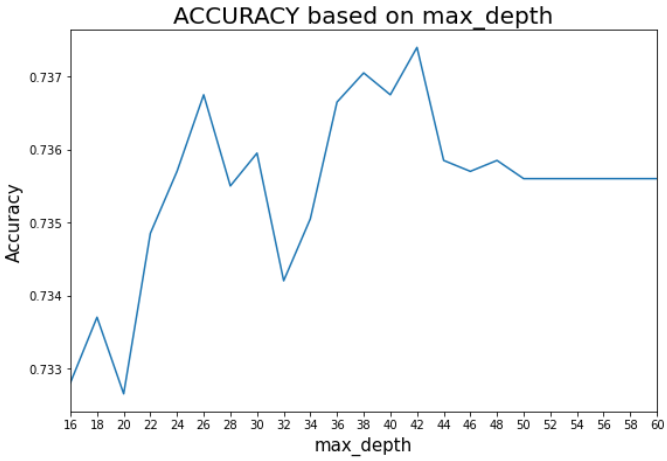


We can see that the time increases linearly as we increase the number of estimators. Seeing that 200 estimators do not represent a great improvement in accuracy compared to 50 and verifying the considerable increase in execution time that this implies, it is appropriate to modify the range we have chosen before.

Trying to achieve the best balance between accuracy-execution time, we will consider 50-100 estimators as the optimal range.

B. Analysis max_depth

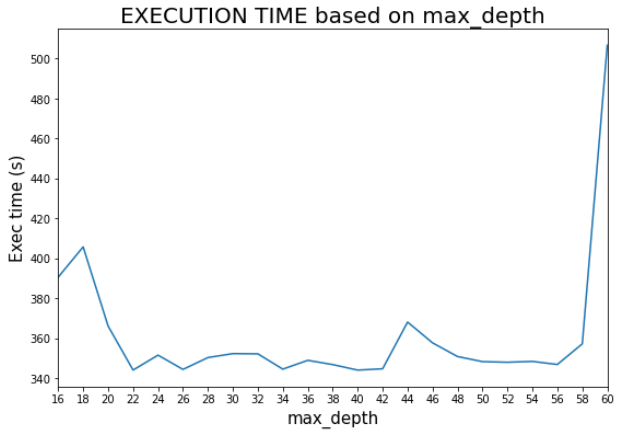
For this analysis we will train the model with a fixed *n_estimators* value of 50 and test *max_depth* values from 16 to 60. The results obtained are shown in the following graph:



Looking at the graph we can see that with *max_depth* = 42 we obtain the best result with an accuracy of 0.737.

The depth of the tree is deeply related to the number of features used to train the model, as we have discussed above, we have 49 different parameters. We can see in the graph that the accuracy stabilizes when reaching *max_depth* = 50, this is because as there are no more features to analyze the accuracy cannot increase more.

Looking at the graph we can see that a good range would be 35-50, now as we have done before, let's look at the execution times in this proposed range.



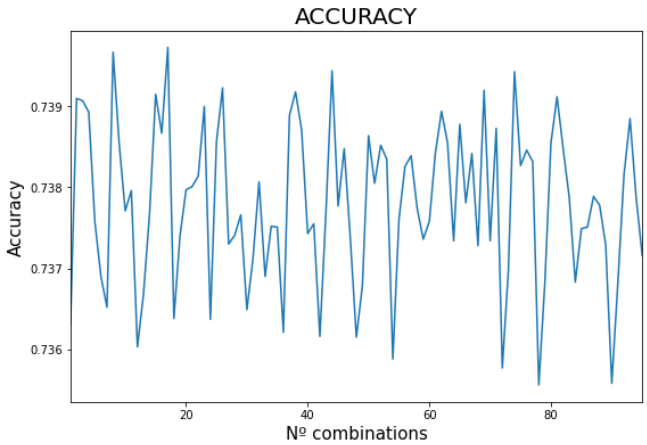
We can see that for *max_depth* the execution time makes practically no difference regardless of the values we use. Therefore, we promote the previously proposed range of 35-50 to the optimal range.

C. Analysis n_estimators & max_depth

Now that we have the ranges in which we are going to obtain the best achievable results with a reasonable execution time, we proceed to execute all possible combinations with these values:

OPTIMAL RANGES				
-	Min Value	Max Value	Step	Nº values
n_estimators	50	100	10	6
max_depth	35	50	1	16

We have a total of 96 possible combinations with these values



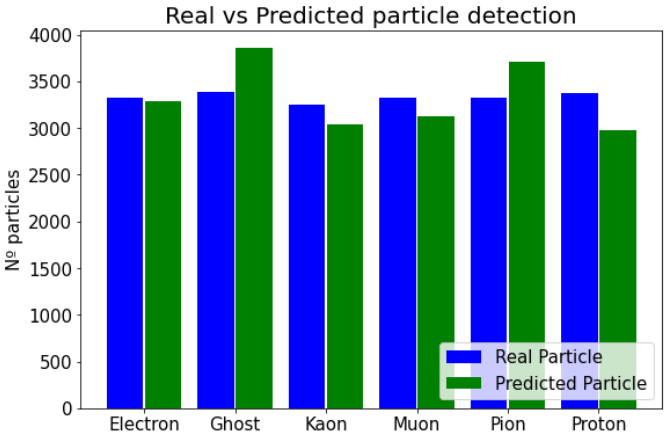
This graph shows the accuracies obtained with all the combinations discussed above. The best result obtained with our algorithm is the following:

n_estimators = 100
max_depth = 37

With these parameters we arrive at an accuracy of **0.740**.

D. Final Results

Once we have the optimal combination of *n_estimators* and *max_depth* we will analyze in more detail how the model has performed in predicting the particles resulting from collisions.



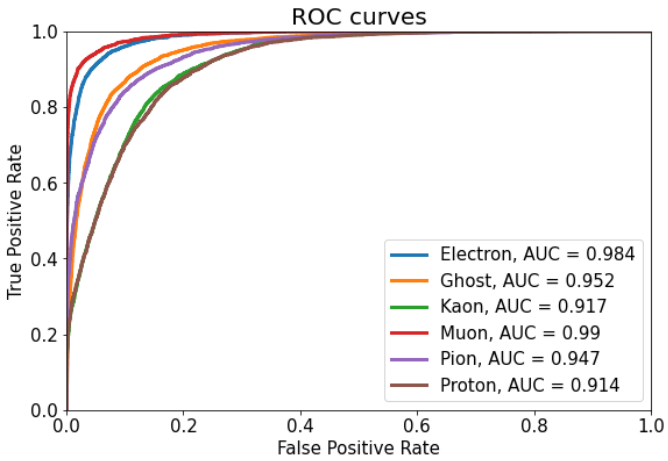
In the figure above we can see in blue the number of particles that the detector classified of each type and in green those that our machine learning model has predicted.

As discussed in the previous section, we have achieved an accuracy of **0.740**. From the previous graph we get the following results:

- The electron prediction of the model is excellent.
- The model has a tendency to misclassify Kaon, Muon and Proton particles.
- Misclassified particles are often considered as Pion or Ghost by the model.

These are the conclusions that we can draw visually from the graph, but they are observations without much scientific basis. To improve the conclusions about the performance of our model we will use the ROC curves. ROC curves consist of plotting the proportion of True Positives (TPR) against the proportion of False Positives (FPR), the closer the curve is to 100% True Positives the better the model.

In addition, we will also use the AUC (Area under the ROC curve) parameter, which provides an aggregate measure of performance at all possible classification thresholds. The AUC value ranges from 0 to 1. A model whose predictions are 100% wrong has an AUC of 0, while one whose predictions are 100% correct has an AUC of 1.



As we can see from the ROC curves, our model is particularly good at predicting Electrons and Muons. On the other hand, it makes some errors in the classification of Kaons and Protons. We see that the conclusions drawn from the previous bar chart are confirmed by these data.

Additionally, with the obtained AUC values we can see the order of the particles best predicted by our model:

1	Muon
2	Electron
3	Ghost
4	Pion
5	Kaon
6	Proton

VI. Related studies

As we have seen, the main problem here is that LHC produces a vast amount of data, and we need advanced techniques to be able to handle all this information. There exist several approaches, and some of these try to reduce the amount of data by discarding information that is not very likely to provide interesting results and others try to streamline hardware and software to manage a larger amount of data successfully. Here we mention the most important research regarding this so the reader can understand what the different approaches are and can locate this research in the entire problem.

- The first explored technique tries to minimize the number of parameters used to identify a particle, i.e., trying to use a kind of dimensionality reduction but instead of creating the uncorrelated variables algebraically as in other techniques as PCA, using neural networks. The conclusion of this research is that it is only useful in very specific classification tasks, so it is a little improvement [6].

- The second explored technique also uses neural networks but with a different goal. Simulators that faithfully describe the low-level interactions of particles with matter cannot be used as much as needed because they demand a lot of computing power too and are very expensive. This approach tries to create a fast simulator using a neural network that learns from the events simulated by the full simulators. The performance of the simulation in terms of computing cost increases considerably, but the accuracy is not that high [6].

- The last research focuses on one technique to limit models size using quantization minimizing the reduction of the performance. This is a key point to meet latency and energy consumption requirements. Researchers show how on-chip resource consumption can be reduced by a factor of 50 without much loss in accuracy, but it is not capable to manage the most latency-constrained computing environments [3].

VII. Discussion

In this section we would like to make some observations about the project.

As mentioned at the beginning of the report, we have used only 100,000 lines of the dataset for the development of the project. To obtain better results we should use all the available data, for which we would need a higher computational capacity.

As for the development of the project, it has been very interesting to go deeper into machine learning algorithms and their inner workings in order to make small modifications to optimize the results. We consider that we have learned a lot about how to approach a project to write a paper, as well as to develop the experimental part of the project for its later explanation.

VIII. Conclusion

In conclusion, we have achieved good accuracy in particle prediction with the computing power we had available. In the case of applying an algorithm like this in reality, it could be improved by using a larger amount of data to train the model and with the use of more powerful computers. This way we would achieve good results using far fewer resources than currently needed to identify particles in collisions.

IX. References

[1] Speeding up machine learning for particle physics, CERN (2021). https://home.cern/news/news/physics/speeding-machine-learning-particle-physics

[2] The Large Hadron Collinder, CERN. https://home.cern/science/accelerators/large-hadron-collider

[3] Coelho, C.N., Kuusela, A., Li, S. *et al.* Automatic heterogeneous quantization of deep neural networks for low-latency inference on the edge for particle detectors. *Nat Mach Intell* **3,** 675–686 (2021).

[4] Apollinari, G., B´ejar Alonso, I., Br¨uning, O., Lamont, M. & Rossi, L. High-luminosity large hadron collider (hl-lhc): Preliminary design report. Tech. Rep., Fermi National Accelerator Lab.(FNAL), Batavia, IL (United States) (2015).

[5] Zhuang, B., Shen, C., Tan, M., Liu, L. & Reid, I. Towards effective low-bitwidth convolutional neural networks. In Proceedings of the

IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)

[6] Dan G, K. C., D.W. Deep learning and its application to LHC physics.

[7] LHCb - Dataset (training.csv) https://www.kaggle.com/lavanyashukla01/particle-identification

[8] AdaBoostClassifier https://scikit-learn.org/stable/modules/classes.html#module-sklearn.ensemble

[9] Code repository - GitHub https://github.com/pablolostao/lhcparticlesdetection/blob/main/cs550_project.py

[10] Detector layers https://sites.uci.edu/energyobserver/2012/11/26/introduction-to-the-atlas-detector-at-the-lhc/