Particle Position Estimation: A Case Study of Resistive Silicon Detectors

Arda Eren Dogru
Politecnico di Torino
Student ID: S331330
ardaeren.dogru@studenti.polito.it

***Abstract*—The goal of this data science project is to estimate the locations of particles that go through a resistive silicon detector (RSD) sensor, using the features derived from the signals recorded by the sensor pads. The project builds a multi-output regression pipeline with three machine learning algorithms: k-nearest neighbors (KNN), random forest, and gradient boosting (CatBoost). The average Euclidean distance metric evaluates the performance of the pipeline. The report shows the data analysis, feature engineering, model selection, hyperparameter optimization and results of the project, and also provides some discussion and future work. The proposed method outperforms a simple benchmark set for the problem and produces satisfactory results.**

I. Problem overview

The problem that we aim to solve in this project is to predict the positions of particles passing through a resistive silicon detector (RSD) sensor, based on the features extracted from the signals measured by the sensor pads. This is a multi-output regression problem, where the outputs are the x and y coordinates of the particle hit on the sensor plane.The dataset is divided into two parts:

- a *development* set, containing 385,500 records with labels
- an *evaluation* set, containing 128,500 records without labels

The objective is to construct a regression model using the development set, which can accurately assign x and y coordinates to the instances in the evaluation set.

Based on the development set, we can draw the following observations:

- The signals from 18 different sources are characterized by 5 different numerical features, 90 features in total, namely: root mean square value, maximum positive voltage value in mV, negative maximum voltage value in mV, delay from a reference time in ns, and area under the signal for each source.
- The labels x and y are discrete variables that take values from 200 μm to 600 μm, with increments of 5 μm.
- Due to hardware limitations in the data acquisition phase, only 12 pads are available in the sensor, while 18 readings of each feature are recorded for each event. Therefore, some of the 18 features do not represent actual signals, but noise.

To get a better understanding of the data at hand, we can inspect features of signals from different sources visually.

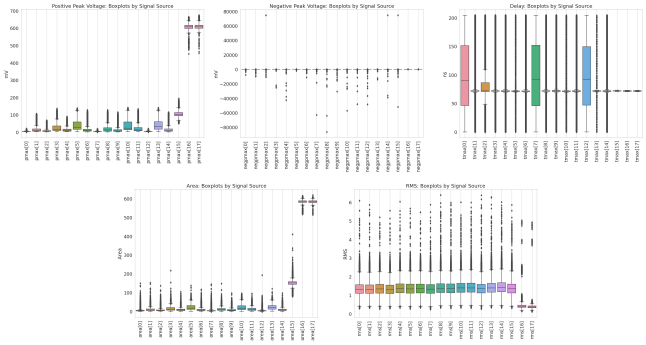

Fig. 1. Boxplots of the Measurements from Different Sources

Fig. 1 shows boxplots of signal features from different sources. The figure shows some interesting patterns. The maximum positive voltage values are between 0 mV and 200 mV, with the exception of sources 0, 7, 12, 15, 16, and 17, which have a distinct distribution. Also, the distribution of area measurement shows a similar pattern with the positive maximum voltage feature. The negative maximum voltage values have some outliers that are either positive or extremely negative. The delay values are also between 0 ns and 200 ns, but some sources, such as 0, 7, 12, 15, 16, and 17, statistically deviate from the rest. A possible explanation for these patterns is that these sources are the ones with noise, while the pads are the sources of the other measurements.

To examine the signal features from different sources, we plotted the mean of the feature values for each pair of x-y coordinates. Fig. 2 illustrates the mean of the maximum positive voltage values for each source. The figure indicates the locations of the pads on the sensor plane, which correspond to the regions with higher voltage values. Furthermore, the figure reveals that the sources 0, 7, 12, 15, 16, and 17 have a distinct pattern from the rest of the sources, suggesting that they are likely to be noise. Additionally, source 15 exhibits an interesting phenomenon: the voltage values are lower on the edges of each triangular-like region, which may indicate information about the location of the particle.

We also plotted the mean of the negative maximum voltage values, delay and the area values for each source, and observed analogous patterns as the positive maximum voltage values. These features provided relevant information about the

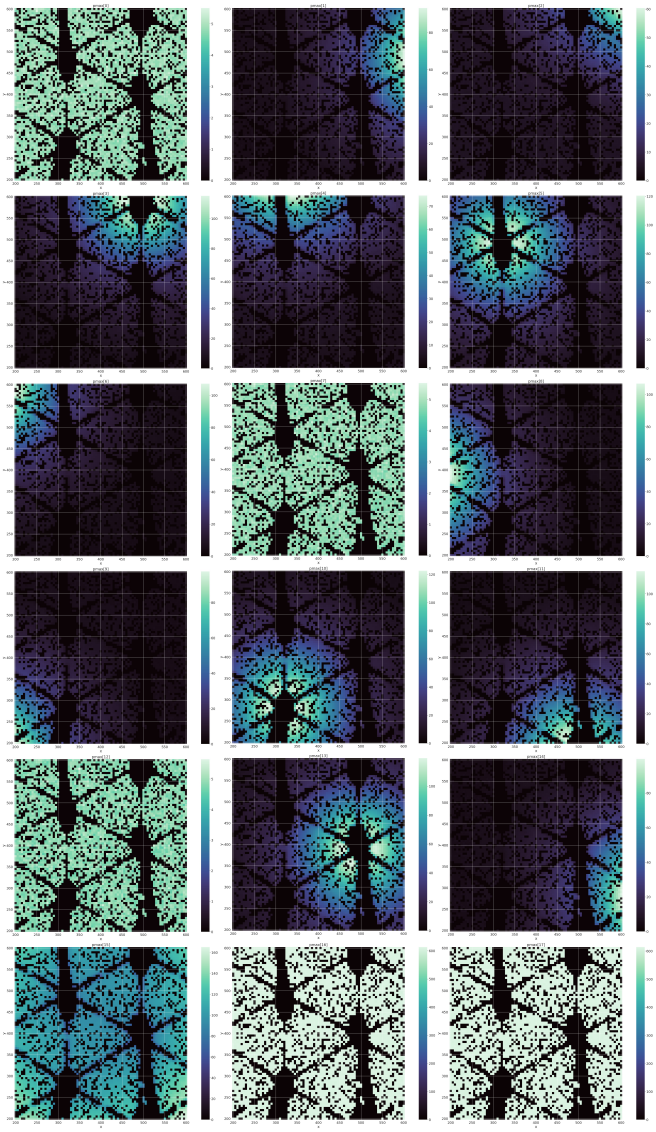

Fig. 2. Mean of maximum positive voltage values for each pair of x-y coordinates for different sources

location of the particle, as they varied according to the position of the particle. However, the root mean square (rms) values did not provide any meaningful information in the graphs, as they were mostly constant across the sources.

II. Proposed approach

A. Preprocessing

In order to preprocess the data, we first checked if data have any missing or duplicate values. Then, we removed the noisy features. As shown in Fig. 1 and Fig. 2, sources 0, 7, 12, 15, 16, and 17 are distinct from the rest of the sources, indicating that they do not represent actual signals, but noise. Therefore, we deleted all the features coming from these sources, except for maximum positive voltage feature of source 15. Source 15 is noisy, but rather it reveals information about the location of the particle on the sensor plane, as it exhibits lower voltage values on the edges of each triangular-like region. This information can be useful for predicting the position of the particle. We also decided to drop the delay feature for the pad sources, as it has a relatively low feature importance value to other features according to a simple the decision tree model.

We also removed the some records with maximum negative voltage features, as they might be outliers of the data acquisition process. Fig. 1 displays the boxplot of the negative maximum voltage values for all the sources. As expected, most of the values are negative, but there are some positive values that diverge from the rest of the data. These values could introduce noise or bias to the models, so we excluded them from the dataset. Likewise, we eliminated the rows with extremely negative maximum voltage below a threshold of -200, since the amplitude of maximum negative voltage must be less than the amplitude of maximum positive voltage due to the nature of the signal. These values might also be outliers. By removing these records, we reduced the size of the dataset from 385,500 to 385,192.

To reconstruct the hitting positions of particles, we developed novel features based on the fractional amplitude, which is inversely proportional to the distance from the pad [1]. We defined three types of fractional values as follows:

- Positive maximum voltage value fraction: the fraction of the positive maximum voltage value of each pad over the sum of the positive maximum voltage values of all pads.
- Negative maximum voltage value fraction: the fraction of the negative maximum voltage value of each pad over the sum of the negative maximum values of all pads.
- Area fraction: the fraction of the area of each signal measured in a pad over the total area of all signals measured on pads.

After preprocessing, we obtained 73 features. 72 of these features are the positive maximum voltage, negative maximum voltage, and area features of the pads, and their corresponding fractions. The remaining feature is the positive maximum voltage value of the source 15.

B. Model selection

We considered three machine learning algorithms for the multi-output regression task of predicting the x and y coordinates of the particle hit on the sensor plane. The algorithms are:

- *Random Forest Regressor:* This is an ensemble learning algorithm that combines multiple decision trees to create a more accurate and robust model. It can handle non-linear and complex data well, as well as missing values and outliers. It can also provide feature importance and variable selection . Random forest regressor is suitable for our data because it can handle a large number of numerical features and perform well on regression tasks [2], [3].
- *KNN Regressor:* This is a simple and intuitive algorithm that predicts the target by local interpolation of the targets

associated with the nearest neighbors in the training set. It can adapt to irregular feature spaces and different data types, but it suffers from curse of dimensionality. KNN regressor is suitable for the given data because it can capture the local patterns and similarities among the numerical features and predict regression labels accurately. [4].

- *Catboost Regressor:* This is a gradient boosting algorithm that successfully handles both numerical and categorical features and outperforms existing publicly available implementations of gradient boosting in terms of quality on a set of popular publicly available datasets. It has a GPU implementation of learning algorithm and a CPU implementation of scoring algorithm, which are significantly faster than other gradient boosting libraries on ensembles of similar sizes . Catboost regressor is suitable for the our data because it can handle numerical features efficiently and achieve high performance and speed on regression tasks [5], [6].

We chose these algorithms because they have different strengths and weaknesses, and they can provide complementary information about the problem. We also wanted to compare the performance of a traditional ensemble method (Random Forest), a simple distance-based method (KNN), and a modern boosting method (Catboost) on our dataset.

C. Hyperparameters tuning

As our data is large, we split our data 90/10 train/test, and used the test set to evaluate the performance of the tuned model.

The hyperparameter tuning process was performed with Optuna, a framework for automated optimization of machine learning models . Optuna is better than grid search because it uses Bayesian optimization to efficiently explore the hyperparameter space and find the optimal values. Grid search, on the other hand, exhaustively tests all the possible combinations of hyperparameters, which can be very time-consuming and wasteful [7].

To speed up the training and inference of the models, we used the NVIDIA's rapids.ai stack, which is a collection of open source libraries that enable GPU-accelerated data science [8]. We used cuml, cudf, and cupy libraries to perform data manipulation, machine learning, and array operations on the GPU, respectively. These libraries are compatible with popular Python frameworks, such as scikit-learn, pandas, and numpy, and can significantly improve the performance and efficiency of the models.

Table 1 shows the hyperparameters that we tuned for each model using Optuna. We ran each Optuna study with 20 trials for random forest and KNN and 50 trials for the CatBoost with overfitting detector, and selected the best trial based on the lowest average Euclidean distance on the test set.

TABLE I
HYPERPARAMETERS TUNED FOR EACH MODEL

Model	***Hyperparameter***	***Values***	***Step***
Random Forest	n_estimators	low: 100, high:1000	100
	max_features	['auto', 'sqrt', 73]	
	min_samples_split	low:2, high:6	1
	min_samples_leaf	low:1, high:6	1
KNN	n_neighbors	low:5, high:1000	log
	algorithm	['auto', 'brute']	
CatBoost	l2_leaf_reg	low:1, high:21	2
	depth	low:4, high:14	1
	border_count	low:128, high:512	log
	bagging_temperature	low:0.1, high:5	log
	random_strength	[0, 0.25, 0.50, 0.75, 1]	
	iterations	5000	
	learning_rate	0.4	
	od_type	'iter'	

TABLE II
TUNED HYPERPARAMETERS FOR EACH MODEL

Model	***Hyperparameter***	***Values***
Random Forest	n_estimators	1000
	max_features	'sqrt'
	min_samples_split	5
	min_samples_leaf	2
KNN	n_neighbors	19
	algorithm	'auto'
CatBoost	l2_leaf_reg	13
	depth	7
	border_count	319
	bagging_temperature	0.2
	random_strength	1

III. RESULTS

As shown in Table 2, the optimization process resulted in the best configurations for each of the three models: Catboost regressor, Random Forest, and KNN. The performance of the models was evaluated using the Euclidean distance metric on the training, test, and public datasets. The Catboost regressor achieved the lowest Euclidean distance values on the training and public datasets, with 3.52 and 4.54 respectively, but had same value with KNN on the test dataset with 4.18. The Random Forest model had slightly higher Euclidean distance values than the Catboost regressor on the training, test and public datasets, with 3.57, 4.56 and 4.97 respectively. The KNN model performed well on the training and test datasets, with 3.96 and 4.18 Euclidean distance values respectively, but had a much higher value of 5.6 on the public dataset.

The results indicate that all models were able to fit the training and test datasets well, but the Catboost regressor was the most robust and generalizable model for the public dataset, which contained unseen and potentially noisy data. One possible explanation for the superior performance of the Catboost regressor is that it uses gradient boosting, which is an ensemble learning technique that combines multiple weak learners to create a strong learner. Gradient boosting can reduce the bias and variance of the model, and improve its accuracy and stability. On the other hand, the KNN model is a simple and non-parametric method that relies on the distance between the data points to make predictions. The KNN model is sensitive to outliers and noise, which can affect its performance on the public dataset. Therefore, the

Catboost regressor is the best model for this problem, as it can handle complex and heterogeneous data better than the Random Forest and KNN models.

For the public score, we used the entire development dataset to train the Catboost model and increased the iteration count to 100,000 and decreased the learning rate to 0.01. This process improved the model accuracy by reducing the overfitting and increasing the stability of the gradient boosting algorithm. According to the Catboost documentation, the number of iterations and the learning rate are two of the most important hyperparameters that affect the quality of the model. By increasing the number of iterations, we allowed the model to learn more from the data and create more complex decision trees. By decreasing the learning rate, we controlled the speed of the model update and prevented it from converging too quickly to a suboptimal solution. Additionally, we rounded the x and y coordinates to nearest 5, which is a very basic approach, but suitable for our problem. The reason for rounding is that the x and y labels are increasing by 5 units, so there is no need to keep more precision than that. As a result, we achieved a score of 3.95 on the public dataset, which is lower than the previous score of 4.54 and indicates a better performance.

IV. Discussion

In this section, we discuss the key findings and insights, the limitations and future work of our project.

The catboost model's feature importance analysis revealed that feature engineering had a significant impact on the model performance. By introducing fractional amplitude features, which are derived from the physical properties of the sensor signals, we were able to capture more information about the position of the particle and reduce the noise in the data. This demonstrates the importance of domain knowledge in feature construction and selection.

We further optimized the hyperparameters of each model using Optuna, which is a framework for automated optimization of machine learning models [7]. We found that optimizing hyperparameters led to noticeable performance gains, highlighting the value of automated tuning for model refinement. We also used NVIDIA's RAPIDS.ai stack, which is a collection of open source libraries that enable GPU-accelerated data science. We found that the use of GPU-accelerated computing significantly reduced training and inference times, demonstrating the potential of GPU-accelerated computing for large-scale data science tasks [8].

However, our project also has some limitations and challenges that need to be addressed in the future. One of the main limitation is the feature exploration. We only used a subset of the available features and derived some novel features based on the fractional amplitude. However, there may be other feature engineering approaches, such as spectral analysis or time-series analysis of the sensor signals, that could potentially extract more informative and relevant features for the position estimation problem. The second limitation is the model ensembles. We only evaluated the performance of individual models, but not the performance of model ensembles. Combining the strengths of different models through model ensembling techniques could potentially yield even better accuracy and generalization capabilities. A third limitation is the interpretability. While CatBoost achieved the best results, its black-box nature makes it challenging to interpret the underlying relationships between features and predictions. Investigating techniques like SHAP values could shed light on the model's decision-making process and provide more transparency and explainability.

This project successfully demonstrated the effective use of machine learning techniques to estimate particle positions in a resistive silicon detector. By carefully addressing data preprocessing, feature engineering, model selection, hyperparameter tuning, and computational efficiency, we achieved satisfactory results that outperform a simple baseline.

References

[1] M. T. et al, "Resistive ac-coupled silicon detectors: Principles of operation and first results from a combined analysis of beam test and laser data," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 2021.

[2] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, p. 5–32, 2001.

[3] R. Genuer, J.-M. Poggi, and C. Tuleau-Malot, "Variable selection using random forests," *Pattern Recognition Letters*, vol. 31, no. 14, p. 2225–2236, 2010.

[4] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, p. 175–185, 1992.

[5] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "Catboost: unbiased boosting with categorical features," in *Advances in neural information processing systems*, p. 6638–6648, 2018.

[6] A. V. Dorogush, V. Ershov, and A. Gulin, "Catboost: gradient boosting with categorical features support," in *Workshop on ML Systems at NIPS 2017*, vol. 1, 2018.

[7] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[8] *RAPIDS: Libraries for End to End GPU Data Science*.