Position estimation in Resistive Silicon Detectors

Davide Fassio
Politecnico di Torino
s323584
s323584@studenti.polito.it

Davide Elio Stefano Demicheli
Politecnico di Torino
s331531
s331531@studenti.polito.it

***Abstract*—In this report we present the techniques we used to estimate the 2-D position of subatomic particles passing through a RSD (Resistive Silicon Detector), in presence of noisy readings. The proposed approach leverages the noise resistance of Random Forests and their ability to predict multiple output variables. The trained model is able to correctly estimate the position of the particle with an average error of $\sim 1\%$. The proposed approach outperforms a naive solution we defined for the problem obtaining, overall, satisfactory results.**

I. Problem overview

The task we are facing is a multi-output regression. Given relevant features of the signal measured by the pads of the sensor at the passage of each particle, we have to predict the x and y coordinates of the position in which the particle hit the detector. For each event 18 readings are provided for these features, but only 12 of them come from the pads of the detector, the remaining 6 contain noise.

The dataset is divided into two parts:

- a development set, containing 385500 labeled events.
- an evaluation set, comprised of 128500 events.

A record is composed of 90 attributes, 18 readings for each of the 5 features:

- pmax: the amplitude of the positive peak.
- negpmax: the amplitude of the negative peak.
- area: the area under the signal.
- tmax: the time at which pmax is recorded.
- rms: the root mean square value of the signal, also called effective value.

There are no missing values.

By plotting the position of the particles in the training set (Fig. 1) we can see that the measurements were conducted in a square area of side 400 µm. We can also clearly identify the position of the sensors because reflective pads and wires were used to acquire the signals. It is also interesting to note that to build the development set the laser was positioned only on multiples of 5 µm.

In order to distinguish the noisy readings from the pads' ones we also plotted, for each of the 18 measurements, the x and y position as a function of the different features. Figure 2 and 3 show the relationship between pmax (the maximum height of the input signal) and the x and y coordinates respectively. Analyzing the correlation between the two variables we can clearly observe that the readings with id: 0, 7, 12, 15, 16, 17 contribute only with noise.

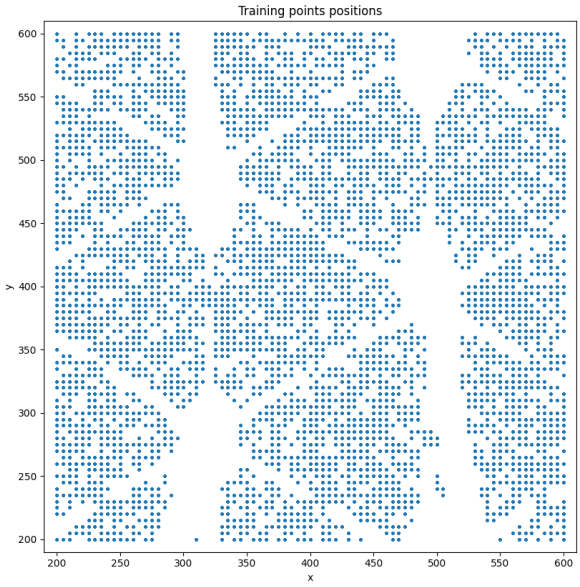

Fig. 1. Position of the points in the training dataset

Given the knowledge we extracted from the visualizations shown in Fig. 1, 2, 3, and the layout of the RSD provided in the text of the assignment, we are also able to reconstruct the position of the 12 pads (see table I). The values obtained are consistent with the ones presented by F. Siviero et al. [1] . This information can be used to extract the *distance from the pad* feature and train more advanced models.

II. Proposed approach

A. Preprocessing

We first trained a random forest with the default hyperparameters to establish an initial baseline.

To be able to appreciate how the model was performing at each step of the feature selection stage, we used the holdout technique (70% training / 30% validation) to split the dataset. This is possible because the development dataset has enough records for the validation sample to be considered representative.

We then checked whether all the features carried significant information or not. Thanks to the random forest we trained, we

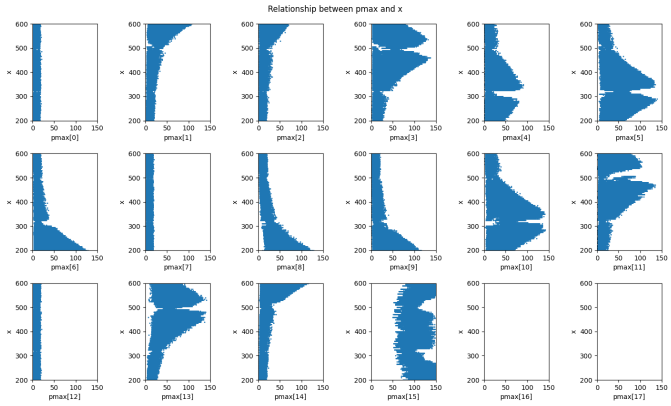

Fig. 2. x coordinate as function of pmax.
The plot of the readings 16 and 17 appear empty because their values are out of range ($pmax > 400$)

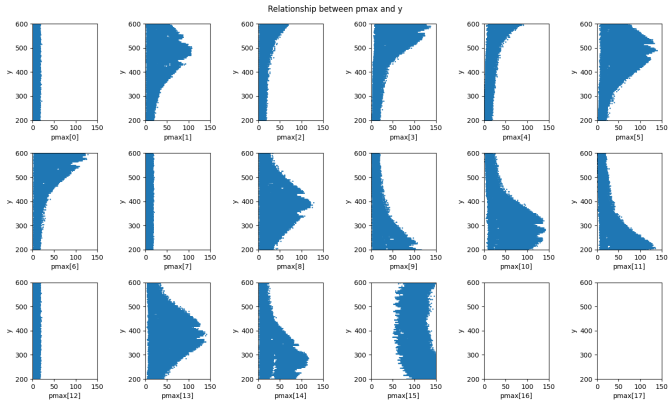

Fig. 3. y coordinate as function of pmax.
The plot of the readings 16 and 17 appear empty because their values are out of range ($pmax > 400$)

Sensor ID	Position (x, y)
1	(675, 490)
2	(675, 700)
3	(495, 595)
4	(315, 700)
5	(315, 490)
6	(135, 595)
8	(135, 385)
9	(135, 175)
10	(315, 280)
11	(495, 175)
13	(495, 385)
14	(675, 280)

TABLE I
POSITION OF THE SENSORS IN THE RSD

Feature	Importance
pmax[8]	0.346
pmax[11]	0.233
rangemax[10]	0.116
rangemax[5]	0.066
pmax[10]	0.043
rangemax[13]	0.025
pmax[13]	0.021
rangemax[11]	0.018
negpmax[13]	0.018
rangemax[8]	0.018
pmax[9]	0.017
rangemax[9]	0.016
rangemax[4]	0.012
pmax[5]	0.010
rangemax[3]	0.010
rangemax[1]	0.005
negpmax[3]	0.005
negpmax[10]	0.004
rangemax[6]	0.004
rangemax[2]	0.002
rangemax[14]	0.002
...	...

TABLE II
IMPORTANCE OF THE BEST 20 FEATURES GIVEN BY A DEFAULT RANDOM FOREST AFTER ADDING RANGEMAX

were able to determine the importance of each feature for the model. Since the set of features related to tmax and rms were negligibly contributing to the regression, we discarded them obtaining a significant improvement in the model performance.

We proceeded testing whether to use or not the noisy readings, and, surprisingly, the model was more accurate including them, so we decided to keep them.

We then experimented with the creation of a new set of features, consisting in the difference between the positive and negative peaks for each of the 18 readings, which we called rangemax ($rangemax = pmax - negpmax$). The performance of the random forest trained with these new features improved and the *feature_importances_* attribute reflected the fact that rangemax is a relevant feature for the model (see table II).

Since random forests are based on decision trees, which work on one feature at a time, we did not normalize the input data.

B. Model selection

We elected to use a Random Forest Regressor for its noise resistance and its native support to multi-output regression. This approach has also been employed by M. Tornago et al. [2].

These good properties derive from the structure of the model. The random forest is an ensemble model, in other words it leverages multiple simpler models, in this case decision trees trained on various subsets of data and features, to make predictions. By tweaking the hyperparameters of the embedded models (for example decreasing max_depth or max_features), it is possible to reduce the accuracy of the fitting on training data for each tree, so that the final model is less prone to overfitting. Moreover, averaging all the outputs the final result is less affected by noise. Another advantage of this model is that, deriving from the decision trees, it inherits some degree of interpretability.

C. Hyperparameters tuning

The Random Forest uses a wide set of hyperparameters that need careful tuning to extract all its potential.

We focused our efforts on:

- Max features: number of features examined at each split.
- Max depth: maximum depth of the decision trees.

As splitting criterion we used the squared error, because it is based on the same computations required for the scoring metric of the problem ($avg\_distance$).

We identified the best configurations of hyperparameters through thorough grid searches. To balance performance and accuracy we used k-fold cross validation with $k = 3$. We initially explored a broader range of combinations with a lower number of estimators. Subsequently, we focused our attention on a more refined set of values, increasing the number of estimators, to strike a balance between execution time constraints and accuracy of the model.

The first grid search was conducted with 100 estimators (see table III). From the results we noticed that it was better to not limit the growth of the trees (setting the max_depth parameter to None).

Parameter	Values
Number of estimators	100
Max features	['sqrt', 10, 20, 30, 40, None]
Max depth	[10, 20, None]

TABLE III
HYPERPARAMETERS CONSIDERED IN THE FIRST GRID SEARCH

Then we refined max_features with 200 estimators (see table IV). Out of 72 features the model performed better with fewer attributes to choose at each split, around $10 \sim 20$.

It is important to note that usually, considering a greater number of estimators, these work better with fewer features at each split, because the larger number of trees compensate the lower accuracy of the single one.

Parameter	Values
Number of estimators	200
Max features	['sqrt', 10, 20, 30]

TABLE IV
HYPERPARAMETERS CONSIDERED IN THE SECOND GRID SEARCH

The last step was to decide the number of estimators, using again the holdout technique (with the same proportions used previously) to split the dataset.

This hyperparameter is a trade-off between precision and time (see Fig. 4). In our case we were interested in the accuracy of the results, so we decided to use 1000 estimators since the model training required a reasonable amount of time.

III. Results

The best performing configuration of the Random Forest Regressor, selected in the previous section, was n_estimators: 1000, max_features: 10, max_depth: None. In the local validation step it achieved: 3.902. We then trained it on all available development data and evaluated it on the public leaderboard

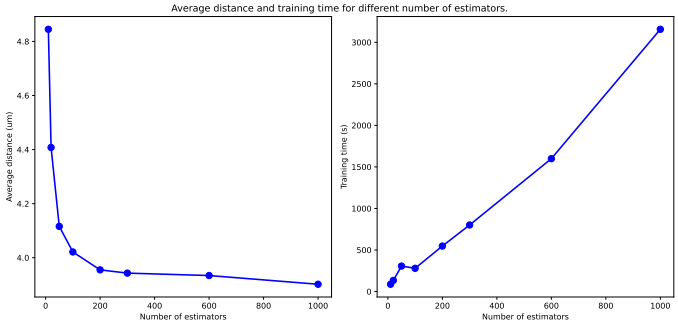

Fig. 4. The time increases linearly with the number of estimators, while the average distance plateaus for n=200

where it scored: 4.635. Since the local and public scores are similar we expect the model to be able to generalize well with new data, as it does not present sign of overfitting.

This solution clearly outperforms the public baseline set at 6.629.

It is also a great improvement over a simple ML pipeline we defined for comparison, in which no preprocessing was done and the model used was a default random forest. This naive solution scored 4.436 over the validation dataset and 5.595 on the public leaderboard.

IV. Discussion

The proposed approach accomplished the task of detecting the correct position of the particle with satisfactory accuracy in presence of a considerable amount of noise.

We report here some aspects that might be worth considering to further improve the obtained results within this research context.

Due to time constraints, the grid search tested considered only a small subset of hyperparameters. Other ones that can be analyzed are:

- Max samples (setting Bootstrap=true): the number of records sampled from the training dataset to train each tree.
- Min samples leaf: minimum number of samples required to be a leaf node. Setting it to a higher number can reduce overfitting. P. Probst [3] recommends 5 for a regression task.

As proposed by Bergstra and Bengio [4] it is also possible to use randomly chosen trials to achieve better and faster convergence than a classical grid search.

Moreover, new nonlinear features can be created to aid the model with this task or more advanced models can be used:

- Feed-forward neural network: the tabular nature of this task can be exploited by neural networks without the need to overhaul the existing features representation.
- Convolutional neural network: this particular architecture can be used in two different ways:
  - Converting the 60 input features (excluding the noise) in a 3-tensor of shape (features per pad)x(rows of pads)x(columns of pads), in our case 5x3x4.

– Using the knowledge of the positions of the pads it is possible to create an image encoding the annuli where it is likely that the particle hit.

REFERENCES

[1] F. Siviero et al. First experimental results of the spatial resolution of rsd pad arrays read out with a 16-ch board. *NIM*, A, 2022.

[2] M. Tornago et al. Silicon sensors with resistive read-out: Machine learning techniques for ultimate spatial resolution. *NIM*, A, 2022.

[3] P. Probst et al. Hyperparameters and tuning strategies for random forest. *WIREs*, 9, 2019.

[4] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13, 2012.