Real-time implementation with FPGA-based DAQ system of a probabilistic disruption predictor from scratch

Real-time implementation with FPGA-based DAQ system of a probabilistic disruption predictor from scratch

J. Vega[1,] *, M. Ruiz[2], E. Barrera[2], R. Castro[1], G. A. Rattá[1], S. Dormido-Canto[3], A. Murari[4] and JET Contributors*

EUROfusion Consortium, JET, Culham Science Centre, Abingdon, OX14 3DB, UK
[1]Laboratorio Nacional de Fusión, CIEMAT. Avda. Complutense, 40. 28040 Madrid. Spain
[2]Grupo de Investigación en Instrumentación y Acústica Aplicada: Universidad Politécnica de Madrid, Madrid, Spain
[3]Departamento de Informática y Automática, UNED, Madrid. Spain
[4]Consorzio RFX. Corso Stati Uniti 4, 35127 Padova, Italy
**See the author list of "Overview of the JET results in support to ITER" by X. Litaudon et al. to be published in Nuclear Fusion Special issue: overview and summary reports from the 26th Fusion Energy Conference (Kyoto, Japan, 17-22 October 2016)*

Abstract

Real-time (RT) disruption prediction (DP) is essential to trigger mitigation actions that avoid irreversible damage to the devices. This paper deals with disruption mitigation alarms and performs the RT implementation of a probabilistic predictor. The RT implementation has been carried out with a fast controller with DAQ FPGA-based data acquisition devices corresponding to ITER catalogue (in particular, a reconfigurable Input/Output platform has been used). Up to three input signals have been used and relevant information for the prediction is extracted from the temporal and the frequency domains. The signals are read from the JET database. Then D/A conversions are carried out and used as inputs to the data acquisition card. In this way, the whole process of digitization, data analysis and prediction is performed. The computation time for each prediction takes less than 200 μs.

Keywords: Disruption prediction, real-time, machine learning, Venn predictors, data acquisition, FPGA

1. Introduction

At present, important efforts to develop physics models for disruption prediction are being carried out. However, so far, they do not cope with success rates close to 100%. Instead, data-driven models can be used. The objective is to build classifiers to distinguish between disruptive and non-disruptive behaviours. The main drawback for DP is the need of large training sets to generate reliable classifiers. For example, the JET APODIS predictor was trained with 7648 non-disruptive discharges and 521 unintentional disruptions [1]. Of course, ITER or DEMO cannot wait for such number of discharges to have reliable predictions.

This article performs a specific implementation of the probabilistic *predictor from scratch* described in

* Corresponding author. Tel: +34.91.346.6474
Fax: +34.91.346.6124
E-mail address: jesus.vega@ciemat.es

[2]. The name *predictor from scratch* comes from the fact that the learning process starts with only 1 disruptive discharge and at least 1 non-disruptive discharge. Its specific implementation uses an adaptive Venn predictor [3] that requires very few training examples and shows a high learning rate. The objective of the paper is not to analyse the predictor results but the real-time requirements to perform predictions in a FPGA-based data acquisition card.

Section 2 describes the steps to make Venn predictions, where these steps are the ones implemented in the FPGA. Section 3 shows the general architecture of the predictor. Section 4 details the implementation in a FPGA-based system that meets the requirements established in [4].

2. Venn predictors and a toy example

A general Venn predictor [3] requires a training set $\{z_1, \ldots, z_n\}$ that verifies the assumption of independent and identically distributed data. Each z_i is a pair $(\mathbf{x}_i, y_i)$ where $\mathbf{x}_i$ is a feature vector (whose components are characteristics of distinctive nature to distinguish between vectors) and $y_i \in \{Y_1, \ldots, Y_M\}$ is the label of the class to which $\mathbf{x}_i$ belongs to. The objective of a Venn predictor is the estimation of the probability that a new feature vector $\mathbf{x}$ belongs to one class from the possible M classes.

The Venn prediction framework assigns each of the possible classifications Y_j to $\mathbf{x}$ and divides all examples $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n), (\mathbf{x}, y)\}$ into a number of categories by means of a taxonomy function. This taxonomy function assigns to each pair $(\mathbf{x}_i, y_i)$ its category τ_i from a finite set of categories.

A Venn predictor computes a probability matrix where the row j is the empirical probability distribution of the labels in a category τ that contains $(\mathbf{x}, Y_j)$:

$$P_M = \left(\begin{array}{ccc} p^{Y_1}(Y_1) & p^{Y_1}(Y_2) & p^{Y_1}(Y_M) \\ p^{Y_2}(Y_1) & p^{Y_2}(Y_2) & p^{Y_2}(Y_M) \\ \hdashline p^{Y_M}(Y_1) & p^{Y_M}(Y_2) & p^{Y_M}(Y_M) \end{array}\right)$$

Once the above matrix has been formed, the Venn prediction process assigns as label of $\mathbf{x}$ the label of the column with the highest mean value. The prediction probability is the interval between the minimum of the winner column and the maximum of such column.

For the sake of clarity, let's assume the two-class problem of figure 1. It is a one-dimensional problem and the objective is to classify the point x = 5.5.

Fig. 1: Given the training set {(-1, *A*), (1, *A*), (4, *B*), (6, *B*)}, predict the class and probability interval of x=5.5 with the nearest centroid taxonomy (NCT).

1st step: x=5.5 is assumed to belong to class *A*.
Centroid(class A) = (-1 + 1 + 5.5)/3 = 1.83.
Centroid(class B) = (4 + 6)/2 = 5.
Table 1 summarizes the possible categories of the elements taking into account that the NCT implies that the category of an element is equal to the label of the nearest centroid.

Table 1: categories with the assumption (5.5, *A*).

Element	Nearest centroid	Category
(-1, *A*)	(1,83, *A*)	τ_A
(1, *A*)	(1,83, *A*)	τ_A
(4, *B*)	(5, *B*)	τ_B
(6, *B*)	(5, *B*)	τ_B
(5.5, *A*)	(5, *B*)	τ_B

Because the point to classify belongs to category τ_B, τ_B={(4, *B*), (6, *B*), (5.5, *A*)}, the first row of matrix P_M is $P^A(A) = 1/3$, $P^A(B) = 2/3$.

2nd step: x=5.5 is assumed to belong to class *B*.
Centroid(class A) = (-1 + 1)/2 = 0.
Centroid(class B) = (4 + 6 + 5.5)/3 = 5.17.
Table 2 summarizes the category of the elements.

Table 2: categories with the assumption (5.5, *B*).

Element	Nearest centroid	Category
(-1, *A*)	(0, *A*)	τ_A
(1, *A*)	(0, *A*)	τ_A
(4, *B*)	(5.17, *B*)	τ_B
(6, *B*)	(5.17, *B*)	τ_B
(5.5, *B*)	(5.17, *B*)	τ_B

Therefore, the second row of P_M is $P^B(A) = 0/3 = 0$ and $P^B(B) = 3/3 = 1$. In this way

$$P_M = \begin{pmatrix} 1/3 & 2/3 \\ 0 & 1 \end{pmatrix}$$

As the second column has a larger mean value, the Venn predictor predicts label *B* for x = 5.5 with a probability interval [2/3, 1].

3. General architecture of the disruption predictor

The high level predictor algorithm is described in figure 2. The block 'operation start' defines the starting point to collect data to develop the adaptive predictor. All signals required are stored from that moment until having one disruptive discharge and at least one non-disruptive discharge.

```
operation start
signal storage
wait for{
    one disruptive discharge
    one non-disruptive discharge
} end 'wait for'
first model creation{
    compute normalization factors
    signal normalization
    disruptive example
    non-disruptive example
} end 'first model creation'
wait for discharge {
    real-time prediction with last model created
    signal storage
    if missed alarm{
        new model creation {
            recompute normalization factors
            signal normalization
            add disruptive example
            add non-disruptive example
        } end 'new model creation'
    } end 'if missed alarm'
} end 'wait for discharge'
```

Fig. 2: predictor pseudo-code.

In the 'first model creation' block, a previous step is to normalise the signals to avoid a larger weight of quantities with larger absolute values (for instance, the plasma current in JET is $O(10^6)$ and the locked mode is $O(10^{-4})$). Quantities are normalised according to

$$Q_{norm}(t) = \frac{Q(t) - \min(Q)_{\text{from 'operationstart'}}}{\max(Q)_{\text{from 'operationstart'}} - \min(Q)_{\text{from 'operationstart'}}}$$

where max(·) and min(·) are the maximum and minimum values respectively. In this way, the variation range of the signals is [0, 1].

At this point, it is important to describe the feature vectors of the predictor. According to the best results obtained in [2], three signals are used: plasma current (Ip), locked mode (LM) and internal inductance (LI). However, each feature vector, **x**, will have 4 components, two of them from the time domain of the signals and the others from the frequency domain. The signals are digitised at 1 ksample/s and are analysed in time windows 32 ms long:

$$\mathbf{x} = \left(std\left(|fft(Ip)|_{+}^{*} \right), mean(LM), \right.$$
$$\left. std\left(|fft(LM)|_{+}^{*} \right), mean(LI) \right) \quad (1)$$

where mean(·)is the mean value during the time window, fft(·) is the Fourier transform, |·| is absolute value, + means that only the positive frequency components are taken into account, * means that the DC component is removed and std(·) is the standard deviation. The explanation about this selection of features can be found in [2].

As mentioned, the first model to make predictions is made up of 1 non-disruptive example and 1 disruptive example. The non-disruptive example is a feature vector that tries to condense the non-disruptive behaviour of all non-disruptive discharges from the 'operation start'. To this end, each non-disruptive discharge is divided into time windows 32 ms long between the time instants in which Ip is above the threshold of 750 kA. The components of the non-disruptive feature vector are the mean values of the components of equation (1) determined from all time windows 32 ms long of all non-disruptive discharges. The example of class disruptive is determined from equation (1) with the closest time window 32 ms long to the disruption time.

Once the first model has been created, a loop starts to synchronise the predictor operation with the production of discharges. During a running discharge, the predictor is enabled whenever the plasma current is above the threshold of 750 kA and feature vectors are generated every 32 ms. Each feature vector is classified as 'disruptive' (and an alarm is triggered) or 'non-disruptive'. This prediction is qualified with a probability interval that establishes the reliability of the classification.

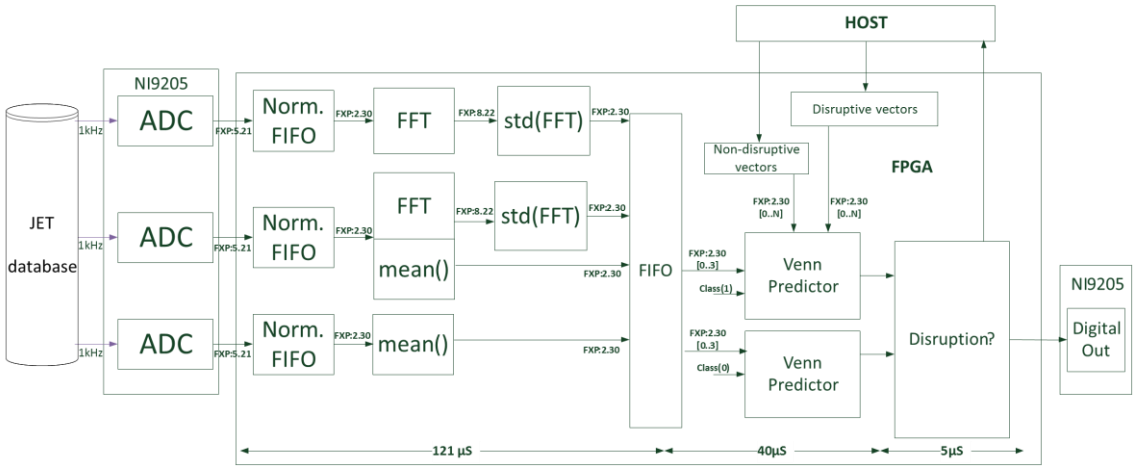
If a disruption happens and no alarm has been triggered, it means that the predictor needs to incorporate additional knowledge to be able to deal with similar disruptions in the future. Therefore, in this case, a new training is needed.

The objective of this re-training is to include one additional example of each class (disruptive and non-disruptive) to the preceding training set. Previously, new normalization factors are computed to take into account possible changes in the maximum and minimum values of the signals.

The disruptive example is created in the same way that the one of the first model. The non-disruptive example is computed following the same procedure of the first model, with all non-disruptive discharges from the last training process.

4. Predictor implementation

Fig. 3 depicts the architecture of the solution implemented with compact RIO (cRIO) technology focusing the detail on the implementation in the FPGA device [5].



The cRIO system uses only one module containing the three analog inputs required and the digital output signalling that a disruption has been predicted. Apart from the three analog signals, the FPGA needs as inputs the vectors with the disruptive and non – disruptive discharges. These vectors are provided by the HOST codified in fixed-point format. The FPGA computes the FFT and the mean to obtain the feature vectors and executes the Venn-predictor generating the alarm in the output signal. FPGA needs less than 80% of its resources and completes the computation in 166 µs. The interaction between the HOST and the FPGA device is implemented using the ITER CODAC standard software layers for these FPGA-based devices (IRIO) [6]. Every time a missed alarm is detected, the HOST application re-calculates the new model that is fed to the FPGA. All used devices correspond to ITER catalogue [7, 8].

Acknowledgements

This work has been carried out within the framework of the EUROfusion Consortium and has received funding from the Euratom research and training programme 2014-2018 under grant agreement No 633053. The views and opinions expressed herein do not necessarily reflect those of the European Commission.

This work was partially funded by the Spanish Ministry of Economy and Competitiveness under the Projects No. ENE2015-64914-C3-1-R/2-R/3-R.

References

[1] J. Vega, S. Dormido-Canto, J. M. López, A. Murari, J. M. Ramírez, R. Moreno, M. Ruiz, D. Alves, R. Felton and JET-EFDA Contributors. “Results of the JET real-time disruption predictor in the ITER-like wall campaigns”. Fusion Engineering and Design 88 (2013) 1228-1231.

[2] J. Vega, A. Murari, S. Dormido-Canto, R. Moreno, A.

Pereira, A. Acero and JET-EFDA Contributors. “Adaptive high learning rate probabilistic disruption predictors from scratch for the next generation of tokamaks”. Nuclear Fusion. 54 (2014) 123001 (17pp).
[3] V. Vovk, A. Gammerman, G. Shafer. “Algorithmic learning in a random world”. Springer (2005).
[4] P. Makijarvi. ITER catalog of I&C products – Fast Controllers. ITER 2016.
[5] M. Ruiz et al. “ITER Fast Plant System Controller prototype based on PXIe platform”. Fusion Engineering and Design, 87 (2012) pp-pp.
[6] M. Ruiz, et all. “IRIO technology: Developing applications for advanced DAQ systems using FPGAs”. Real Time Conference IEEE-NPS-2016.
[7] E. Barrera et al. “Implementation of ITER fast plant interlock system using FPGAs with cRIO”. Real Time Conference IEEE-NPS-2016.
[8] M. Ruiz et all. “Real Time Plasma Disruptions Detection in JET Implemented With the ITMS Platform” Using FPGA Based IDAQ. IEEE Transactions on Nuclear Science (Volume: 58, Issue: 4, Aug. 2011)