19th International Conference on Renewable Energies and Power Quality (ICREPQ'21)
Almeria (Spain), 28th to 30th July 2021
Renewable Energy and Power Quality Journal (RE&PQJ)
ISSN 2172-038 X, Volume No. 19, September 2021

# Wind Turbine Multi-Fault Detection based on SCADA Data via an AutoEncoder

Á. Encalada-Dávila[1], C. Tutivén[1], B. Puruncajas[1] and Y. Vidal[2,3]

[1] Mechatronics Engineering
Faculty of Mechanical Engineering and Production Science, FIMCP
Escuela Superior Politécnica del Litoral, ESPOL
Campus Gustavo Galindo Km. 30.5 Vía Perimetral, P.O. Box 09-01-5863, Guayaquil, Ecuador
Phone number: +593 9 9103 5259, e-mail: **angaenca@espol.edu.ec, cjtutive@espol.edu.ec, bpurunca@espol.edu.ec**

[2] Control, Modeling, Identification and Applications, CoDAlab
Department of Mathematics, Escola d'Enginyeria de Barcelona Est, EEBE
Universitat Politècnica de Catalunya, UPC
Campus Diagonal-Besós (CDB) 08019, Barcelona, Spain

[3] Institut de Matemàtiques de la UPC - BarcelonaTech, IMTech
Pau Gargallo 14, 08028 Barcelona, Spain
Phone number: +34 934 137 309, e-mail: **yolanda.vidal@upc.edu**

**Abstract.** Nowadays, wind turbine fault detection strategies are settled as a meaningful pipeline to achieve required levels of efficiency, availability, and reliability, considering there is an increasing installation of this kind of machinery, both in onshore and offshore configuration. In this work, it has been applied a strategy that makes use of SCADA data with an increased sampling rate. The employed wind turbine in this study is based on an advanced benchmark, established by the National Renewable Energy Laboratory (NREL) of USA. Different types of faults on several actuators and sensed by certain installed sensors have been studied. The proposed strategy is based on a normality model by means of an autoencoder. As of this, faulty data are used for testing from which prediction errors were computed to detect if those raise a fault alert according to a defined metric which establishes a threshold on which a wind turbine works securely. The obtained results determine that the proposed strategy is successful since the model detects the considered three types of faults. Finally, even when prediction errors are small, the model is able to detect the faults without problems.

## Key words

Wind Turbine, Multi-Fault Detection, SCADA Data, AutoEncoder, Normality Model

## 1. Introduction

Wind energy is one of the alternative energy sources, which has a huge acceptance around the world, since the growth of installation of wind turbine (WT) farms is more frequent along last years. In order to produce a lower environmental impact there is an strength attraction towards renewable energy. The annual report presented by the Global Wind Energy Council (GWEC) showed that by the end of 2019 the installed wind power capacity (WPC) has reached 650 GW, which represents a gain of 10% if it is compared with 2018 [1]. By the end of 2020 the report showed that installed WPC was in 840 GW and for this 2021 it estimates that there will be an installed WPC around 800 GW. If the efforts for wind industry continue increasing along time, the technology will also continue improving and growing, i.e. that the costs of installing wind projects will decrease [2], [3].

These costs can easily overcome approximately the 20% of the total energy generation cost for wind power plants (WPPs) [4]. When a WT has a lot downtime, in efficiency it means that productivity decreases while the operation and maintenance (O&M) costs increase [5], [6]. Condition monitoring systems (CMSs) appear as a solution for this problem, since with this strategy it is possible to know the current state of a component and provide an indication of fault or non-fault [7]. For offshore and onshore WTs condition monitoring is applicable, even considering that for offshore environment weather conditions could be worse, like storms, rays, high waves, high tides, etc. Since providing timely maintenance and preventing failures reduce the O&M costs and the downtime of affected WTs [8], early fault detection has become a meaningful strategy to increase the competitiveness and productivity of WTs.

Fault detection systems (FDSs) are used to activate alarms when a component or system experiences an abnormal behaviour within its operation. The obtained information to provide this kind of analysis comes from different kind of sensors (thermal, electric, inductive, capacitive, etc) installed in WTs. WT components such as gearbox, electric system, generator and breaking system are the most prone to fault [9]. Nowadays, several strategies focus its analysis in specific parts of WTs to monitor its state in real time and rise

up alarms when a fault is presented, however, it involves huge costs. Against to this, it is important to highlight that currently all WTs installed are integrated with SCADA systems which monitor the main subsystems or components collecting data related to temperatures of bearings, winding and lubricating oil, as well as vibration levels of the main drive train [10]. Thus, with all these collected data through SCADA systems it is possible to provide a diagnostic about the WT condition [11].

Along time with the development of Machine Learning (ML) and Artificial Intelligence (AI) several studies using only SCADA data have been carried out to build models which help to detect and/or classify faults associated to WTs. For example, a FD model for WT gearboxes by using artificial neural networks (ANNs) is proposed by researchers in Italy, where they used only real SCADA data [12]. In China, by using strategies of Principal Component Analysis (PCA) a FD and diagnosis framework for the WT generator is proposed, using real SCADA datasets of two WPPs located in that country [13]. Other algorithms like SVM, XGBoost and Random Forest have been also used to develop FD and classification models employing SCADA data, as it is described in [14].

One the major drawbacks using SCADA data is that sampling rate is averaged values each 10 min. This low frequency is a disadvantage, since the diagnosis capabilities decrease and a lot of short-lived events maybe overlooked. Thus, a high and feasible resolution in SCADA data should allow to identify the FD with a higher fidelity. In this paper the work is developed using SCADA data, which have been taken with a frequency of 1 Hz [15]. Since this, a strategy is proposed to do multi-fault detection by using an autoencoder (AE), which is a characteristic architecture of neural networks (NNs). The methodology is based on a normality model, i.e. training the AE only with healthy data and then testing it with healthy and faulty data. Applying a defined metric it is possible to establish a threshold which puts a barrier between healthy and faulty samples.

The rest of the paper is organized by sections as follows: Section 2 describes all about collected and used data in this study. Next, Section 3 talks about the multi-fault detection model (MFDm) and the methodology employed to carry out the proposed strategy. Section 4 contains the results and its respective discussion. Finally, the conclusions and future work are given in Section 5.

## 2. Data Description

### *A. FAST WT Model Description*

The simulations are performed in a 5 MW WT model by using the aeroelastic WT simulator software called FAST, which was designed by NREL's National Wind Technology Center of USA. In [16] there are more details about this WT and its characteristics. Due to electrical noise it was important to include noise blocks in order to represent somehow this phenomenon. FAST simulation software uses a characteristic sampling period, which is set in 0.0125 s. Nevertheless, the used data in this work were sampled with a sampling rate of 1 s. For this WT there are several sensors which collect data into SCADA system and are shown in Table I.

Several fault scenarios have been include in the WT model. These scenarios are related to gain factors, offsets, changes in the system dynamics, stuck, from which three types of faults of interest were chosen, as shown in Table II. These faults have its origin in other sources which are of public domain [17]. Additionally, the interested reader can find a detailed description about these faults in [18].

In the other hand, a turbulent-wind simulator, TurbSim -developed by NREL-, was used also in these simulations since this is a useful tool to do wind modelling in a realistic way. TurbSim simulations were performed setting the following wind parameters: Kaimal turbulence model with a intensity of 10 % at hub height, wind moving logarithmically at an average speed of 18.2 m/s and a roughness factor of 0.01 m.

### *B. Data Collection*

A set of 260 simulations of 60 s each one were performed, establishing one entire dataset per simulation. Of these 260 datasets it is important to highlight that 100 correspond to healthy simulations and the rest corresponds to faulty simulations, where 20 simulations are related to each type of fault (originally there are eight types of faults) from which three types of faults were selected. Each simulation contains 600 s of collected data, however, the first 200 s are related to transient state [19] while the resting 400 s are associated with a stable state, which are considered as the useful data.

As it was before-mentioned the original sampling rate of simulations is 0.0125 s, however, the data have been down-sampled with a sampling rate of 1 s. In [20] is proposed a strategy to use SCADA data with a higher frequency than 10 min, i.e. for example 1 s. Thus, following this pipeline the data were down-sampled to 1 s.

### *C. Data Split: Train, Validation and Test*

In this work one of the first activities is data split, in which data are distributed into train, validation and test sets. Thus, as in this paper a normality model is proposed, data split is applied only for healthy data, which represent about 100 simulations and these are distributed as follows: 70 % for train, 20 % for validation and 10 % for test. Recall that in this case it is talking about simulations and not samples, since each simulation has 400 samples.

Together with test set is joined faulty data, since to test the model a few samples of healthy data (10 % of healthy data) and all samples corresponding to faulty simulations -which sum around 20 subsets by type of fault- are needed.

### *D. Data Standardization*

Almost all times the data are standardize in order to deal with sets of values that come from different sources and with different magnitudes, as well as in this case that there are several sensors which measure different magnitudes. Before

Table I. - Description of the several available sensors in the WT.

| ID | Sensor Type | Symbol | Unit | Noise Power |
|---|---|---|---|---|
| S1 | Generated electrical power | $P_{e,m}$ | W | $1.0 \times 10^{1}$ |
| S2 | Rotor speed | $\omega_{r,m}$ | rad/s | $1.0 \times 10^{-4}$ |
| S3 | Generator speed | $\omega_{g,m}$ | rad/s | $2.0 \times 10^{-4}$ |
| S4 | Generator torque | $\tau_{c,m}$ | Nm | $9.0 \times 10^{-1}$ |
| S5 | Pitch angle of first blade | $\beta_{1,m}$ | deg | $1.5 \times 10^{-3}$ |
| S6 | Pitch angle of second blade | $\beta_{2,m}$ | deg | $1.5 \times 10^{-3}$ |
| S7 | Pitch angle of third blade | $\beta_{3,m}$ | deg | $1.5 \times 10^{-3}$ |
| S8 | Tower top fore-aft acceleration | $\alpha_{fa,m}$ | m/s$^2$ | $5.0 \times 10^{-4}$ |
| S9 | Tower top side-to-side acceleration | $\alpha_{ss,m}$ | m/s$^2$ | $5.0 \times 10^{-4}$ |

Table II. - Details of defined faults in the WT during the simulations.

| ID | Fault | Type |
|---|---|---|
| F1 | Pitch actuator - Hydraulic leakage | Change in system dynamics |
| F2 | Generator speed sensor | Gain factor (1.2) |
| F3 | Torque actuator | Offset value (2000 Nm) |

to continue with data preprocessing it is necessary to apply this strategy. In this work Min-Max scaler [21] is used, which scales the data in a range from 0 to 1. This scaling strategy many times is called also normalization. Thus, Min-Max scaler is defined as:

$$\widehat{x}_{i,j}^{(k)} = \frac{x_{i,j}^{(k)} - x_{min,j}^{(k)}}{x_{max,j}^{(k)} - x_{min,j}^{(k)}} \tag{1}$$

where $\widehat{x}_{i,j}^{(k)}$ is the scaled value, $x_{i,j}^{(k)}$ is the value to be scaled, $x_{max,j}^{(k)}$ is the maximum value of a certain column while $x_{min,j}^{(k)}$ is the minimum value of that column.

### *E. Data Reshaping*

Minimizing the detection time while preserving overall accuracy is the main aim, using all available SCADA data. The smaller the required sample, the smaller the detection time ($T_d$), since less time is required to collect data from the sensors. $T_d$ is the elapsed time between the fault occurrence and its detection. Thus, the fault detection requirements given in the model [17] are described in terms of the sampling time ($T_s$) which is equal to 1 s.

- F1 is related to the pitch actuator where faults have a very slow dynamic. Thus, the condition that must be fulfilled is that $T_d < 100T_s$.
- F2 is the related to the generator speed sensor and the pitch sensors, so the condition that must be fulfilled is that $T_d < 10T_s$.
- F3 is related to the torque actuator, so it has a restrictive $T_d$, in which $T_d < 3T_s$ is the condition that must be fulfilled [18].

Since this, the two most restrictive requirements are chosen, whereby the data are organized in samples of only $J = 3$ and $J = 10$ time steps. Thus, in Eq. 2 the representation of data for each sensor in a matrix is showed.

$$\begin{pmatrix} x_{1,1}^{(k)} & x_{1,2}^{(k)} & \cdots & x_{1,\lfloor \frac{400}{J} \rfloor J}^{(k)} \\ x_{2,1}^{(k)} & x_{2,2}^{(k)} & \cdots & x_{2,\lfloor \frac{400}{J} \rfloor J}^{(k)} \\ \vdots & \vdots & \ddots & \vdots \\ x_{260,1}^{(k)} & x_{260,2}^{(k)} & \cdots & x_{260,\lfloor \frac{400}{J} \rfloor J}^{(k)} \end{pmatrix} \tag{2}$$

where $k$ represents the sensor, hence $k \in \{1, 2, 3, ..., 9\}$.

Likewise, there are 160 rows since there are 160 simulations and $\lfloor \frac{400}{J} \rfloor J$ columns, where $J$ represents the defined time steps. Next applying the respective data reshaping, Eq. 3 shows the reshape of the matrix (Eq. 2) in terms of $J$ also, which represents the time steps with each sensor-values vector has been split. Thus, in Fig. 1 the complete data distribution is showed after data reshaping, considering the defined $J$ time steps and all include sensors.

$$\begin{pmatrix} x_{1,1}^{(k)} & x_{1,2}^{(k)} & \cdots & x_{1,J}^{(k)} \\ x_{1,J+1}^{(k)} & x_{1,J+2}^{(k)} & \cdots & x_{1,2J}^{(k)} \\ \vdots & \vdots & \ddots & \vdots \\ x_{1,400-J+1}^{(k)} & x_{1,400-J+2}^{(k)} & \cdots & x_{1,\lfloor \frac{400}{J} \rfloor J}^{(k)} \\ x_{2,1}^{(k)} & x_{2,2}^{(k)} & \cdots & x_{2,J}^{(k)} \\ x_{2,J+1}^{(k)} & x_{2,J+2}^{(k)} & \cdots & x_{2,2J}^{(k)} \\ \vdots & \vdots & \ddots & \vdots \\ x_{2,400-J+1}^{(k)} & x_{2,400-J+2}^{(k)} & \cdots & x_{2,\lfloor \frac{400}{J} \rfloor J}^{(k)} \\ \vdots & \vdots & \ddots & \vdots \\ x_{260,1}^{(k)} & x_{260,2}^{(k)} & \cdots & x_{260,J}^{(k)} \\ x_{260,J+1}^{(k)} & x_{260,J+2}^{(k)} & \cdots & x_{260,2J}^{(k)} \\ \vdots & \vdots & \ddots & \vdots \\ x_{260,400-J+1}^{(k)} & x_{260,400-J+2}^{(k)} & \cdots & x_{260,\lfloor \frac{400}{J} \rfloor J}^{(k)} \end{pmatrix} \tag{3}$$

In a nutshell, the initial matrix has been rewrote as a matrix with X columns and Y rows, where X is equal to $J$ multiplied

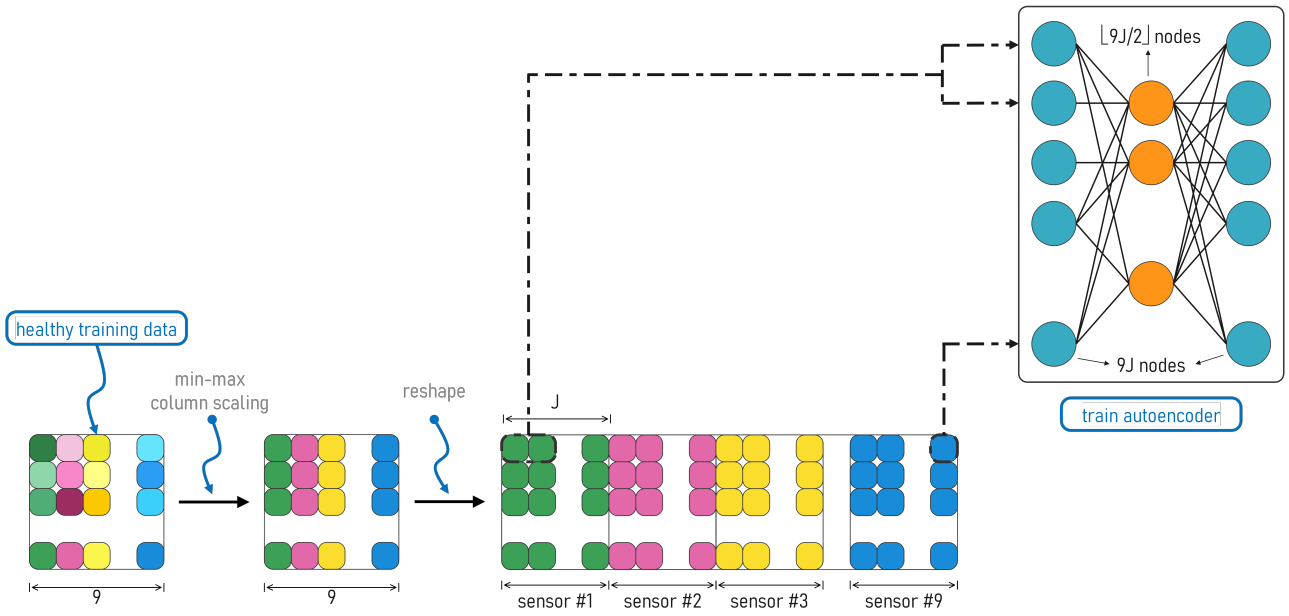


Fig. 1: Data preprocessing stages from the original data to training stage of the AE.

by the number of sensors and Y is equal to the number of simulations multiplied by the number of splits of each row, obtaining the following results:

- A matrix with 21280 samples (or rows) and 27 columns, when $J = 3$.
- A matrix with 6400 samples (or rows) when 90 columns, when $J = 10$.

## 3. Multi-Fault Detection Model

### *A. AutoEnconder Architecture for MFDm*

An AE is an architecture that has three sections clearly defined: an encoder, a feature vector, and a decoder. The main aim of an AE is to build the best feature vector, which contains the best of the entered information at the input. Thus, the encoder maps the information from the input to the feature vector, while the decoder rebuilds the input in the output by using the stored information in the feature vector [22].

For this case, an AE is trained to rebuild the input at the output, using only healthy data. Here is where the normality model [23] works, since by contrast, if a faulty sample is entered, the AE will not rebuild the input well, thus, the residual error will help to detect a fault since it will be used as an indicator.

In the other hand, sometimes a good AE is not the one that has many hidden layers, since an AE with a few hidden layers maybe better. Thus, in this case, the architecture of the AE is set as the input layer has a length equal to the input vector length, which depends of the used $J$ time steps. The hidden layer has the middle of nodes of the input and finally the output layer has the same number of nodes like the input, as shown in Fig. 1. Likewise, Table III summarizes the configured hyperparameters for the AE architectures.

Table III. - Hyperparameters configuration for AE training respect to $J$ time steps.

| Hyperparameter | J=3 | J=10 |
|---|---|---|
| Epochs | 500 | 500 |
| Number of Layers | 3 | 3 |
| Nodes in Input Layer | 27 | 90 |
| Nodes in Hidden Layer | 14 | 45 |
| Nodes in Output Layer | 27 | 90 |
| Loss Function | MSE | MSE |
| Activation Function | ELU | ELU |
| Optimizer | Adam | Adam |
| Learning Rate | 0.001 | 0.001 |
| $\beta_1$ - $\beta_2$ | 0.9 - 0.999 | 0.9 - 0.999 |
| $\epsilon$ | $1 \times 10^{-8}$ | $1 \times 10^{-8}$ |
| Weight Decay | 0 | 0 |

### *B. Error Computation Strategy for Predictions*

As it was before-mentioned, an AE receives an input and tries to rebuild it at the output. If this is extrapolated to all tested samples, finally a predictions matrix is obtained, as shown in Fig. 2. Thus, basically an error matrix can be defined also as of the subtraction between the input and output matrix, as shown in Fig. 2.

It is necessary to reduce the errors by rows to a one representative error, since it will correspond to an error value by tested sample. To achieve it root mean square error (RMSE) is applied to compute the single error by rows. Eq. 4 shows the representation of RMSE for this case.

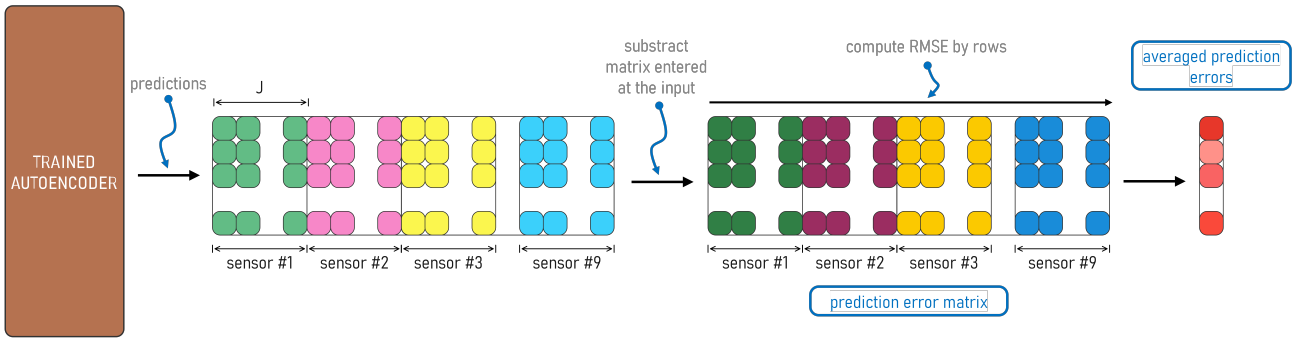


Fig. 2: Steps to compute the prediction-error matrix considering the computation of RMSE by rows or samples.

$$RMSE = \frac{1}{G} \cdot \sum_{j=0}^{G} E_j^2 \tag{4}$$

where $G = 9J$, since this represents the number of columns and $j$ is the iterator along rows from 0 to $G$. Finally, the result is a vertical vector which contains a RMSE value by sample.

### C. Moving Average applied to Prediction Errors

The concept of moving average (MA) comes from statistics and it refers to a calculation process used to analyze data points, creating a series of averages of several subsets of a full dataset. The main aim of the MA is to help to smooth out the short-term fluctuations and impacts of random values into the original curves, over a defined time-frame.

In the used dataset for this work, it is clear that several variables describe a sinusoidal behaviour deriving from the set parameters in the simulations. MA is usually computed in order to identify the trend direction of a curve, steady values, etc [24]. It is necessary to consider also that the longer the time period for the MA, the greater the lag, thus, these parameters must be handled with criterion. Eq. 5 shows the representation of MA as follows:

$$MA_L = \frac{1}{L} \cdot \sum_{i=1}^{L} H_{t-i} = \frac{H_{t-1} + H_{t-2} + ... + H_{t-L}}{L} \tag{5}$$

where $L$ is the lag used in the MA and $H_{t-i}$ is the data point $H$ at time $t - i$ where $i \in \{1, 2, 3, ..., L\}$.

### D. Fault Detection Metric based on Prediction Errors

Over the error vector computed in Subsection 3-B the detailed strategy in Subsection 3-C is applied. In this way it is obtained the prediction errors based on the MA. Recall that the metric is based on training and validation data, since these are samples which have been seen by the AE during the training and validation steps. Also remember that these data correspond to just healthy samples.

Thus, based on these errors a fault detection threshold, $t_{FD}$, [23] is computed, which defines when the samples are considered as faulty or healthy. To calculate the $t_{FD}$, the mean and standard deviation are used. Eq. 6 shows the representation of this metric:

$$t_{FD} = \mu \pm 3\sigma \tag{6}$$

where $\mu$ represents the mean of the values vector and $\sigma$ describes the standard deviation computed over that vector.

## 4. Results and Discussion

As previously mentioned, the test set contains both, healthy and faulty samples. Entering this bundle of samples to the AE the predictions are obtained and comparing them to the real input the prediction errors are computed. Then, applying the mentioned strategies in Subsections 3-B and 3-C a processed output as of the initial prediction error matrix is obtained.

Using the prediction errors of train and validation sets the maximum and minimum values for the $t_{FD}$ are calculated also, with which the range on which a tested healthy-sample error must be, is defined. In this way, for the three types of faults, MFD is clearly defined, as shown in Fig. 3 and 4. In those figures it is clear that the average prediction errors for healthy data are correctly located inside the threshold. In the other hand, F1 is closer to the threshold zone, however this type of fault is detected well. Likewise, F2 and F3 are correctly detected and even there is a considerable distance with respect to the $t_{FD}$ zone.

At a glance it is notable that the curve for healthy points is shorter than the curve for the other faults. This is due to two main reasons. First one, when the MA is applied it reduces the amount of samples in function of the used lags, which were set in 180 and 600, for $J = 10$ and $J = 3$, respectively. This means that the lag uses a temporal window of 30 min, recalling that each sample in the SCADA system is taken per each second. The second one is that few data, exactly 10 %, are used in the test set. However, this does not interfere on the effectiveness of the proposed strategy to do MFD.

## 5. Conclusions and Future Work

As mentioned earlier, an important aim at the moment of using SCADA data is to reduce the $T_d$. In the results it is noticeable that either for $J = 3$ and $J = 10$ the FD

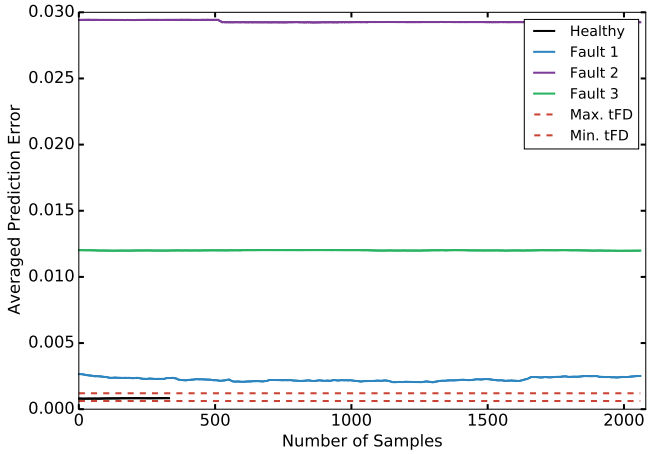


Fig. 3: Averaged prediction errors when the data are reshaped with $J = 3$ time steps.

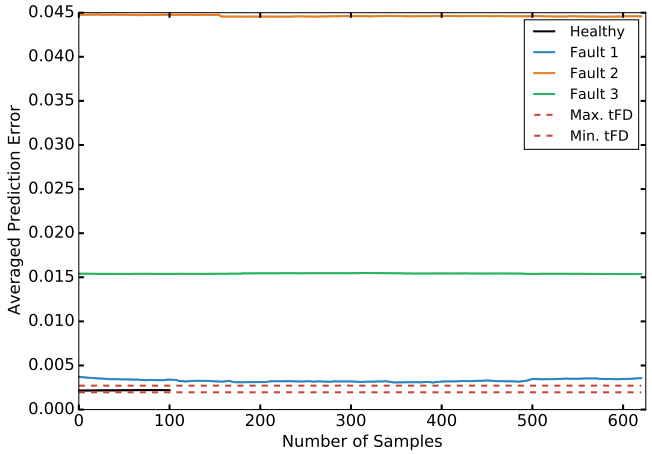


Fig. 4: Averaged prediction errors when the data are reshaped with $J = 10$ time steps.

achieves excellent results. However, for the first value of $J$ the obtained errors are lower, attaining a better performance.

The combination between an AE and a normality model shows that a high performance can be reached, as in this study is seen. Another point to highlight is that the down-sampling strategy is successful in order to fit the data as if a SCADA system has generated them. It allowed also to obtain excellent results in the FD process.

As future work it will be interesting to work on assembled architectures, i.e. for example using a basic multilayer perceptron (MLP) to do fault classification on the computed outputs. Likewise, it will be interesting to include data augmentation techniques in order to apply other deep leaning strategies by using images and convolutional neural networks (CNNs).

## References


[1] W. E. C. G. GLOBAL, Gwec global wind report 2018 (2019).

[2] R. Pandit, D. Infield, Gaussian process operational curves for wind turbine condition monitoring, Energies 11 (7) (2018) 1631.

[3] L. Zhang, K. Liu, Y. Wang, Z. B. Omariba, Ice detection model of wind turbine blades based on random forest classifier, Energies 11 (10) (2018) 2548.

[4] Y. Eroğlu, S. U. Seçkiner, Early fault prediction of a wind turbine using a novel ann training algorithm based on ant colony optimization (2019).

[5] P. Tavner, J. Xiang, F. Spinato, Reliability analysis for wind turbines, Wind Energy: An International Journal for Progress and Applications in Wind Power Conversion Technology 10 (1) (2007) 1–18.

[6] C. McKinnon, A. Turnbull, S. Koukoura, J. Carroll, A. McDonald, Effect of time history on normal behaviour modelling using scada data to predict wind turbine failures, Energies 13 (18) (2020) 4745.

[7] W. Yang, P. J. Tavner, C. J. Crabtree, Y. Feng, Y. Qiu, Wind turbine condition monitoring: technical and commercial challenges, Wind Energy 17 (5) (2014) 673–693.

[8] F. P. G. Márquez, A. M. Tobias, J. M. P. Pérez, M. Papaelias, Condition monitoring of wind turbines: Techniques and methods, Renewable Energy 46 (2012) 169–178.

[9] J. Maldonado-Correa, S. Martín-Martínez, E. Artigao, E. Gómez-Lázaro, Using scada data for wind turbine condition monitoring: A systematic literature review, Energies 13 (12) (2020) 3132.

[10] E. Becker, P. Poste, Keeping the blades turning: condition monitoring of wind turbine gears, Refocus 7 (2) (2006) 26–32.

[11] Z.-Y. Zhang, K.-S. Wang, Wind turbine fault detection based on scada data analysis using ann, Advances in Manufacturing 2 (1) (2014) 70–78.

[12] D. Astolfi, F. Castellani, L. Scappaticci, L. Terzi, Diagnosis of wind turbine misalignment through SCADA Data, Diagnostyka 18 (1) (2017) 17–24.

[13] Y. Zhao, D. Li, A. Dong, D. Kang, Q. Lv, L. Shang, Fault prediction and diagnosis of wind turbine generators using SCADA data, Energies 10 (8) (2017) 1–17. doi:10.3390/en10081210.

[14] K. Leahy, C. V. Gallagher, K. Bruton, P. O'Donovan, D. T. O'Sullivan, Automatically identifying and predicting unplanned wind turbine stoppages using scada and alarms system data: Case study and results, in: Journal of Physics: Conference Series, IOP Publishing, 2017, pp. 1–14.

[15] K.-S. Wang, V. S. Sharma, Z.-Y. Zhang, Scada data based condition monitoring of wind turbines, Advances in Manufacturing 2 (1) (2014) 61–69.

[16] Y. Vidal, F. Pozo, C. Tutivén, Wind turbine multi-fault detection and classification based on scada data, Energies 11 (11) (2018) 3018.

[17] P. Odgaard, K. Johnson, Wind turbine fault diagnosis and fault tolerant control-an enhanced benchmark challenge, in: Proc. of the 2013 American Control Conference–ACC,(Washington DC, USA), 2013, pp. 1–6.

[18] M. Ruiz, L. E. Mujica, S. Alférez, L. Acho, C. Tutivén, Y. Vidal, J. Rodellar, F. Pozo, Wind turbine fault detection and classification by means of image texture analysis, Mechanical Systems and Signal Processing 107 (2018) 149–167. doi:https://doi.org/10.1016/j.ymssp.2017.12.035.

[19] M. A. Lackner, M. A. Rotea, Passive structural control of offshore wind turbines, Wind Energy 14 (3) (2011) 373–388. doi:10.1002/we.426.
URL http://doi.wiley.com/10.1002/we.426

[20] E. Gonzalez, B. Stephen, D. Infield, J. J. Melero, On the use of high-frequency SCADA data for improved wind turbine performance monitoring, Journal of Physics: Conference Series 926 (2017) 012009. doi:10.1088/1742-6596/926/1/012009.

[21] C. wei Zheng, Z. niu Xiao, Y. hua Peng, C. yin Li, Z. bo Du, Rezoning global offshore wind energy resources, Renewable Energy 129 (2018) 1–11. doi:https://doi.org/10.1016/j.renene.2018.05.090.

[22] M. Tschannen, O. Bachem, M. Lucic, Recent advances in autoencoder-based representation learning, CoRR abs/1812.05069 (2018). arXiv:1812.05069.
URL http://arxiv.org/abs/1812.05069

[23] Encalada-Dávila, B. Puruncajas, C. Tutivén, Y. Vidal, Wind turbine main bearing fault prognosis based solely on scada data, Sensors 21 (6) (2021). doi:10.3390/s21062228.
URL https://www.mdpi.com/1424-8220/21/6/2228

[24] D. Song, J. Yang, Y. Liu, M. Su, A. Liu, Y. H. Joo, Wind direction prediction for yaw control of wind turbines, International Journal of Control, Automation and Systems 15 (4) (2017) 1720–1728. doi:10.1007/s12555-017-0289-6.