Electrical Power and Energy Systems 136 (2022) 107594

Contents lists available at ScienceDirect

International Journal of Electrical Power and Energy Systems

journal homepage: www.elsevier.com/locate/ijepes

Performance analysis of a DNN classifier for power system events using an interpretability method

Orlem L.D. Santos [a,*], Daniel Dotta [a], Meng Wang [b], Joe H. Chow [b], Ildemar C. Decker [c]

[a] University of Campinas - UNICAMP, 13083-852 Campinas, SP, Brazil
[b] Rensselaer Polytechnic Institute, Troy, NY 12180, USA
[c] Federal University of Santa Catarina, Florianopolis, SC, Brazil

ARTICLE INFO

Keywords:
Interpretability
SHAP values
PMU
WAMS
LSTM

ABSTRACT

Nowadays, there is a clear need for Machine Learning methods capable of extracting relevant and reliable information from synchrophasor data. In this paper, the application of an explainable data-driven method is carried out in order to inspect the performance of DNN classifier for event identification using synchrophasor measurements. The DNN classifier is the Long–Short Term Memory (LSTM) which is suitable for the extraction of dynamic features. The key advantage of this approach is the use of an interpretability inspection named SHAP (SHapley Additive exPlanation) values, based on cooperative game theory (Shapley values), which provide the means to evaluate the predictions of the LSTM and detecting possible bias. The main contributions are stated as follows: (i) it explains how the LSTM classifier is making its decisions; (ii) it helps the designer to improve the training of the classifier; (iii) certify that the resulting classifier has a consistent and coherent performance according to domain knowledge of the problem; (iv) when the user understands that the classifier is making coherent decisions, it clearly reduces the concerns of the application of DNN methods in critical infrastructure. Additionally, the proposed approach is evaluated using real synchrophasor event records from the Brazilian Interconnected Power System (BIPS).

Acronyms

ANN Artificial Neural Network

BA Balanced Accuracy

FC Fully Connected

GT Generation Tripping

IAR Identification Accuracy Rate

LT Line Tripping

OS Oscillation

PMU Phasor Measurement Unit

RNN Recurrent Neural Network

WAMS Wide-Area Monitoring Systems

1. Introduction

Nowadays, the use of Phasor Measurement Unit (PMU) data for the power system operators is a reality and there are several commercial Wide-Area Monitoring Systems (WAMS) systems spread around the world. Thanks to the advance of the WAMS, the power system operators have direct accesses to a large amount of data bring a valuable source of information to the control centers. When compared to the traditional SCADA systems, the main advantages of PMU measurements are the synchronization and higher sampling rate making possible to observe the power system dynamics.

Therefore, there is a necessity to explore algorithms of data science, such as Artificial Neural Network (ANN), since they may allow fast and efficient extraction of significant information about the EPS. In the previous studies, the majority of the approaches have explored the application of machine learning (ML) techniques using feature extraction for event identification. In [1], the authors proposed a method for event detection and identification in real-time based on a moving window Principal Components Analysis (PCA). The authors

* Corresponding author.
E-mail address: o211501@g.unicamp.br (O.L.D. Santos).

https://doi.org/10.1016/j.ijepes.2021.107594
Received 21 December 2020; Received in revised form 2 May 2021; Accepted 6 September 2021
Available online 1 October 2021

were able to precisely detect and classify generation loss, load shedding, and islanding events using real synchrophasors data from the U.K. power system. In [2], the authors proposed an event identification method that performs a postmortem analysis on real synchrophasors data of the Indian power system. The method combines an Empirical Wavelet Transform (EWT), for feature extraction, with a random forest classifier. In [3], a real-time event detection and classification were performed using a signal energy transformation. The detection was realized with the Teager–Kaiser Energy Operator (TKEO) and the identification was noticed using an Energy Similarity Measure (ESM), for feature extraction with a 1-NN (Nearest Neighbor) classifier. The authors were able to accurately detect and classify real synchrophasors events of the Indian grid. Recently, Deep Neural Network (DNN) models can take advantage of representation learning, i.e., learning representations of the data that makes it easier to extract useful information when building classifiers or other predictors [4]. Therefore, by using this technique other representations can be learned directly from the data without the necessity of featuring engineering techniques [4]. By definition, feature engineering is the process of using domain knowledge to extract features from raw data. These features can be used to improve the performance of machine learning algorithms [5]. For example, traditional ML methods can easily create a prediction for structured data when feature engineering is done beforehand. The DNN models can extract the main features of the data without the necessity to use additional methods reducing the necessity of manual application of feature engineering. Additionally, recent advances in deep learning demonstrate that deep models, which are formed by a cascade of non-linear transformation without the assistance of any feature extraction technique can achieve higher accuracy rates [6,7].

In these DNN models, the representations are formed by the composition of multiple non-linear transformations to yield more abstract and more useful representations [4]. In a recent paper [8], the authors emphasized the potential of DNN models to solve problems in different power system areas. One of these works is [9], where the authors presented an interesting idea about application of Convolutional Neural Network (CNN) for event identification using postmortem analysis without feature extraction methods by relaying in a DNN model, and taking advantage of the representation learning. Besides, the authors in [10] proposed an identification of successive events using a CNN for classification. Also, the authors proposed a method to train the extracted dominant eigenvalues of the dynamical system and the singular values of the data matrix instead of direct measurements time-series.

One of the most relevant DNN models is the LSTM, which is a special Recurrent Neural Network (RNN), and has been performing well in the extraction of features of time series capable of solving long-term dependencies [11]. This recurrent model has shown superior performance in many research fields such as image captioning [12], NLP (Natural Language Processing) [13], involving a series of dynamic problems like text classification [14], machine translation [15], and speech recognition [16]. This superior performance in these dynamic problems is due to the recurrent nature of the LSTM and its great capability of learning time-series patterns [11]. The LSTM has been applied to many power systems solving many problems such as real-Time identification of power fluctuations [17], using the LSTM to estimate the power fluctuations from real-time frequency signal, short-term residential load forecast [18], comparing the LSTM with some state-of-the-arts models in load forecasting, wind power forecasting [19], using the LSTM in the forecasting along with a sequential correlation feature extraction, detection of non-technical losses [20], using the LSTM to detect the irregularities in power usages, and power disaggregation, based on a supervised LSTM trained to extract the power the target power signal of the appliance or sub-circuit. Regarding the event identification, the authors in [21] proposed a method for line trip fault prediction in power systems using the LSTM and a Support Vector Machines (SVM). The authors proved that Long–Short Term Memory (LSTM) is suitable for extracting the features of numerous time-series in an EPS application. In [17], the LSTM is employed for real-time identification of power fluctuations, showing that these fluctuations could be accurately identified.

In all these works, the performance evaluation is limited to the Identification Accuracy Rate (IAR) of the resulting *black-box* classifier for a specified data-set. These deep models lack the interpretability of their predictions in the sense that is not clear how and why they arrive at a particular prediction [22]. This is further aggravated by the cascade of non-linear transformations, which is becoming deeper and deeper in recent models. The lack of knowledge about the way the classifier performs the identification can raise concerns for high-risk environments (critical infrastructure) such as the EPS, and especially in event identification when controlling actions are taken after the event identification. The level of reliability must be high, or the actions could have serious consequences in the EPS. In addition, when the classifier fails the designer will not have a clear direction to improve the classifier performance. Even though the classifier is retrained and incorporated the fail events in the data-set, there is no indication that the problem is solved. This can be a clear constraint for the application to DNN methods in power systems. To overcome this problem, we have proposed to use an interpretability technique called SHAP (SHapley Additive exPlanation) values [23], which is based on the game theory method named Shapley value [24], to understand if the LSTM classifier takes its decisions in a reasonable and coherent way according to the domain knowledge of the power system events problem. To reach this goal, we have explored a deep LSTM network (without the assistance of feature extraction techniques) for the identification of events in practical power systems. The performance evaluation is carried out by the IAR and the interpretability inspection using SHAP values. This technique allows us to identify what is really important in the input, highlighting the parts of the time-series with the most contribution to the identification of each event type. Furthermore, this interpretability technique can also provide a better understanding of how a DNN can be trained. This helps the designer to improve the training process of the classifier.

The paper is organized as follows. In Section 2, we present the problem statement and how our approach is contributing. In Section 3, the data-set of Brazilian Interconnected Power System (BIPS) events records is described. In Section 4, the design of the LSTM classifier is presented. The interpretability method is presented in Section 5. In Section 7, the performance of the classifier is assessed using the real events records from BIPS. The conclusions are presented in Section 8.

2. Problem statement and our approach

In the event identification problem, the power system operators need something more than a black-box classifier that just attribute a label to an specific event such as: Generation Tripping (GT), Loading Shedding (LS), Oscillation (OS), Line Tripping (LT), and Islanding. It is also useful to know why and how the classifier is taken its decisions.

To achieve this goal, we have proposed an approach based on time-series by means of an LSTM network. For instance, we have 10 seconds frequency record time-series classified as OS event by the LSTM, and sub-divided into six time-steps $\boldsymbol{x}^{(t)}, t = 1, \ldots, 6$. The proposed approach is described in Fig. 1. Therefore, using the SHAP inspection the magnitude of the contribution $\Phi^{(t)}$ for each time step $t = 1, \ldots, 6$ can be estimated separately. Finally, by evaluating the size of the magnitudes of the contributions, a ranking for contributions is built that helps to evaluate that which ones are more important (or not) to the identification of this event.

Realizing how the event was classified is mainly important to detect possible bias, patterns that are not according to domain knowledge of the events and inconsistencies. Thus, the main focus of this work is to propose an explainable data-driven classifier, named LSTM-SHAP and presented in Fig. 2(b), that is re-trained not only observing the highest IAR but after the SHAP inspection of the LSTM classifier.

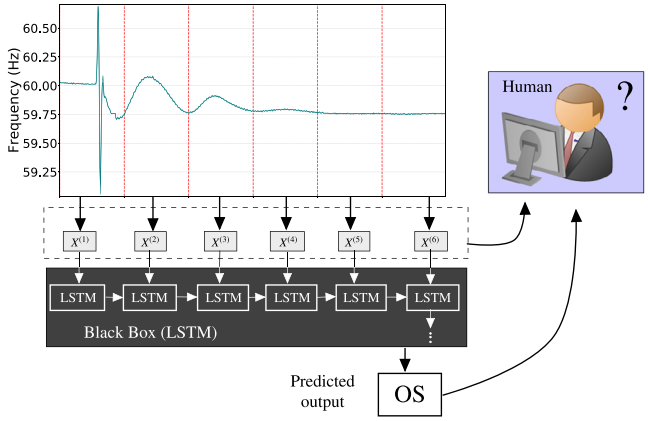


Fig. 1. Event identification using an usual LSTM data-driven method (black-box).

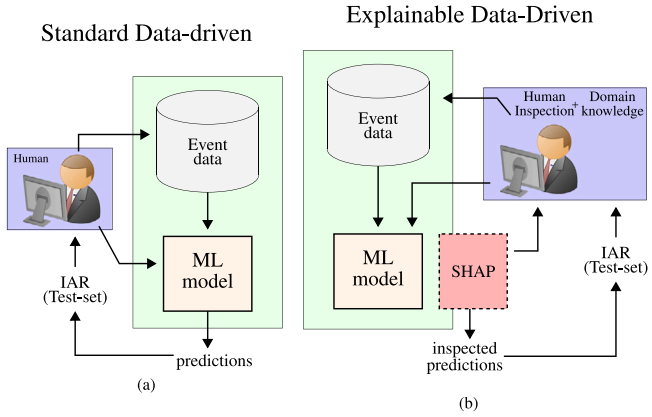


Fig. 2. Our approach using LSTM with SHAP Inspection compared with standard data-driven. (a) Standard Data-driven approach, trained focusing only to obtain the highest IAR of the Test-set. (b) Explainable Data-driven approach (LSTM-SHAP).

2.1. Contributions

In summary, the main contributions of our explainable data-driven approach are as follows: (i) Application of SHAP inspection for real time-series analysis using the LSTM; (ii) Identification and location of the parts of the input time-series with most contribution to the classification of the events; (iii) Understanding of how the DNN classifier is making its decisions in the identification of the events in a EPS critical infrastructure; (iv) A new methodology for training using both the IAR and the interpretability inspection.

3. Data-set description

The data-set is a collection of 168 real events, occurred between June 2010 to July 2015 [25], acquired by the Medfasee Project Low Voltage synchrophasor system (LV-WAMS) which covers all the BIPS [26].

Events data-set

Four types of events are considered: Generation Tripping (GT), Loading Shedding (LS), Oscillation (OS) and Line Tripping (LT). It should be noted, reflecting what happens in practical systems, the majority of relevant events collected are GT and LS resulting in an unbalanced data-set. To overcome this issue, the following data augmentation techniques [27] were applied for LS, OS and LT events: *variable median filter*, which is used to change the size of the spike; *time-shifting*, which is used to change the position of the spike in time; and *injection of noise*.

Table 1
Data-set splitting.

Set	Event type				
	GT	LS	LT	OS	Total
Training	36	13	6	3	58
Test	49	34	18	9	110

Table 2
Examples generated in training-set.

Events type	GT	LS	LT	OS	Total
No of examples	523	522	522	522	2089

All the time-series are frequency records (acquired at 60 samples/second) because the PMUs are connected to the distribution system. A total time window of 10 s (600 samples), consisting of one second of pre-event (system frequency normal behavior) and 9 s of post-event, is considered. In order to train the classifier the data-set was split into a Training-set (34.524%), and a Test-set (65.476%), represented in Table 1, by randomly selecting the events in the database. The resulting training-set obtained by the application of the data augmentation technique is presented in Table 2. It can be observed that the dataset has almost the same number of training examples per event. Regarding the size of training-set, in general, for traditional problems such as image and text classification the data-sets are split in 70%–30% (training–testing), where the training-set is greater than the testing-set. However, time-series problems usually have different train–test splits with almost the same number of samples for training-test such as presented in [28].

The data was normalized according to

$$\boldsymbol{x}_{scaled} = \left(\frac{\boldsymbol{x} - \boldsymbol{x}_{min}}{\boldsymbol{x}_{max} - \boldsymbol{x}_{min}}\right)(\max - \min) + \min \tag{1}$$

where min = −1, max = 1 are the feature ranges for scale. The limits are set to $\boldsymbol{x}_{min}$ = 59 Hz and $\boldsymbol{x}_{max}$ = 61 Hz, which are the boundaries established for this normalization. This normalization rescales the input time-series into the fixed range (−1,1) making the training faster and improving the convergence.

4. LSTM classifier

The main goal of a machine learning classification model is to predict the class label for unlabeled input instances. These features instances can be described by feature values from a feature space. These predictions are realized based on background knowledge and knowledge extracted from a sample of labeled instances (usually in the form of a Training-set) [29]. In this section, we present an overview of the LSTM classifier used in this paper. For the LSTM the input vector is $\boldsymbol{x} \in \mathbb{R}^{600}$, representing a frequency record time-series of 600 samples and the output vector is $\hat{\boldsymbol{y}} \in \mathbb{R}^4$, representing the specified classes (GT, LS, OS and LT). For the given Training-set, described in the precious section, $\mathbb{D} = \{\boldsymbol{x}_i \in \mathbb{R}^{600}, \boldsymbol{y}_i \in \mathbb{R}^4, i = 1, \ldots, N\}$ contains N = 2089 pairs of training data and the corresponding labels. $\boldsymbol{y}_i^k$ is a binary vector where only the kth entry is 1, if $\boldsymbol{x}_i$ belongs to class k. This is known as one-hot encoding. When the input to the LSTM classifier is $\boldsymbol{x}_i$, the output class score for $\boldsymbol{x}_i$ is $\hat{\boldsymbol{y}}_i$.

4.1. LSTM network

The LSTM is based on a gradient-based method proposed by [11]. The LSTM is a upgraded RNN model with a special memory cell. The architecture of the LSTM memory cell is presented in Fig. 3.

The recurrent transition of the LSTM model is given by

$$\begin{pmatrix} \tilde{\boldsymbol{f}}^{(t)} \\ \tilde{\boldsymbol{i}}^{(t)} \\ \tilde{\boldsymbol{o}}^{(t)} \\ \tilde{\boldsymbol{g}}^{(t)} \end{pmatrix} = \boldsymbol{W}_h \boldsymbol{h}^{(t-1)} + \boldsymbol{W}_x \boldsymbol{x}^{(t)} + \boldsymbol{b} \tag{2}$$

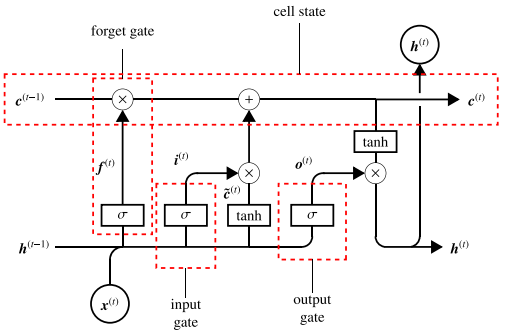


Fig. 3. LSTM memory cell.

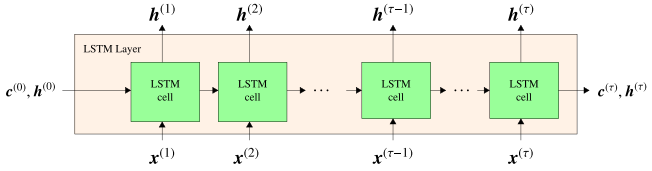


Fig. 4. LSTM layer. $c^{(\tau)}$ and $h^{(\tau)}$ are the final states.

$$c^{(t)} = \sigma(\tilde{f}^{(t)}) * c^{(t-1)} + \sigma(\tilde{i}^{(t)}) * \tanh(\tilde{g}^{(t)}) \tag{3}$$

$$h^{(t)} = \sigma(\tilde{o}^{(t)}) * \tanh(c^{(t)}) \tag{4}$$

where $W_h \in \mathbb{R}^{d_h \times 4d_h}$ denotes the input weight matrix, $W_x \in \mathbb{R}^{d_h \times 4d_h}$ denotes the recurrent weight matrix, $b \in \mathbb{R}^{4d_h}$ denotes the bias matrix and the initial states $h^{(0)} \in \mathbb{R}^{d_h}$; $c^{(0)} \in \mathbb{R}^{d_h}$ are model parameters. The $*$ operator denotes the Hadamard product and d_h is the number of the units.

The time t presented in Fig. 3 and in the set of Eqs. (2)–(4) is know as time-step. Following the definition of the term time-step used in [30] we refer to the term time-step in this work as set of samples subdivided from the time-window defined that establish the order of the time-series. Usually in electrical engineering especially for transient studies the term *time-step* refers to the integration step used in the solution of the differential equations. However, in Deep Learning the term *time-step* is related to the order of a sequence, for example in a NLP problem such as text classification each word of a sentence will be the time-step t used in the LSTM for classify the text.

One of the main differences of this model in compare with RNNs is that the LSTM has an additional memory cell $c^{(t)}$ with nearly linear update which allows the gradient to flow back through time more easily [31]. The LSTM memory cell is regulated by the set of gates $\tilde{i}^{(t)}$, $\tilde{f}^{(t)}$ and $\tilde{o}^{(t)}$. The forget gate $\tilde{f}^{(t)}$ determines the extent to which information is carried over from the previous time-step $t-1$, and the input gate $\tilde{i}^{(t)}$ controls the flow of information from the current input $x^{(t)}$. The output gate $\tilde{o}^{(t)}$ allows the model to read from the cell. This model can be represented by

$$h^{(t)} = LSTM([c^{(t-1)}, h^{(t-1)}], x^{(t)}, \theta) \tag{5}$$

where the $LSTM(\cdot)$ computes Eqs. (2)–(4). At the time-step t, the $LSTM(\cdot)$ uses the previous state of the network $[c^{(t-1)}, h^{(t-1)}]$ and the current time-step t of the sequence $x^{(t)}$ to compute the output $\tilde{o}^{(t)}$ and the updated cell state $c^{(t)}$. The LSTM layer, represented in Fig. 4, illustrates the flow of the information for a time-series $X = \{x^{(1)}, x^{(2)}, \ldots, x^{(\tau)}\}$ with m features (channels) of length τ, where each point $x^{(t)} \in \mathbb{R}^m$. The parameters of the LSTM layer are represented by $\theta = \{W_h, W_x, b\}$.

4.2. Proposed LSTM classifier

In this work, better generalization results have been obtained with the function ReLU (see Eq. (6)) in place of **tanh**$(\cdot)$ as in Eq. (3) and (4). Also, the output layer activation function is **softmax**$(\cdot)$. The LSTM classifier structure is represented in Fig. 5. The penultimate layer is a Fully Connected (FC) layer with the main purpose of reducing the over-fitting of the previous LSTM layers with the use of dropout technique in this layer.

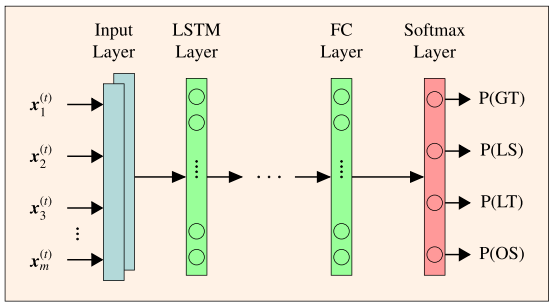


Fig. 5. LSTM classifier.

$$\mathrm{ReLU}(z) = \max(0, z) \tag{6}$$

The input vector $x \in \mathbb{R}^{600}$ is translated (or subdivided) into τ time-steps, as described in Fig. 4. Then, the LSTM receives a time-series in the form $X = \{x^{(1)}, x^{(2)}, \ldots, x^{(\tau)}\} \in \mathbb{R}^{\tau \times m}$, where each point $x^{(t)} \in \mathbb{R}^m$ in the time series is a m-dimensional vector $x^{(t)} = [x_1^{(t)}, x_2^{(t)}, \ldots, x_m^{(t)}]^T$. m is dependent of τ computed as $m = \frac{600}{\tau}$.

In this structure, the LSTM hidden layers, for the given time-step t, are given by

$$h^{(s,t)} = LSTM_{\mathrm{ReLU}}([c^{(s,t-1)}, h^{(s,t-1)}], x^{(t)}, \theta^{(s)}), \quad s = 1 \cdots n_h - 1 \tag{7}$$

where n_h denotes the last hidden layer (number of hidden layers). The information flows to all LSTM layers along all the times-steps $t = 1, \ldots, \tau$, as described in Fig. 4. Then, the output of the last LSTM layer in the last time-step $h^{(n_h-1,\tau)}$ is fed to the FC layer with the ReLU to increase the sparsity. Therefore, the FC output is computed as

$$h_{FC} = \mathrm{ReLU}(W^T h^{(n_h-1,\tau)} + B) \tag{8}$$

where $W \in \mathbb{R}^{d_h^{(n_h-1)} \times d_h^{(n_h)}}$ denotes the weight matrix, $B \in \mathbb{R}^{d_h}$ denotes the bias matrix of the FC layer, $d_h^{(s')}, s' = 0, \ldots, n_h$ is the dimension of the layer s', $d_h^{(0)} = 600$ is the dimension of the input layer and $d_h^{(n_h)}$ is the dimension of the FC layer. The output class scores $\hat{y} \in \mathbb{R}^4$ are computed from

$$\hat{y} = \mathrm{softmax}((W^o)^T h_{FC} + B^o) \tag{9}$$

where $W^o \in \mathbb{R}^{d_h^{(n_h)} \times 4}$, denotes the output weight matrix, $B^o \in \mathbb{R}^4$ denotes an output bias matrix.

To train a suitable parameter set $\Theta = \{\theta^{(s)}, s = 1, \ldots, n_h - 1, W, B, W^o, B^o\}$, we minimize the cross-entropy loss function, and the optimal parameter set Θ can be computed as

$$\Theta = \arg\min_{\Theta} \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{4} y_i^k \log \hat{y}_i^k \tag{10}$$

where $N = 2089$ is the number of training examples, y_i^k corresponds to the kth element of one-hot encoded label of the example x_i, $\hat{y}_i^k$ corresponds to the kth element of $\hat{y}_i$.

The stochastic gradient descent method RMSprop [32] is employed with a decay $\gamma = 1 \times 10^{-6}$ and learning rate $\eta = 0.001$. Also, the Xavier initialization [33] and batch normalization with momentum α are used. In order to improve the generalization capacity, the dropout technique is used. The dropout is applied to LSTM layers and FC. Due to a acknowledged noise problem of applying dropout to the standard LSTM [34], the dropout applied to the LSTM layers is known as

variational dropout that uses the same dropout mask at each time-step, the standard dropout uses different masks at different time steps [35]. Furthermore, the dropout technique is used along with the constraint $\|\boldsymbol{w}\| \leq c$, where $\boldsymbol{w}$ represents the vector of weights incident on any hidden unit and c is a fixed constant. This constraint is imposed during training by projecting $\boldsymbol{w}$ onto the surface of a ball of radius c, whenever $\boldsymbol{w}$ goes out of it [36]. The dropout is applied to the LSTM and FC layers. This combination has been proven to be one of the most efficient ways of avoiding over-fitting [36].

5. Interpreting LSTM predictions using SHAP inspection

To understand how complex models take decisions is a relevant problem for data science applications [24]. Several methods were proposed in literature to cope with this issue, such as LIME [37], DeepLIFT [38], Layer-wise and Relevance Propagation (LRP) [39]. However, these methods still lack of a theoretical background in order to be properly applied in real-world applications. Recently, the authors in [23] proposed the SHAP values method that is embedded with some formal definitions, axioms, and proprieties based on cooperative game theory that help to fill this gap. First, the authors state that any explanation of a prediction model must be a model itself. Thus, they introduce the term *explanation model* g which is the best interpretable approximation of the original prediction model f [23]. Second, based on game theory concepts known as Shapley values, they show theoretical results that guarantee a measure of feature importance that other methods only approximate.

5.1. Game theory: Credit allocation

The area of game theory is generally divided into two branches called non-cooperative game theory and cooperative game theory. In the non-cooperative game, the actors in the game are individuals players, i.e., they do not cooperate with each other in any way [40]. In the cooperative game, the players in the game are coalitions (group of players) where they cooperate with other forming sets with no order or hierarchy. Thus, given the coalitions and their sets of feasible payoffs as primitives, the question tackled is the identification of final payoffs awarded to each player [40]. The problem of credit allocation is direct related to cooperative game theory working with coalitions of players instead of individuals players [41]. The process of credit allocation is based on subdivide the credit of an activity between the players according to some responsibilities and benefits [41]. This credit allocation must encourage the cooperation between the players to induce the efficient use of the resources. More specifically, the Shapley value ($\phi_j(f,x)$) is one way to distribute the total gains to the players, assuming that they all collaborate. In the event identification problem, a coalition is defined by a set of interpretable input feature values (or players) and the output of the coalition is the value of the prediction made by the model. So, there is a set Z (of M players or features) and a conditional expectation function $f_x(S) = E[f(x)|S]$ that maps subsets S of players to the real numbers: $f_x(S) : 2^S \rightarrow \mathbb{R}$, with $f_x(S)(\emptyset) = 0$, where $\emptyset$ denotes the empty set [24].

5.2. Shapley values

The Shapley values $\phi_j(f,x)$ are used to explain a prediction $f(x)$ by a set of single numerical values representing the impact of each feature x_j on the prediction model $f(x)$ given by the single input x. The authors in [42] states that Shapley values $\phi_j(f,x)$ are the only method of allocation that obeys a set of desirable properties known as Shapley properties 5.2.1. These properties are important to guarantee that the resulting explanation model g is able to properly interpret the original function (f).

5.2.1. Shapley values properties

Local accuracy. The local accuracy property is given by the following equality

$$f(x) = \phi_0(f,x) + \sum_{j=1}^{M} \phi_j(f,x) \tag{11}$$

where $\phi_0(f,x) = E[f(x)]$. The local accuracy property forces the attribution values to correctly capture the difference between the expected model output $E[f(x)]$ and the output for the current prediction $f(x)$ [42].

Consistency. For any two models f and f', if

$$f'_x(S \cup \{j\}) - f'_x(S) \geq f_x(S \cup \{j\}) - f_x(S) \tag{12}$$

For all $S \in Z \setminus \{j\}$, then $\phi_j(f',x) \geq \phi_j(f,x)$. This property guarantee that if a feature j is more important (greater) in one model f' than another f, then the importance attributed ϕ_j to that feature j should also be higher. The feature value contribution $\phi_j(f,x)$ represents the contribution to the payout, weighted and summed over all possible feature value combinations (coalitions). Following game theory considerations [23], it can be proven that only one solution of credit allocation satisfies the Shapley values properties 5.2.1 and that solution $\phi_j(f,x)$ is given by

$$\phi_j(f,x) = \sum_{S \subseteq Z \setminus \{j\}} \frac{|S|!(M-|S|-1)!}{M!} [\underbrace{f_x(S \cup \{j\})}_{\substack{\text{player } j \\ \text{presented}}} - \underbrace{f_x(S)}_{\substack{\text{player } j \\ \text{not presented}}}] \tag{13}$$

where S is a subset of the features used in the model, $f_x(S) = E[f(x)|S]$ is the expected value of the model over the training subset S, M is the number of input features, Z is the set of all M input features and $\phi_j \in \mathbb{R}$. These feature contributions $\phi_j(f,x)$ can be negative or positive depending on how they contribute to the prediction. The major drawback to compute the traditional Shapley values is the high computational burden because the sum in Eq. (13) has 2^M coalitions, i.e., it is necessary to estimate (retrain) 2^M models to compute the Shapley values. In order to tackle this issue, the authors in [23] proposed a sampling processes that approximates the terms of Eq. (13).

5.3. SHapley Additive exPlanation (SHAP) framework

The main innovation from SHAP is the use of the linear model to represent the classic Shapley values matching the Shapley properties. The additive feature attribution methods are the basis on that the SHAP framework compute its values. This simplified linear model is given by [23]:

$$g(z') = \phi_0 + \sum_{j=1}^{M} \phi_j z'_j \tag{14}$$

where $z' \in \{0,1\}^M$ is the simplified inputs, M is the number of simplified inputs features, and $\phi_j \in \mathbb{R}$. Explanation models often use simplified inputs z', also known as the coalition vector, that map to the original inputs through a mapping function $z = h_x(z')$. In the coalition vector, an entry of 1 means that the corresponding feature value is present and 0 that it is absent, this is very similar to Shapley values where we need to simulate that only some features values are playing ("present") and some are not ("absent") [24].

5.3.1. SHAP values

Mathematically, the SHAP values can estimated by the conditional expectation function $f_x(z')$ of the original function f defined as

$$f_x(z') = f(h_x(z')) = E[f(z)|z_S], \tag{15}$$

where S is the set of non-zero indexes in z', to define simplified inputs [23]. To compute the SHAP values we must combine the conditional expectations $f_x(z')$ with the classic coalition game theory from

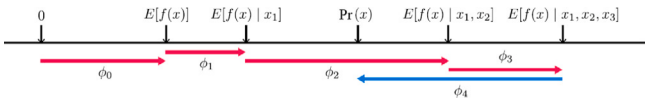


Fig. 6. SHAP values explain how to get from the base value $E[f(x)]$ that would be predicted if we did not know any features to the current probability output $\Pr(x)$. This diagram shows a single ordering $[x_1, x_2, x_3, x_4]$ of the 4!. For non-linear functions the order in which features are introduced matters. SHAP values $\phi_j(f, x)$ arise from averaging the ϕ_j values across all possible orderings that follow the Shapley properties. *Source:* Adapted from [23].

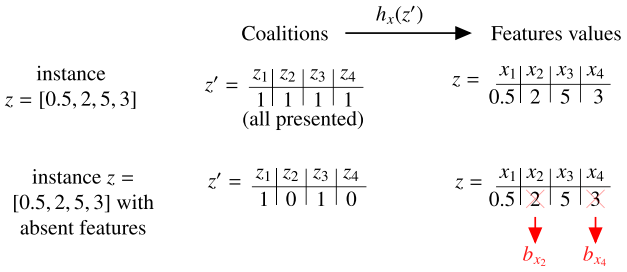


Fig. 7. Function h_x maps the coalition z' to a valid instance z. For present features 1, maps to the feature values of z. For absent features 0, maps to the values of a randomly sampled background data instance b_{x_j}. So, the $E[f(z)|z_S] = f(h_x(z')) = f(z)$. *Source:* Adapted from [24].

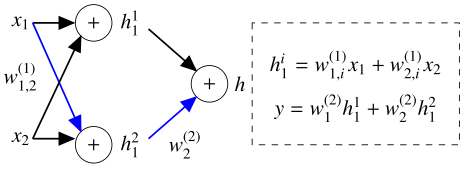


Fig. 8. Fully linear model. *Source:* Adapted from [43].

Shapley value (Eq. (13)). SHAP values attribute to each feature the change in the expected model prediction when conditioning on that feature x_j.

For example, consider a simple binary classification problem represented by the logistic regression model f, with output $y = f(x) \in \{0, 1\}$. This model, used for probability estimation, is composed by a linear regression g and a sigmoid function:

$$\Pr(x) = \frac{1}{1 + \exp(-g(x))} \tag{16}$$

where $\Pr(x) = \Pr(y = 1|x)$, which ranges from 0 to 1, is the output probability of class 1, $g(x) = w_0 + \sum_{j=1}^{4} w_j x_j$ is a linear regression with bias w_0, weights $w_j, j = 1, \ldots, 4$ and input features $\boldsymbol{x} = [x_1, x_2, x_3, x_4]$.

The main purpose of SHAP is to provide a methodology to compute the individual contribution ϕ_j of a particular input feature x_j to the overall model estimative. The procedure is illustrated in Fig. 6 for the case of $n = 4$, representing one single ordering. This is realized computing the contribution ϕ_j using the conditional expectations $E[f(z)|z_S]$ (Eq. (15)) starting with a null entry (no feature) that results in base value ($\phi_0 = E[f(x)]$). Afterwords, the other features are added one by one and their respective contributions are estimated by the $E[f(z)|z_S]$ using the base value as a reference. It should be noted that the output of the contribution $\phi_j(f, x)$ is a real number that can be positive $\phi_j(f, x) > 0$ (in red) or negative $\phi_j(f, x) < 0$ (in blue). In this particular case, like the classification problem, the positive and negative values increase or decrease the value of the probability estimated, respectively. There are $n = 4!$ possible orderings (or permutations) that can have different contributions. However, only the ones that follow the Shapley properties are evaluated and their average is used as final result. Finally, the sum of contributions ($\sum_j^4 \phi_j(f, x)$) must satisfy the local accuracy property (Eq. (11)).

In Fig. 6, the features $\phi_j, j = 1, \ldots, 3$ are positive because they push the $E[f(z)|z_S]$ higher (increasing the probability output $\Pr(x)$) and ϕ_4 is negative because it pushes the $E[f(z)|z_S]$ lower. These values explain the output probability $\Pr(x)$ as sum of the effects ϕ_j of each feature being introduced into the conditional expectation [23]. In practice, the base value $E[f(x)]$ are computed by taking the average of $\Pr(x)$ over a background data-set, a sub-set of the training set. Also, for ANN is very difficult to emulate missing features and compute the $E[f(z)|z_S]$. So, the mapping function h_x must emulate the missing features from features samples of the background data b_{x_j}, as represented in Fig. 7.

The exact computation of SHAP values are still hard, but one advantage is that is not necessary to retrain 2^M models as in the classic Shapley values estimation. So, in order to speed up the computation of SHAP values we apply the Deep SHAP method that is suitable for deep neural networks.

5.3.2. Deep SHAP

The Deep SHAP is a combination between DeepLIFT and Shapley values [23] taking advantage of compositional nature of ANN (net of neurons). The DeepLIFT is a recursive prediction explanation method for deep networks that attributes to each input x_j a value $C_{\Delta x_j \Delta y}$ that represents the effect of that input being set to a reference value as opposed to its original value [38].

The basic idea for computing SHAP values using DeepSHAP can be understood considering a fully linear model, as presented in Fig. 8. In this simple model the input feature x_j can follow two possible paths. The exact SHAP value for this input is obtained by summing the attributions along all possible paths between that input x_j the model's output y [43].

Focusing on the blue path highlighted, the path's contribution to $\phi(x_1)$ is the product of the weights along the path with the difference among x_1 and its base value $E[x_1]$ given by

$$\phi(x_1) = w_2^{(2)} w_{1,2}^{(1)} (x_1 - E[x_1]). \tag{17}$$

The contribution of the blue path to $\phi(h_1^2)$ is

$$\phi(h_1^2) = w_2^2 (h_1^2 - E[h_1^2]) \quad \Rightarrow \quad w_2^2 = \frac{\phi(h_1^2)}{h_1^2 - E[h_1^2]} \tag{18}$$

Substituting Eq. (18) in Eq. (17) results in the contribution to $\phi(x_1)$ in terms of $\phi(h_1^2)$

$$\phi(x_1) = \frac{\phi(h_1^2)}{h_1^2 - E[h_1^2]} w_{1,2}^{(1)} (x_1 - E[x_1]) \tag{19}$$

For Deep models, with hidden activation's functions (non-linearities) such as ReLU, sigmoid, and tanh we cannot simply backward the products weights along the paths. So, DeepSHAP uses some of DeepLIFT properties known as rules that simplify the ANN, linearizing the non-linear components of the ANN. DeepLIFT established the following properties [38]: Chain, Rescale, and RevealCancel rules. The chain rule is important to backpropagating the multipliers dy/dx across all neurons. Rescale rule and RevealCancel can be used to simplify the non-linearities and propagate the attributions to get ϕ_j.

In Fig. 9(a), a non-linear neuron model g is presented as example to show how SHAP values are approximated using the Rescale rule. Under this rule SHAP values are computed for h as $\phi(h) = g(h) - g(E[h])$, given the local accuracy property and because the g node has only one input, so

$$\frac{dy}{dh} = \frac{\phi(h)}{h - E[h]} \tag{20}$$

Then, dy/dh is propagated it back linearly using the chain rule, obtaining

$$\phi(x_i) = \frac{\phi(h)}{h - E[h]} w_i (x_i - E[x_i]),$$

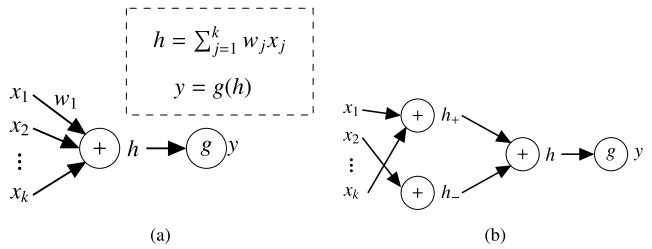


Fig. 9. Neuron model where g is a non-linear function and $h = \sum_j^k w_j x_j$. (a) Rescale rule (b) RevealCancel rule.
Source: Adapted from [43].

approximating the non-linear attributions [43].

The RevealCancel rule, presented in Fig. 9(b), partitions x_j into positive and negative components (intermediate nodes h_+ and h_-) based on the condition $w_j(x_j - E[x_j]) < t$ (where t = mean value of $w_j(x_j - E[x_j])$ across j) forming the nodes

$$h_+ = \sum_j 1\left\{w_j(x_j - E[x_j]) > t\right\} w_j x_j, \text{ and} \tag{21}$$

$$h_- = \sum_j 1\{w_j(x_j - E[x_j]) < t\} w_j x_j. \tag{22}$$

This rule computes the exact SHAP attributions for h_+ and h_-

$$\phi_{h_+} = \frac{1}{2}[g(h_+ + h_-) - g(E[h_+] + h_-) + g(h_+ + E[h_-]) - g(E[h_+] + E[h_-])] \tag{23}$$

$$\phi_{h_-} = \frac{1}{2}[g(h_+ + h_-) - g(E[h_+] + h_-) + g(h_+ + E[h_-]) - g(E[h_-] + E[h_-])] \tag{24}$$

then propagates the resultant SHAP values back linearly.

$$\phi_j = \phi(x_j) = \begin{cases} \frac{\phi_{h_+}}{h_+ - E[h_+]} w_j(x_j - E[x_j]), & \text{if } w_j(x_j - E[x_j]) > t. \\ \frac{\phi_{h_-}}{h_- - E[h_-]} w_j(x_j - E[x_j]), & \text{otherwise.} \end{cases} \tag{25}$$

Note that the contribution ϕ_j is derived in term of the intermediary nodes using the chain rule [43]. The RevealCancel rule exactly explains the non-linearity and a partition of the inputs to the linearity as a single function prior to backpropagating, thus improving the rescale rule as demonstrated in [43].

6. Methodology

The proposed methodology is detailed as follows putting all the concepts together showing how to apply the SHAP inspection on the LSTM for the event identification problem.

Step 1: Train the LSTM classifier. The LSTM classifier is trained the on the Training-set, minimizing the cross-entropy loss according to Eq. (10). The performance of identification is given by IAR (Identification Accuracy Rate) and Balanced Accuracy (BA). The IAR or Classification Accuracy is a simple and very used metric to evaluated the performance of the classifier. By definition, IAR is the ratio of number of correct predictions to the total number of input events:

$$\text{IAR}\% = \frac{\text{N}_{\text{correct}}}{\text{N}_{\text{total}}} \times 100\% \tag{26}$$

where $\text{N}_{\text{correct}}$ is the number of correctly classified events and N_{total} is the total number of the events. It should be noted that IAR is an index of overall performance and it is not useful to evaluate the performance of a specific class. For example, the overall performance of the classifier may be satisfactory however it may be failing to classify one specific type of event. Thus, in order to verify the performance of the classifier for each class the precision per class is computed. Thus, the precision's GT%, LS%, LT% and OS% are as follows:

$$\begin{aligned} \text{GT}\% &= \frac{\text{N}^{\text{GT}}_{\text{correct}}}{\text{N}_{\text{GT}}} \times 100\% \\ \text{LS}\% &= \frac{\text{N}^{\text{LS}}_{\text{correct}}}{\text{N}_{\text{LS}}} \times 100\% \\ \text{LT}\% &= \frac{\text{N}^{\text{LT}}_{\text{correct}}}{\text{N}_{\text{LT}}} \times 100\% \\ \text{OS}\% &= \frac{\text{N}^{\text{OS}}_{\text{correct}}}{\text{N}_{\text{OS}}} \times 100\% \end{aligned} \tag{27}$$

where $\text{N}^{\text{GT}}_{\text{correct}}$, $\text{N}^{\text{LS}}_{\text{correct}}$, $\text{N}^{\text{LT}}_{\text{correct}}$, $\text{N}^{\text{OS}}_{\text{correct}}$ denote, respectively, the number of correctly classified events for GT, LS, LT, and OS. The number of events for GT, LS, LT, and OS are denoted, respectively, by N_{GT}, N_{LS}, N_{LT}, N_{OS}. The BA is calculated as the average of the proportion of each class individually. This is necessary due to the imbalance of the number of classes in the Test-set.

Step 2: Sample the background set for DeepSHAP. The background data-set are sampled from the Training-set to compute the base values $E[f(x)]_k, k = 1, \ldots, 4$ (one base value for each event type GT, LS, LT and OS), $E[x_j], j = 1, \ldots, 600$ and conditional expectations $E[f(z)|z_S]$.

Step 3: Select events from test-set for inspection. Some events of the Test-set are selected for inspection. We choose to select one correctly classified event of each type, to give insights about what is being learned to correctly classify the events, and the misclassified events, to give insights about why the classifier made the mistake, using the base value $E[f(x)]$ of the correct label to compute the SHAP values.

Step 4: Apply DeepSHAP for the LSTM to compute the SHAP values. The DeepSHAP method is applied to the LSTM classifier to compute the SHAP values in order to provide explanations about the selected events. The most important parts are highlighted in the time-series input. The direct result of the application of the Deep SHAP method are the sample value contributions $\bar{\phi}_j^{(t)}$ for each sample j in the time-step t, representing the average contribution over the PMUs signals of the event. These results allow the user to see what samples are most significant in all time series to identify a specific event. Other way to extract information from SHAP values is to verify which time-steps t (set of samples) most contributed to the event identification. This is realized by calculating the sum value of the sample contribution $\bar{\phi}_j^{(t)}$ for each time-step t, combining set of samples of the same time-step t, using

$$\boldsymbol{\Phi}^{(t)} = \sum_{j=1}^{m} \bar{\phi}_j^{(t)} \tag{28}$$

where m is the dimension of $\boldsymbol{x}^{(t)}$, and $t = 1, \ldots, \tau$. One interesting point of using the time-steps contributions $\boldsymbol{\Phi}^{(t)}$ instead of using the sample contribution $\bar{\phi}_j^{(t)}$ is that we still maintain the local accuracy propriety (Eq. (11)), so

$$\sum_{t=1}^{\tau} \boldsymbol{\Phi}^{(t)} = \sum_{t=1}^{\tau} \sum_{j=1}^{m} \bar{\phi}_j^{(t)} = f(x) - E[f(x)] \tag{29}$$

Step 5: Inspect the predictions of the LSTM. For each event type (GT, LS, LT and OS) the SHAP inspection is applied identifying the main contributions (SHAP values) involved in the classification of these events, both local and global explanations. The local explanation stands for explaining single predictions, and the global explanation deals with understanding the predictions globally in general by combining multiples local explanations providing summaries of the classifier and time-steps. As presented in [44], by combining many local explanations is an effective way of obtain global explanations about the classifier. The global explanation are presented using some visualizations tools of SHAP values: the bar chart of the SHAP average magnitude, and

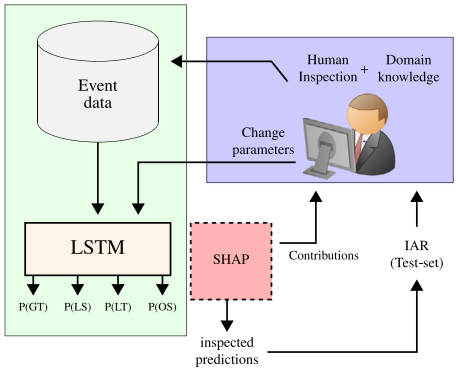


Fig. 10. LSTM-SHAP classifier training process.

beeswarm plot (or dot plot). These contributions are computed using the DeepSHAP with the RevealCancel rule. The predictions of the LSTM are inspected observing the coherence of the classifier and detecting possible bias in the LSTM predictions based on the knowledge domain of the events.

Step 6: Improvements of LSTM based on the SHAP inspection. The use of the knowledge obtained through the SHAP inspection is used to improve the LSTM creating a new model LSTM-SHAP, correcting the bias and improving the coherence of the classifier predictions.

The obtained knowledge through the SHAP inspection is used to improve the LSTM by creating a new classifier LSTM-SHAP, correcting the possible bias and inconsistencies, and improving the coherence of the classifier predictions. Therefore, after applying modifications to data and/or model, for every trained LSTM model with different initials conditions and hyper-parameters, we observe not only the IAR and BA of the Test-set, but also observe the SHAP values of the selected events, both locally and globally. This process is represented in Fig. 10.

The main purpose of SHAP inspection is to obtain an LSTM that have the main contributions $\Phi^{(t)}$ according to domain knowledge of the events. For example, the LSTM should be able:

1. To identify the downfall in frequency for GT events;
2. To identify the rise in frequency for LS events;
3. To identify the spikes in the beginning of the time-series for LT events;
4. To identify the oscillation peaks (maximum and minimum) soon after the spike for OS events.

Thus, implicitly we introduce this domain knowledge into the LSTM training process.

7. Performance evaluation in BIPS events

The evaluation of the LSTM classifier are performed using both the identification rate and the interpretability inspection using SHAP values.

7.1. Performance of classifying the events (IAR and BA)

The LSTM classifier is trained offline using the Training-set. We chose different initialization values of LSTM parameters and obtain multiple sets of parameters. The trial and error method has used for evaluation of LSTMs results on the Test-set, and the best one evaluated in the Test-set is selected. According to Table 3, the LSTM classifier has three hidden layers with 60, 40 and 30 neurons, respectively. The FC layer has 30 neurons. The dropout rate applied to each layer is 0.3, 0.4, and 0.5, respectively.

The results presented in Table 4 displays that the LSTM classifier shows a high IAR for GT, LS, LT and OS events. The classifier can identify the events with an overall IAR equal to 98.182%. Also, the BA% achieved for the LSTM classifier is presented.

Table 3
Best parameters of LSTM classifier.

n_h	Neurons	Dropout rate	α
3	60|40|30	[0.3, 0.4, 0.5]	0.55

Table 4
Performance of LSTM classifier for identifying the events.

GT%	LS%	LT%	OS%	IAR %	BA%
97.959	100.0	94.444	100.0	98.182	98.101

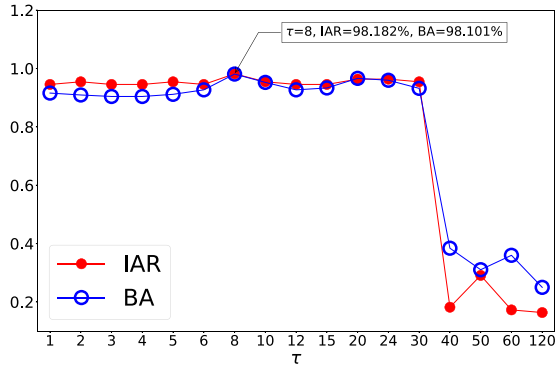


Fig. 11. Performances of LSTM classifier in relation to the number of times-steps τ. Best performance at $\tau = 8$.

Table 5
Performances of MSVM, MLP, and LSTM.

Classifier	GT%	LS%	LT%	OS%	IAR %	BA%
SVM	85.714	94.118	66.667	88.889	87.273	83.847
MLP	**100.0**	91.176	66.667	88.889	92.727	86.683
LSTM	97.959	**100.0**	**94.444**	**100.0**	**98.182**	**98.101**

Table 6
Misclassifications of MSVM, MLP, and LSTM.

Classifier	GT	LS	LT	OS	Total
MSVM	7	2	2	3	14
MLP	0	3	2	3	8
LSTM	**1**	**0**	**0**	**1**	**2**

Performance relation to time-steps τ. The number of time-steps τ, described in Fig. 4, is important in the performance of LSTM. The results in Fig. 11 indicate the IAR and BA in the test-set of the LSTM classifier as a function of τ. Usually, the LSTM has an optimal τ, in most case the LSTM has worse performance when τ is higher than this optimal value. The LSTM classifier has the best performance with $\tau = 8$. An interesting point is the terrible performance of LSTM with higher time-steps values $\tau > 40$. This is due to the saturation in the extraction of LSTM in learning the dynamic features.

7.1.1. Comparison of different classifiers

In order to show the advantage of our classifier in compare with other known methods from literature, we compared the LSTM with MLP and Multi-Class SVM (MSVM). The IARs of the MSVM, MLP and LSTM are compared in Table 5.

The misclassifications presented in Table 6 show that the LSTM classifier made only one error for GT and OS events. Therefore, the LSTM incorrectly classifies only 2 events in the total of the Test-set. The misclassifications are labeled as event 9 and event 98, representing the position in the Test-set. Event 9 is a GT event that was classified as LT, and event 98 is a LT event that was classified as GT.

The MSVM is the extension of binary-class SVM with one-vs-rest scheme using the Radial Basis Function (RBF) as kernel and regularization parameter $C = 6.722$, obtained using a 10-fold Cross-Validation

Table 7
Base values $E[f(x)]$.

$E[f(x)]_{GT}$	$E[f(x)]_{LS}$	$E[f(x)]_{LT}$	$E[f(x)]_{OS}$
0.7725	0.1958	0.0247	0.0071

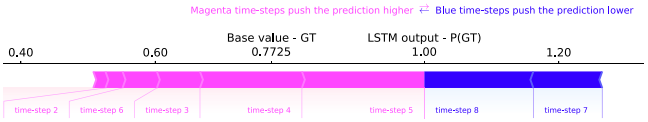


Fig. 12. Force plot of LSTM for Event 3.

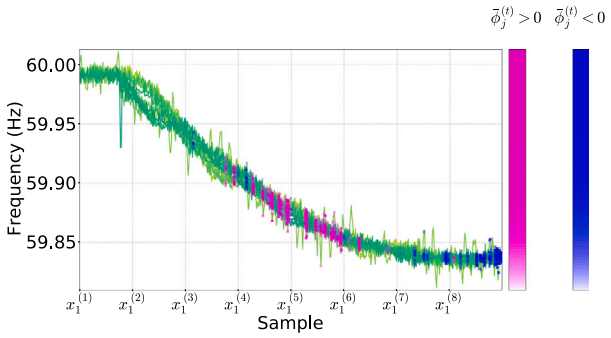


Fig. 13. Event 3. Frequency PMUs signals with the most relevant contributions highlighted. Base value $E[f(x)]_{GT} = 0.7725$.

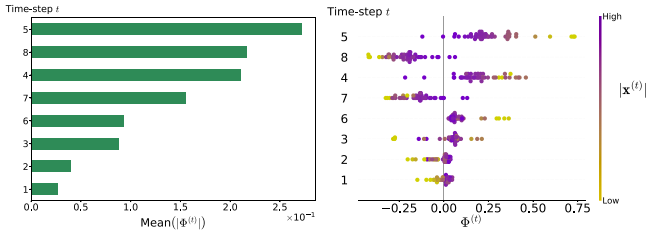


Fig. 14. Global Explanation GT events. (a) Bar chart of average $\Phi^{(t)}$ magnitude. (b) Beeswarm plot of $\Phi^{(t)}$.

and BA% as score function. The MLP classifier also have three layers, similar to LSTM, and the main difference is the replacement of the LSTM layers with the FC layers. Also, the training parameters are equal to the LSTM. The results in Tables 5 and 6 indicate that the LSTM has achieved the highest overall IARs among these three classifiers.

7.2. Interpretability inspection of LSTM classifier

In our inspections, we analyzed the correctly classified events, to understand how the classifier takes its decisions, and incorrectly classifies events, and also we tried to understand the limitations of the classifier. The base values (one from each event type) $E[f(x)]$ displayed in Table 7 are computed from averaging the predictions of the LSTM classifier over the non-augmented training-set, known as background data-set.

7.2.1. Inspection of the events

In this section, we will focus on the inspection of GT and LT events because LS is basically a dual of the GT events and the OS will analyzed after the improvements in the training procedure.

GT events. The main characteristic of the GT events are the downfall in the frequency due to the loss of generation in the system. As suggested in [44] the inspection is carried out by using local and global explanations.

(1) Local Explanation: The focus of the local explanation is to interpret each event separately. Looking at a single event we can numerically estimate the positive and negative contributions given the base value $E[f(x)]_{GT}$. In this case, the Event 3, correctly classified as GT with a probability of P(GT) = 100%, was selected and its force plot is presented in Fig. 12. The magenta time-steps t are the ones that contribute to push the probability P(GT) up and the blue time-steps are the ones that contribute to push the P(GT) down. These $\Phi^{(t)}$ values were computed using Eq. (28) with $\tau = 8$ and $m = \frac{600}{\tau} = \frac{600}{8} = 75$. The temporal evolution of this event, highlighting the positive/negative sample contributions, is presented in Fig. 13. According to this graph it is clear to identify parts of the time-series which are more relevant (positive/negative) to the classifier decision taking. As presented, the time-steps $t = 4, 5$ have the greater contributions. These time-steps represent the downfall in the middle of the time-series. The negative time-steps $t = 7, 8$ did not have a relevant impact on P(GT), according to its higher value.

It can be observed from Fig. 13 that the stream of PMUs signals representing the GT event. The LSTM classifies each PMU signal and a soft-majority vote (mean value of output probabilities) is token to classify the whole event. Also, the sample contributions $\bar{\phi}_j^{(t)}$ are computed by taking the average of contributions over the PMUs signals of this event.

(2) Global Explanation: The main goal of the global explanation is to verify which time-steps t have more influence in the classifier decision for a set of events (in this case GT events). This is obtained by combining multiples local explanations, expressing by the average absolute magnitude of the contributions ($|\Phi^{(t)}|$) for each time-step t. The beeswarm plot is useful to evaluate the relation between the local contributions ($\Phi^{(t)}$) for each time-step t and the input values $\mathbf{x}^{(t)}$.

The global explanation for GT events is presented in Fig. 14. Fig. 14(a) exhibits the mean absolute value of the SHAP values for each time-step t as a standard bar chart, and Fig. 14(b) presents the beeswarm plot. In this plot, each dot corresponds to an individual GT event. The dot's position on the x-axis shows the impact that time-step t has on the LSTM prediction P(GT), i.e., the $\Phi^{(t)}$ value. When multiple dots land at the same x position as in time-step $t = 1$, where the dots are near the vertical line, they pile up to show density. The colors represent the value of the input $|\mathbf{x}^{(t)}|$ from low to high (purple to yellow). This plot is applicable for visualizing the relationship between the input ($|\mathbf{x}^{(t)}|$) with the contributions $\Phi^{(t)}$.

As presented, the time-steps $t = 1, 2$ (beginning of the time-series) have a small contribution to the prediction P(GT), and these contributions get higher negative values with lower values of $|\mathbf{x}^{(t)}|, t = 1, 2$. Moreover, the time-steps $t = 5, 8$, and $t = 4$ have the greater contributions, with positive values for $t = 5, 4$ and negative values for $t = 8$ (Fig. 14(b)). As presented in Fig. 14(b), focusing on the main time-steps $t = 5, 4$, we can observe that the events with lower values of $|\mathbf{x}^{(5)}|$ and $|\mathbf{x}^{(4)}|$ (in blue), have greater values of $\Phi^{(5)}$ and $\Phi^{(4)}$ (x-axis), respectively. This relationship is in agreement with the knowledge domain of GT events that greater deviations of Δf indicate greater loss of generation. However, GT events with no great deviation could have problems in being classified as GT based on this relation, as will be observed for the misclassification (Event 9).

The time-step $t = 8$ also presents an interesting relation with $\mathbf{x}^{(8)}$. The $\Phi^{(8)}$ gets higher values in magnitude when the input $|\mathbf{x}^{(8)}|$ is lower (see Fig. 13). This represents that P(GT) is pushed down by time-step $t = 8$ with higher deviations Δf. As observed, these inconsistencies are common in the trained LSTM models, representing deceptive temporal patterns learned by the LSTM. These inconsistencies are important to understand the reasons for the misclassifications that are usually related to these deceptive patterns.

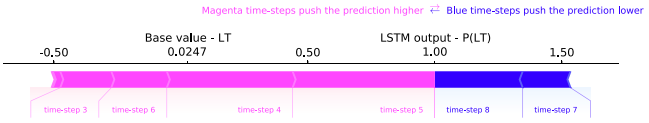


Fig. 15. Force plot of LSTM for Event 39. Base value $E[f(x)]_{\mathrm{LT}}$.

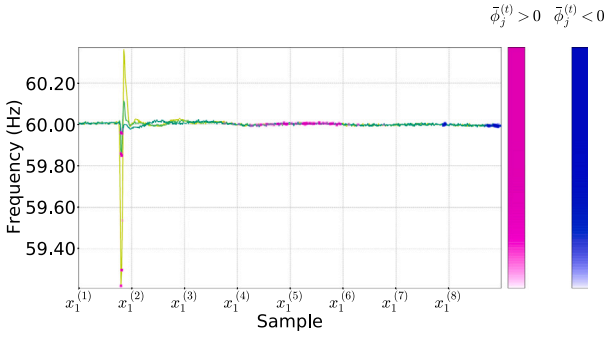


Fig. 16. Event 39. Frequency PMUs signals with the most relevant contributions highlighted. Base value $E[f(x)]_{\mathrm{LT}} = 0.0247$ for both events.

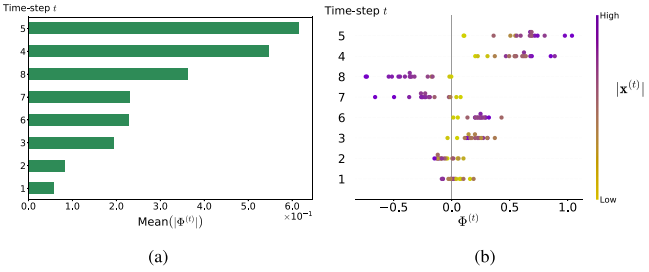


Fig. 17. Global explanation LT events. (a) Bar chart of average $\boldsymbol{\Phi}^{(t)}$ magnitude. (b) Beeswarm plot of $\boldsymbol{\Phi}^{(t)}$.

LT events. The LT events have the main characteristics of the spike frequency in the beginning of the disturbance, after the spike the frequency can rise up, go down or stay in steady state, depending on the post-fault action.

(1) Local Explanation: The LT event selected (Event 39) is shown as follows. The force plot of this event is presented in Fig. 15, given the base value of $E[f(x)]_{\mathrm{LT}}$. The event was classified with a probability of P(LT) = 100%. The time-series of the event are presented in Fig. 16, highlighting the main contributions. It can be notice from Fig. 16 that the spikes and the nominal frequency samples having the most contribution to the identification. However, the higher contributions are from the steady state part (time-steps $t = 5, 4$) after the disturbance. This is a problematic issue because according to the knowledge domain of this event the most important contributions should be from the spike's frequency.

(2) Global Explanation: The global explanation of LSTM for the LT events is presented in Fig. 17. According to Fig. 17(b) the time-steps $t =$ 5, and 4 are the ones with greater contributions, and both of these time-steps are positive. The spike, represented by the time-step $t = 1$, has the lowest contribution and does not have any impact on identification of this type of event. Also, for the inspected LT events, the contributions $\boldsymbol{\Phi}^{(t)}, t = 5, 4$ are higher with greater values of $\left|\boldsymbol{x}^{(t)}\right|, t = 5, 4$, respectively, in this case representing values close to the nominal frequency.

This means that the classifier is learning to identify LT events by examining the steady state part after the disturbance. Therefore, this biased behavior could cause problems in other type of LT events that do not reach the steady state frequency operation after the disturbance. So, even though the LSTM classifier presents a high IAR and BA their performance behavior is biased and this can only be identified using the interpretability inspection.

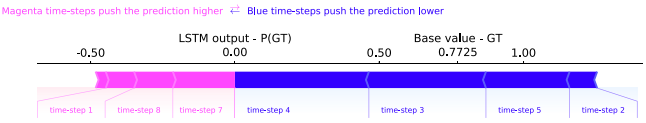


Fig. 18. Event 9. Force plot of the misclassified GT event.

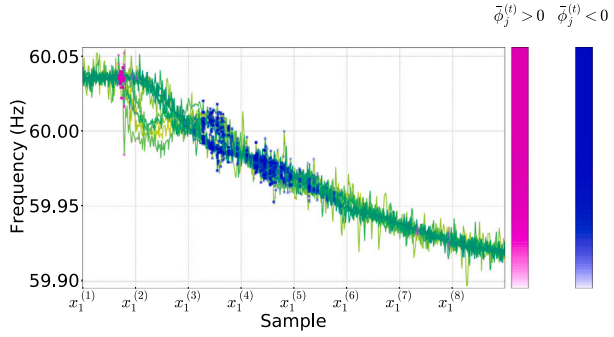


Fig. 19. Event 9. Misclassified GT event with the most relevant contributions highlighted. Base value of GT event $E[f(x)]_{\mathrm{GT}} = 0.7725$.

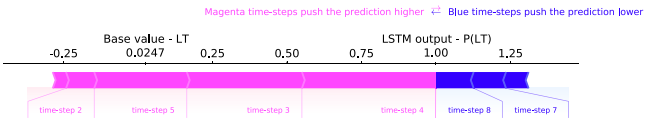


Fig. 20. Event 9. Force plot of the misclassified GT event.

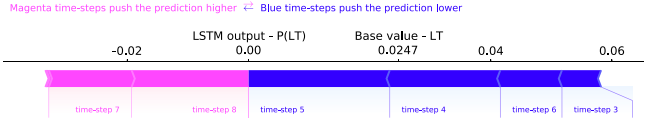


Fig. 21. Event 98. Force plot of the misclassified LT event.

7.2.2. *Inspection of misclassified events*

In this part, two misclassified events (Events 9 and 98) by the LSTM, presented in Table 6, will be discussed.

Event 9. The Event 9 was classified as LT with a probability of P(LT) = 100%, however this is an GT event. The SHAP values are computed using Eq. (13) using the base value of $E[f(x)]_{\mathrm{GT}}$. These contributions are related to the correct GT event affecting P(GT). The output probability is P(GT) = 0%. The force plot of this GT event is presented in Fig. 18. Also, the contributions are displayed in Fig. 19.

It should be noted that the most relevant contributions $\bar{\phi}_j^{(t)}$ are negative. The negative contributions are the ones decreasing the probability P(GT). We also identified that these negative contributions are the positive contributions of the predicted class LT 20. Thus, we attribute this misclassification due to small deviation $\Delta f = 0.05$ Hz. It occurs in the time-steps $t = 4, 3$ and 5 close to a steady state behavior, leading to incorrectly classify this event as LT.

Event 98. The Event 98 was classified as GT with a probability of P(GT) = 100%, however the correct class is LT. The SHAP values are computed with Eq. (13) using the base value of $E[f(x)]_{\mathrm{LT}}$. These contributions are related to identification of the correct event of LT affecting P(LT). The output probability of LT is P(LT) = 0%. The force plot of this LT event is presented in Fig. 21.

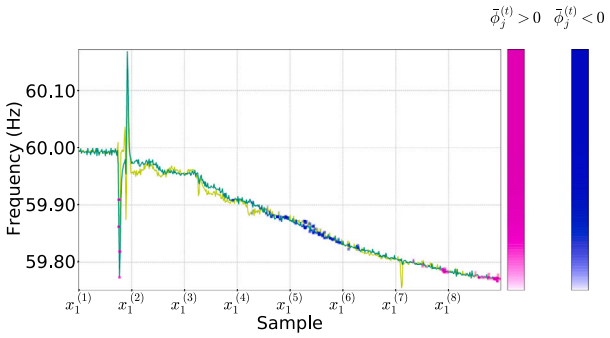


Fig. 22. Event 98. Misclassified LT event with the most relevant contributions highlighted. Base value of LT event $E[f(x)]_{\text{LT}} = 0.0247$.

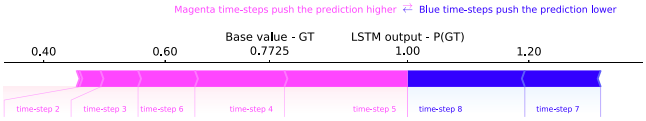


Fig. 23. Force plot of the Event 98. Presenting P(GT) and its base value $E[f(x)]_{\text{GT}} = 0.7725$.

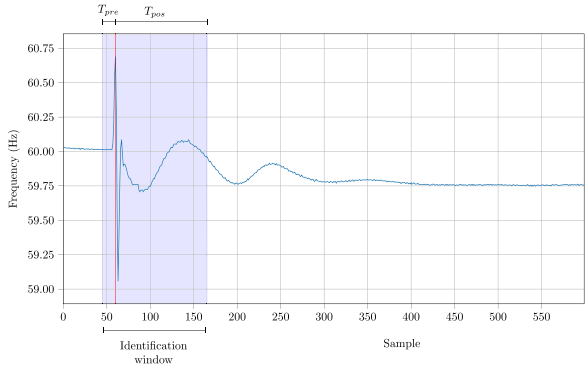


Fig. 24. New identification window after SHAP inspection.

Table 8
Best Hyper-parameters of LSTM-SHAP classifier.

n_h	Neurons	Dropout rate	α
3	60|50|80	[0.1, 0.5, 0.5]	0.55

Table 9
Performances of LSTM, LSTM-SHAP.

Classifier	GT%	LS%	LT%	OS%	IAR %	BA%
LSTM	97.959	100.0	94.444	100.0	98.182	98.101
LSTM-SHAP	**97.959**	**100.0**	**100.0**	**100.0**	**99.490**	**99.091**

Table 10
Misclassifications of LSTM, and LSTM-SHAP.

Classifier	GT	LS	LT	OS	Total
LSTM	1	0	0	1	2
LSTM-SHAP	**1**	**0**	**0**	**0**	**1**

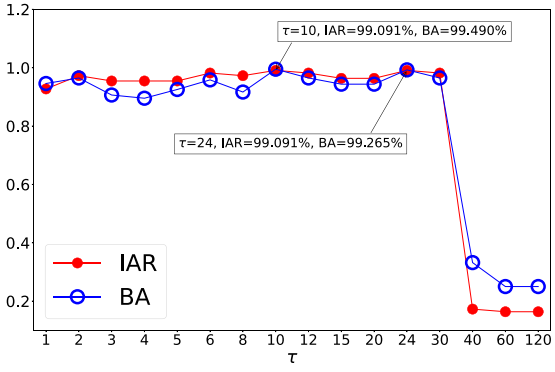


Fig. 25. Performances of LSTM-SHAP classifier in relation to the number of times-steps τ. Best performance at $\tau = 10$.

Table 11
Base values $E[f_\phi(x)]$ of LSTM-SHAP.

$E[f_\phi(x)]_{\text{GT}}$	$E[f_\phi(x)]_{\text{LS}}$	$E[f_\phi(x)]_{\text{LT}}$	$E[f_\phi(x)]_{\text{OS}}$
0.7724	0.1956	0.0249	0.0071

Fig. 22 displays the LT event, highlighting the main contributions. Also, Fig. 23 exhibits the force plot of the predicted class of LT presenting P(GT). It can be observed that the fall in the frequency after the spike has a negative contribution in identification of LT, making the LSTM to classify the event as GT due to the deviation.

One interesting point is that contribution of time-step t ($\Phi^{(t)}$) is the combination of samples contributions $\bar{\phi}_j^{(t)}$ in the same time-step t, and is more important in compare with the samples contributions $\bar{\phi}_j^{(t)}$.

7.3. Improvements of LSTM based on the interpretability inspection

Based on the inspection of the correct and incorrect classified events, some improvements can be applied in order to improve the performance of the classifier. First, the main identified problem is the bias for LT events, since based on EPS domain knowledge the spikes are supposed to be the most relevant parts in the identification of this type of event in compare with steady state part of time window. Second, the initial and final time-steps of the time-series are not useful for the classification of any event.

One clear strategy is reduction of the time-window. The following improvements are proposed to re-train the LSTM classifier: (1) the pre-time (T_{pre}) be reduce to 0.25s (15 samples). (2) the pos-event time (T_{pos}) be reduce from 9s to 1.75s (105 samples), as shown in Fig. 24.

This new LSTM classifier named LSTM-SHAP now is trained by not only looking at IAR of Test-set, but also inspecting the SHAP values and analyzing the obtained contributions. Therefore, we try different hyper-parameters in multiples initial conditions. The IAR of Test-set for the obtained models are compared along with SHAP values selecting the model with the higher IAR and contributions and mainly considering the spike to identify LT events. The hyper-parameters of the LSTM-SHAP classifier are presented at Table 8. The IARs of the LSTM and LSTM-SHAP are compared in Table 9, and misclassifications in Table 10. The results in Fig. 25 indicate that the IAR in the test-set of the LSTM-SHAP classifier is a function of τ. As presented, the LSTM-SHAP classifier has the best performance with $\tau = 10$. The misclassification labeled as event 7 will be inspected later.

In the same way as the inspection executed for the LSTM classifier we are going to inspect the predictions of the LSTM-SHAP classifier. Again, the base values of LSTM-SHAP $E[f_\phi(x)]$ (Table 11) are computed over the background data-sets.

7.3.1. Inspection of the Events using LSTM-SHAP

In this section, the same events discussed for the LSTM classifier are now presented using the LSTM-SHAP classifier.

GT events. **(1) Local Explanation:** The force plot of the Event 3 is presented in Fig. 26, given base value of $E[f_\phi(x)]_{\text{GT}}$. The event was correctly classified with a probability of P(GT) = 100%.

Also, Fig. 27 displays the Event 3, highlighting the main sample contributions. The $\Phi^{(t)}$ values were calculated using Eq. (28) with $\tau =$

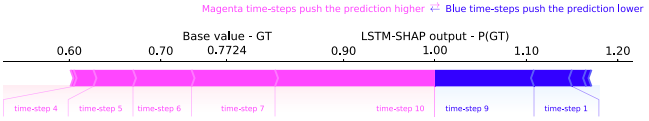


Fig. 26. Force plot of LSTM-SHAP for Event 3.

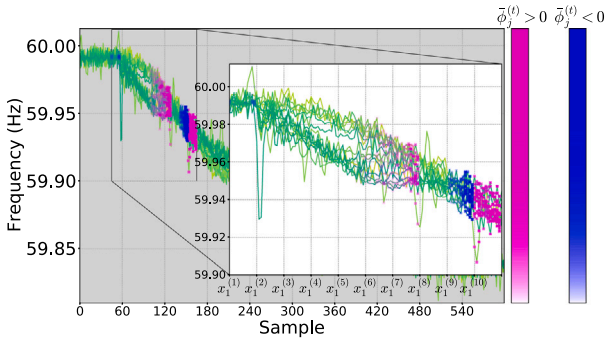


Fig. 27. Event 3. Frequency PMUs signals with the most relevant contributions highlighted. Base value $E[f_\phi(x)]_{\mathrm{GT}} = 0.7724$.

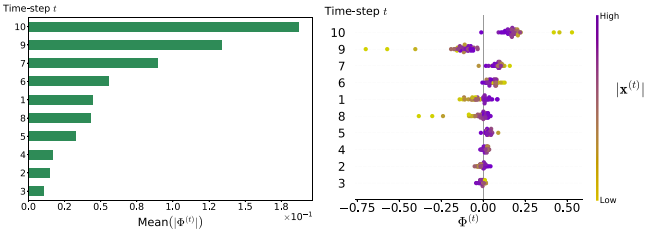


Fig. 28. Global explanation of LSTM-SHAP for GT events. (a) Bar chart of average $\boldsymbol{\Phi}^{(t)}$ magnitude. (b) Beeswarm plot of $\boldsymbol{\Phi}^{(t)}$.

10 and $m = \frac{120}{10} = 12$. The parts of the time-series with the most contribution to classification of this event still face the downfall in the frequency, identifying the deviation or drop similar to LSTM classifier. Although, the time-step $t = 9$ had the negative values for this case and did not affect the prediction P(GT).

(2) Global Explanation: The global explanation of LSTM-SHAP for GT events are presented in Fig. 28. Fig. 28(a) exhibits the bar plot of average SHAP magnitude and Fig. 28(b) presents beeswarm plot. The time-steps $t = 10, 9, 7$ have the most important contributions, where time-steps $t = 10, 7$ are positively captured the downfall in the frequency. Also, the events with lower values of $\left|\boldsymbol{x}^{(10)}\right|$ (in blue) have greater values of $\boldsymbol{\Phi}^{(10)}$, which is coherent with knowledge domain of GT event.

According to the beeswarm plot, it can be observed that the time-step $t = 9$ has a negative contribution, and that the contributions $\boldsymbol{\Phi}^{(9)}$ increase with higher values of $\left|\boldsymbol{x}^{(9)}\right|$. This inconsistency could cause misclassifications for events with higher Δf, depending on the values of other time-steps $t = 7, 6$. As will be discussed later, this is the reason for the misclassification of the Event 7.

LT events. **(1) Local Explanation:** The force plot for Events 39 is presented in Fig. 29, given the base value of $E[f_\phi(x)]_{\mathrm{LT}}$. This event was correctly classified with a probability P(LT) of 78%. Also, the time-series of this event is presented in Fig. 30 showing that the samples with most contribution to the correct identification are now the spikes (time-step $t = 2$ and $t = 3$). However, there are still contributions from the time-steps $t = 10, 8$. This inspection shows that the predictions of LSTM-SHAP are different from LSTM, which identified the LT from

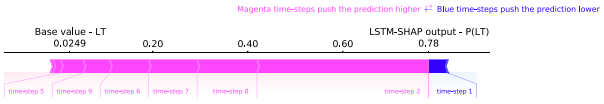


Fig. 29. Force plot of LSTM-SHAP for Event 39. Base value $E[f_\phi(x)]_{\mathrm{LT}}$.

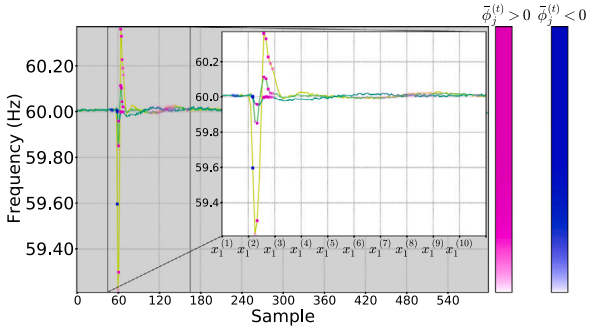


Fig. 30. Event 39. Frequency PMUs signals with the most relevant contributions highlighted. Base value $E[f_\phi(x)]_{\mathrm{LT}} = 0.0249$.

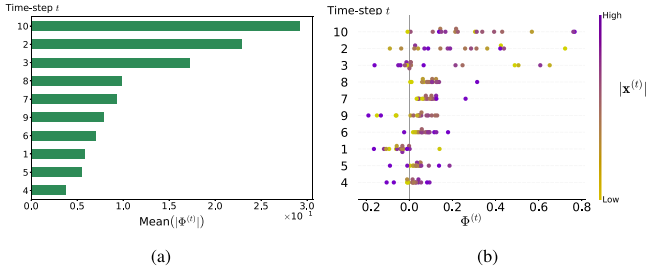


Fig. 31. Global explanation of LSTM-SHAP for LT events. (a) Bar chart of average $\boldsymbol{\Phi}^{(t)}$ magnitude. (b) Beeswarm plot of $\boldsymbol{\Phi}^{(t)}$.

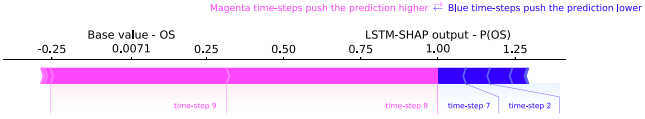


Fig. 32. Force plot of LSTM-SHAP for Event 48. Base value $E[f_\phi(x)]_{\mathrm{OS}} = 0.0071$.

the ambient data part. Now, the spikes play a significant role in the identification of LT.

(2) Global Explanation: Fig. 31(a) displays the bar plot of average SHAP magnitude, and Fig. 31(b) presents beeswarm plot using the LSTM-SHAP. The most important time-steps contributions are $t = 10, 2, 3$, both mostly with positive contributions. The time-step $t = 10$ represents the transitory after spike with the greatest contribution. Also, the spikes (time-step $t = 2$ and $t = 3$) are now having a relevant contribution to the identification, minimizing the bias presented to LSTM for the LT event, because the spikes had the lowest contribution (Fig. 17).

OS events. **(1) Local Explanation:** The Event 48 is inspected using following procedure. The event was correctly classified with a probability of P(OS) = 100%, and the force plot of this event is presented in Fig. 32, given the base value of $E[f_\phi(x)]_{\mathrm{OS}}$. Also, Fig. 33 displays the time-series of Event 48, highlighting the main sample contributions $\bar{\phi}_j^{(t)}$. The samples with the most contribution to correct identification face the minimum oscillation peak soon after the disturbance, representing by time-steps $t = 8, 9$ (Fig. 32).

(2) Global Explanation: The global explanation of LSTM-SHAP for OS events are presented in Fig. 34. It is clear that, the most important

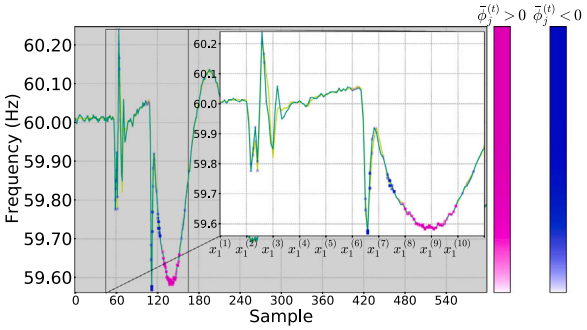
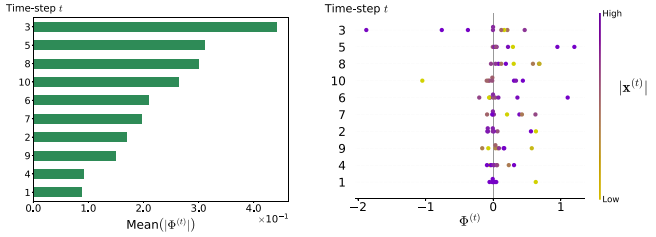


Fig. 33. Event 48. Frequency PMUs signals with the most relevant contributions highlighted. Base value $E[f_\phi(x)]_{OS} = 0.0071$.



(a) (b)

Fig. 34. Global explanation of LSTM-SHAP for OS events. (a) Bar chart of average $\Phi^{(t)}$ magnitude. (b) Beeswarm plot of $\Phi^{(t)}$.

time-steps are $t = 3, 5$ and 8, where the time-steps 5,8 have positive contributions related to the peaks. In most cases, $t = 5$ and $t = 8$ are related to first peak (zenith) and final peak of the window (nadir), respectively. The time-step $t = 3$ presents some very high negative SHAP values (Fig. 34(b)) that push the absolute value of global mean upward, this shows the reason that the time-step $t = 3$ presents the highest mean contribution. As observed from other OS events, this is related to the spikes that usually presenting in the oscillations events, since the spikes are important to identify LT and they contribute to push P(OS) down. Also, the negative contributions of $\Phi^{(3)}$ increase in magnitude with higher values of $\left|x^{(3)}\right|$ (higher spikes in frequency).

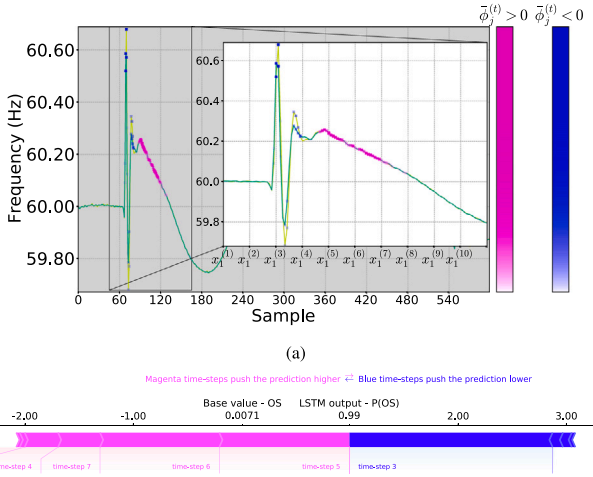
An example of correctly classified as OS is Event 58, presented in Fig. 35, with the force plot displayed in Fig. 35(b). This time-step $t = 3$ represents the spike with a negative contribution. Also, the time-steps $t = 5, 6$ represent the zenith which are relevant part for the correct classification.

In general, the LSTM-SHAP classifier is learning to identify the OS only by the lowest peak at the final time-step of the identification window. Even though the time-step $t = 3$ has a great negative contribution, this will not affect the P(OS) in most OS events. Briefly, the summation of other positive contributions overcome this negative contribution.

7.3.2. Inspection of the misclassified event (event 7) of LSTM-SHAP

The Event 7 was classified as OS with a probability of P(OS) = 54%, however the correct class for this event is GT. The $\Phi^{(t)}$ are computed from Eq. (13) with the base value of $E[f_\phi(x)]_{GT} = 0.7724$. These contributions are essential for the correct identification GT event, affecting P(GT). The output probability for GT is P(GT) = 46%. The force plot of this GT event is presented in Fig. 36. The Event 7 is presented in Fig. 37, highlighting the main sample contributions.

The time-steps $t = 10, 6, 7$ have positive contribution for classification of the event as GT, but the combined contribution of these time-steps had limitations to overcome the negative contributions (Fig. 36). The time-steps $t = 8, 9$, and especially the $t = 9$, present a high negative contribution that push P(GT) downward (Fig. 28) this is due to high deviation of Δf that presents lower values of $\left|x^{(9)}\right|$.



(a) (b)

Fig. 35. Event 58. (a) Frequency PMUs signals with the most relevant contributions highlighted. (b) Force plot of LSTM-SHAP for this event.

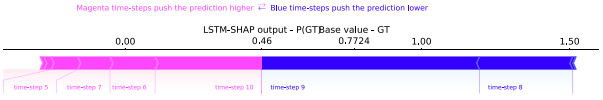


Fig. 36. Event 7. Force plot of LSTM-SHAP for the misclassified GT event.

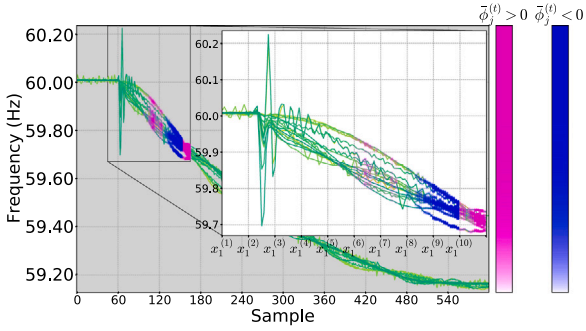


Fig. 37. Event 7. Misclassified GT event with the most relevant contributions highlighted. Base value of GT event $E[f_\phi(x)]_{GT} = 0.7724$.

In a practical power system operation, this type of GT with a deviation of $\Delta f = 0.8$ Hz is extremely rare and unusual. So, this inconsistency is not very problematic for most GT events, making the LSTM-SHAP classifier an appropriate candidate for most real cases. Also, if others GT events have a deviation greater or equal to 0.8 Hz we have a good indication in not trusting the prediction.

7.4. Discussion

The LSTM-SHAP presented a higher BA than LSTM, but above all else the predictions are more coherent with the knowledge of power system events. Furthermore, we were able to minimize the bias for LT in the LSTM classifier. However, there is still an inconsistency with GT events in time-step $t = 9$. According to our observations, these types of inconsistency were normal in all ANN trained models and were generally related to the misclassifications. The positive side is that SHAP inspection allow us to discover these inconsistencies the biases, and as a result be aware about the limitations of these models. This inconsistency also to be solved (or minimize) using the same procedure described in 7.3 by focusing on this inconsistency.

The proposed approach has the main advantage of being explainable, when compared with others traditional methods. It provides explanations to specific predictions (local explanation) and globally. The explanations about the predictions can help understand about how the classifier is taking the predictions. Also, using the interpretability inspection of SHAP values it is possible to highlight the main parts of time-series that are important to the identification of the events. By inspecting the classifier we are able to identify bias and inconsistencies making possible to correct and improve the performance and coherency of the predictions.

8. Conclusion

In this paper, we propose the application of an interpretability inspection of the LSTM for event identification using real events records of the BIPS. The inspection of the LSTM was performed using the SHAP values method, more specify with the DeepSHAP method. The method was able to provide consistencies explanations about the LSTM, both locally and globally. Also, the interpretability inspection was very important to understand and certify that the LSTM is getting a consistent and coherent performance. Also, identifying the most relevant parts of the time-series. By the knowledge extracted from the inspection, we were able to detect the bias of the LSTM classifier for LT events. Also, the classifier was re-trained with improvements in the input data making a new model LSTM-SHAP. This new classifier not only had a greater IAR and BA, but also had a more consistent and coherent performance, correcting the detected bias.

The proposed approach was also validated using real PMU measurements collected from the BIPS. It shows that the method is suitable for application in practical systems in order to improve the training process as well as reducing the concerns of operators to how the classifier is taking its decisions. Finally, the method is also useful for real-time and online applications since the classifier can make a prediction around 0.03 s (average).

Future directions include extending to other DNN models such as Gated Recurrent Unit (GRU), CNN and transformer-based architectures such T5 (Text-to-Text Transfer Transformer) and BERT (Bidirectional Encoder Representation from Transformers) [45,46]. Additionally, it might be useful to incorporate the SHAP values into the loss function of the LSTM introducing the knowledge domain as constraints, improving the generalization performance of the model, and coherent of predictions.

CRediT authorship contribution statement

Orlem L.D. Santos: Conception and design of study, Acquisition of data, Analysis and/or interpretation of data, Writing – original draft, Writing – review & editing. **Daniel Dotta:** Conception and design of study, Analysis and/or interpretation of data, Writing – original draft, Writing – review & editing. **Meng Wang:** Conception and design of study, Writing – review & editing. **Joe H. Chow:** Conception and design of study, Writing – review & editing. **Ildemar C. Decker:** Conception and design of study, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors gratefully acknowledge FAPESP under grants no. 2016/08645-9, 2018/20104-9, 2017/25425-5, 2019/08200-5 and 2019/10033-03. All authors approved the version of the manuscript to be published.

References

[1] Rafferty M, Liu X, Laverty DM, McLoone S. Real-time multiple event detection and classification using moving window PCA. IEEE Trans Smart Grid 2016;7(5):2537–48.
[2] Jena MK, Panigrahi BK, Samantaray SR. A new approach to power system disturbance assessment using wide-area postdisturbance records. IEEE Trans Ind Inf 2018;14(3):1253–61.
[3] Yadav R, Pradhan AK, Kamwa I. Real-time multiple event detection and classification in power system using signal energy transformations. IEEE Trans Ind Inf 2018.
[4] Bengio Y, Courville A, Vincent P. Representation learning: A review and new perspectives. IEEE Trans Pattern Anal Mach Intell 2013;35(8):1798–828.
[5] Ng A. Machine learning and AI via brain simulations. 2018, [Accessed 3 May 2013].
[6] LeCun Y, Bengio Y, Hinton G. Deep learning. Nature 2015;521:436–44. http://dx.doi.org/10.1038/nature14539.
[7] Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. Adv Neural Inf Process Syst 2012;25:1097–105.
[8] Li F, Du Y. From alphago to power system AI: What engineers can learn from solving the most complex board game. IEEE Power Energy Mag 2018;16(2):76–84.
[9] Miranda V, Cardoso PA, Bessa RJ, Decker I. Through the looking glass: Seeing events in power systems dynamics. Int J Electr Power Energy Syst 2019;106:411–9.
[10] Li W, Wang M. Identifying overlapping successive events using a shallow convolutional neural network. IEEE Trans Power Syst 2019.
[11] Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput 1997;9(8):1735–80.
[12] Mou L, Ghamisi P, Zhu XX. Deep recurrent neural networks for hyperspectral image classification. IEEE Trans Geosci Remote Sens 2017;55(7):3639–55.
[13] Young T, Hazarika D, Poria S, Cambria E. Recent trends in deep learning based natural language processing. IEEE Comput Intell Mag 2018;13(3):55–75.
[14] Zuo L-Q, Sun H-M, Mao Q-C, Qi R, Jia R-S. Natural scene text recognition based on encoder-decoder framework. IEEE Access 2019;7:62616–23.
[15] Sutskever I, Vinyals O, Le QV. Sequence to sequence learning with neural networks. In: Advances in neural information processing systems. 2014, p. 3104–12.
[16] Zazo R, Nidadavolu PS, Chen N, Gonzalez-Rodriguez J, Dehak N. Age estimation in short speech utterances based on LSTM recurrent neural networks. IEEE Access 2018;6:22524–30.
[17] Wen S, Wang Y, Tang Y, Xu Y, Li P, Tianyang Z. Real-time identification of power fluctuations based on LSTM recurrent neural network: A case study on Singapore power system. IEEE Trans Ind Inf 2019.
[18] Kong W, Dong ZY, Jia Y, Hill DJ, Xu Y, Zhang Y. Short-term residential load forecasting based on LSTM recurrent neural network. IEEE Trans Smart Grid 2017;10(1):841–51.
[19] Yu R, Gao J, Yu M, Lu W, Xu T, Zhao M, et al. LSTM-EFG For wind power forecasting based on sequential correlation features. Future Gener Comput Syst 2019;93:33–42.
[20] Chatterjee S, Archana V, Suresh K, Saha R, Gupta R, Doshi F. Detection of non-technical losses using advanced metering infrastructure and deep recurrent neural networks. In: 2017 IEEE international conference on environment and electrical engineering and 2017 IEEE industrial and commercial power systems Europe. IEEE; 2017, p. 1–6.
[21] Zhang S, Wang Y, Liu M, Bao Z. Data-based line trip fault prediction in power systems using lstm networks and SVM. IEEE Access 2017;6:7675–86.
[22] Montavon G, Lapuschkin S, Binder A, Samek W, Müller K-R. Explaining nonlinear classification decisions with deep taylor decomposition. Pattern Recognit 2017;65:211–22.
[23] Lundberg SM, Lee S-I. A unified approach to interpreting model predictions. In: Advances in neural information processing systems. 2017, p. 4765–74.
[24] Molnar C. Interpretable Machine Learning. 2019, https://christophm.github.io/interpretable-ml-book/.
[25] Delgado Zarzosa MA, et al. Análise de eventos em sistemas de energia elétrica usando sincrofasores [Master's thesis], Universidade Federal de Santa Catarina; 2016.
[26] Decker IC, e Silva AS, Agostini MN, Prioste FB, Mayer BT, Dotta D. Experience and applications of phasor measurements to the Brazilian interconnected power system. Eur Trans Electr Power 2011;21(4):1557–73.
[27] Goodfellow I, Bengio Y, Courville A. Deep learning. MIT Press; 2016, http://www.deeplearningbook.org.
[28] Karim F, Majumdar S, Darabi H, Harford S. Multivariate LSTM-FCNs for time series classification. Neural Netw 2019;116:237–45.
[29] Kononenko I, et al. An efficient explanation of individual classifications using game theory. J Mach Learn Res 2010;11(Jan):1–18.
[30] Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, et al. TensorFlow: Large-scale machine learning on heterogeneous systems. 2015, Software available from tensorflow.org. URL https://www.tensorflow.org/.

[31] Cubuk ED, Zoph B, Mane D, Vasudevan V, Le QV. Autoaugment: Learning augmentation policies from data. 2018, arXiv preprint arXiv:1805.09501.
[32] Tieleman T, Hinton G. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Netw Mach Learn 2012;4(2):26–31.
[33] Glorot X, Bengio Y. Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the thirteenth international conference on artificial intelligence and statistics; 2010. p. 249–56.
[34] Zaremba W, Sutskever I, Vinyals O. Recurrent neural network regularization. 2014, arXiv preprint arXiv:1409.2329.
[35] Gal Y, Ghahramani Z. A theoretically grounded application of dropout in recurrent neural networks. In: Advances in neural information processing systems. 2016, p. 1019–27.
[36] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 2014;15(1):1929–58.
[37] Ribeiro MT, Singh S, Guestrin C. Why should i trust you?: Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. ACM; 2016, p. 1135–44.
[38] Shrikumar A, Greenside P, Kundaje A. Learning important features through propagating activation differences. In: Proceedings of the 34th international conference on machine learning-volume 70. JMLR. org; 2017, p. 3145–53.
[39] Bach S, Binder A, Montavon G, Klauschen F, Müller K-R, Samek W. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. PLoS One 2015;10(7):e0130140.
[40] Serrano R. Cooperative games: Core and Shapley value. Tech rep., Working Paper; 2007.
[41] Amboni MK, et al. Alocação do sobrecusto operativo de sistemas de energia elétrica via teoria dos jogos. 2001.
[42] Lundberg SM, Nair B, Vavilala MS, Horibe M, Eisses MJ, Adams T, et al. Explainable machine-learning predictions for the prevention of hypoxaemia during surgery. Nat Biomed Eng 2018;2(10):749.
[43] Chen H, Lundberg S, Lee S-I. Explaining models by propagating Shapley values of local components. 2019, arXiv preprint arXiv:1911.11888.
[44] Lundberg SM, Erion G, Chen H, DeGrave A, Prutkin JM, Nair B, et al. From local explanations to global understanding with explainable AI for trees. Nat Mach Intell 2020;2(1):2522–5839.
[45] Raffel C, Shazeer N, Roberts A, Lee K, Narang S, Matena M, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. 2019, arXiv preprint arXiv:1910.10683.
[46] Devlin J, Chang M-W, Lee K, Toutanova K. Bert: Pre-training of deep bidirectional transformers for language understanding. 2018, arXiv preprint arXiv:1810.04805.