Received: 6 April 2021 | Revised: 30 June 2021 | Accepted: 1 July 2021

DOI: 10.1002/int.22581

RESEARCH ARTICLE

WILEY

# A stacked autoencoder-based convolutional and recurrent deep neural network for detecting cyberattacks in interconnected power control systems

Gianni D'Angelo | Francesco Palmieri

Department of Computer Science, University of Salerno, Fisciano (SA), Italy

**Correspondence**
Gianni D'Angelo, Department of Computer Science, University of Salerno, Via Giovanni Paolo II, 132, 84084 Fisciano (SA), Italy.
Email: giadangelo@unisa.it

**Abstract**
Modern interconnected power grids are a critical target of many kinds of cyber-attacks, potentially affecting public safety and introducing significant economic damages. In such a scenario, more effective detection and early alerting tools are needed. This study introduces a novel anomaly detection architecture, empowered by modern machine learning techniques and specifically targeted for power control systems. It is based on stacked deep neural networks, which have proven to be capable to timely identify and classify attacks, by autonomously eliciting knowledge about them. The proposed architecture leverages automatically extracted spatial and temporal dependency relations to mine meaningful insights from data coming from the target power systems, that can be used as new features for classifying attacks. It has proven to achieve very high performance when applied to real scenarios by outperforming state-of-the-art available approaches.

KEYWORDS
anomaly detection, convolutional neural networks, cyber-attacks, IoT, long short-term memory, power grids, recurrent neural networks, remote control systems

## 1 | INTRODUCTION

With the increasing success of the Internet of Things (IoT) and automation technologies, most of the industrial facilities and public service utilities, such as manufacturing plants, electricity, water and gas distribution infrastructures, transportation control and monitoring systems, and many others have been interconnected through public and private cyber-infrastructures, providing advanced automation control and interoperation capabilities. Despite the obvious advantages in terms of technological advancement, such practices have nonnegligible effects on the attack surface exposed by modern cyber-physical infrastructures. We remark that nowadays, such infrastructures include sophisticated field sensor or actuator devices as well as supervisory control and data acquisition (SCADA) systems, often involving legacy hardware or software solutions with a limited resistance and robustness to external cyber-attacks or generic abuses.

In this scenario, the electrical power grid, characterized by a significant improvement in automation and remote control facilities, becomes one of the most critical targets for attacks, potentially affecting public safety and hence involving homeland security-level attention. These infrastructures now include many smart control and metering devices, such as phasor measurement units (PMUs), digital fault recorders, and so on, providing a huge amount of data about the overall system's state and operations, supporting automation decisions and fault monitoring. Specific power system events, like transmission line faults, that can be spontaneous or originated by compromised control devices sending fraudulent control commands or measurement data, can start a reaction chain leading to cascading blackouts or critical equipment faults, if proper reactions take not place in a timely way. More specifically, attacks against power distribution infrastructures operate by exploiting vulnerabilities in remote management or metering devices and corrupting control signals to introduce faults, impersonate disturbances, or fraudulently performing wrong control actions.[1] Proper situational awareness capabilities are needed to analyze these phenomena and discriminate attacks from real disturbance or faults occurring during the normal system operations. This is not an easy task due to both the overall system complexity and the huge variety of attacks available as well as to their unpredictability. Furthermore, some specific attacks can be only spotted by considering the occurrence of a combination of specific events over time, and consequently stateful and context-dependent analysis and inference capabilties are needed exploring correlation between events/observations in both space and time. Consequently, a new generation of anomaly detection systems is needed, capable of recognizing and classifying attacks from disturbances or faults or legitimate control actions, for timely generating alerts to operators as well as automatically triggering the proper actions when an attack is discovered. To provide the needed versatility and situational awareness, these systems must be empowered by the most advanced artificial intelligence (AI) and machine learning (ML) technologies, to significantly outperform the traditional solutions based on the mining and analysis of specific field-related features. These systems must be able to exploit *spatial* and *temporal* correlations among all the different observations characterizing the system activities by autonomously eliciting new field-related knowledge. Besides, the elicited knowledge consists of new (presumably unknown) features capable to describe unambiguously the attacks and the normal power systems operations by also correctly classifying them.

To this purpose, we propose a complex deep neural network (DNN) framework able to mine meaningful information from the power system monitoring data regarding both relations among the different observations and their correlations over time, represented as spatial and temporal dependencies, constituting more expressive and representative features to be used for attack detection. As it is described below, this is accomplished through the usage of two sparse autoencoders (SAEs) whose encoder–decoder functions are replaced with two DNNs, respectively a convolutional and a recurrent ones, providing the ability of capturing the aforementioned correlations between their inputs. Although the convolutional and recurrent networks have proven to be effective also without being used in combination with the autoencoders, their performance tends to decrease if they are trained on an unbalanced data set. Furthermore, for a high number of hidden layers, such networks are affected by the well-known problem of the vanishing gradient.[2] On the contrary, the stacked autoencoder-based deep network solves many issues related to the training of networks enclosing a high number of layers and offers a powerful tool that can be used also in different fields of application. Indeed, since our approach does not preliminarily require any specific field-related knowledge about the system of interest as well as about the kind of devices of interest and generates all the knowledge needed for its operations at the training time, it can be ideally used for any kind of industrial control system (ICS) involving the production of a sufficiently expressive amount of time-sequenced system monitoring observations. The complexity of sophisticated (and hidden) dependencies between events occurring over time as well as among the values of many apparently unrelated observations, that does not emerge with more traditional analysis techniques, can be easily captured by the combination of the stacked networks. Excellent performance evaluation results demonstrated that the proposed contribution and ideas are extremely promising for further investigation and implementation in real-world production scenarios to improve the dependability of critical infrastructures.

The main contributions of the paper are summarized as follows:

- A stacked-autoencoder-based convolutional and recurrent neural network is used for detecting cyberattacks in interconnected power control systems.
- Meaningful information is mined from the power system monitoring data, which are used for inferring the operation status of the control system.
- The elicited knowledge is able to describe unambiguously the attacks and the normal power systems operations.
- Such information is expressed by using spatial and temporal features, which are able to capture correlations existing among the data and among them over time.
- A detailed mathematical description of the network architecture is provided to highlight the effects of any network component in mining the aforementioned features.

The remainder of the paper is organized as follows. In Section 2, the related works are shown. A detailed description of the proposed architecture is reported in Section 3. The experiments and their results are discussed in Section 4. Finally, concluding remarks are provided in Section 5.

## 2 | RELATED WORK

The reliable and timely detection of attacks and anomalous events is an extremely challenging issue and hence a hot topic in IoT and ICS research arena. An anomaly detection scheme for power systems based on whitelisting legitimate transactions within control-layer communications has been presented in Reference [3]. However, this approach is not able to detect fraudulently injected valid sequences of commands that aim at tripping protection relays to cause a blackout. Other detection solutions are based on power system-related theoretical issues such as the one proposed in Reference [4] that by leveraging optimal power flow programming allow the detection of attacks that cause wrong power flow dispatching through the alteration of service measurements. Similarly, the approach presented in Reference [5] is able to spot attacks based on the injection of malicious data for tampering state variables in the power system, through proper weighted state estimation techniques. Also the unsupervised detection solution described in Reference [6] is aimed to spot false data injection attacks in power systems by analyzing historical data and recognizing state vectors the deviate respect to normal trends. Clearly, the above approaches lack of generality and their scope is limited to the attacks on which they are specifically targeted, without any chance of extension to other ones. Host-based detection approaches for intelligent electronic devices (IEDs) operating within smart grids have been proposed in References [7] and [8]. Unfortunately, these localized approaches are only able to identify attacks against individual device without a global and more articulated vision of the phenomena of interest. To determine causal relationships between multiple different observations/information and temporal state transitions the approach presented in Reference [9] leveraged a Bayesian network, resulting in a specification-based detection framework capable of detecting attacks and classifying several substation scenarios. Temporal state-based specifications and sequential patterns have been also used for classifying attacks and power disturbance events[10] with heterogeneous time-synchronized observations.[11] However, all the above approaches based on mining large amounts of data for determining the nature of power system events may be not suitable for timely spotting phenomena of interest from measurements coming from synchro-phasors since all the data to be analysed has to be available in memory to perform pattern classification. However, the combination of more traditional data-oriented solution, leveraging physics-based PMU features, with ML capabilities may result in a more promising approach.[12] Indeed, the use of ML technology is a quite effective approach for timely detection of attacks against industrial power control systems. The authors in Reference [13] extensively explored ML potentialities in classification of power system disturbances, by explicitly focusing on discriminating attacks on power grids. A a kernel ML method has been used in Reference [14] for baselining SCADA systems with the aim of isolating intrusions and faults. However, due to the lack of attack data, such approach resulted of limited effectiveness in discriminating attacks. An ML-based model leveraging autoencoders for detecting PMU manipulation attacks has been presented in Reference [15], whereas two ML-based schemes for detecting stealth attacks in smart power grids are reported in Reference [16]. The first one is based on principal component analysis (PCA) and distributed SVMs and the second, unsupervised, detects measurement deviations. An artificial neural network (ANN)-based approach for baselining power systems at the substations level and generating alarms in presence of anomalous outliers has been presented in Reference [17]. The right choice of features, assumes paramount importance in guaranteeing the success of these approaches. A convolutional neural network (CNN) has been used in Reference [18] together with random forests for learning features to be used in classification from convolution and downsampling of massive smart metering data, with the purpose of detecting electricity theft events. Finally, an approach based on a stacked long short term memory (LSTM) network and softmax classifier has been proposed in Reference [19] for multilevel anomaly detection in ICS. Nevertheless, most of these works have shown a reduction in performance when applied to unbalanced data sets, and also have proven to be time-expensive in the training phase. On the contrary, the experimental results related to our proposal have shown that the achieved performance of the proposed architecture remains steady with respect to the different application scenarios and training/testing data sets.

## 3 | THE ANOMALY DETECTION FRAMEWORK

In this section, the proposed anomaly detection architecture is presented in detail. It is essentially focused on building a DNN able to mine meaningful evidence to be used for detection (i.e., the features) from measurement data related to ICS activity to distinguish between normal and anomalous situations/behaviors associated with disturbance, control, and cyber-attack events. To accomplish this, relations existing among the power system monitoring data (referred to as status variables from here on) and their correlations over space and time are extracted. We refer to them as *spatial* and *temporal* dependencies, respectively. More precisely, the aim of the spatial dependencies is to discover relevant hidden insights capable to represent useful relations among all the involved status variables. On the other hand, the temporal ones are used for learning higher-order dependencies that may exist among a set of status variables over time.

As widely described in the scientific literature, CNNs have proven to be the best candidate in extracting spatial insights from data, especially in computer vision and image processing fields.[20] As it will be shown below, the usage of several stacked convolutional layers in CNNs, used to create a hierarchy of progressively more abstract representations of the input data, constitutes their main strength for mining spatial dependencies. Similarly, recurrent neural networks (RNNs) are the most suitable neural network typology for modeling temporal dependencies in time-series data.[21]

Ultimately, the combination CNN-RNN in a unified network is able to elicit temporal dependencies among spatial relations extracted by a convolutional-based network. This type of combined network is already widely used in the state-of-the-art related to several scientific and industrial fields,[22] where modeling spatio-temporal information is crucial, such as financial applications, speech recognition, weather forecasting, and more.

Nevertheless, training may be almost time-expansive for high-dimensional data. Besides, the well-known problem of the vanishing gradient[2] for networks enclosing numerous layers could cause a significant reduction in performance. To overcome these drawbacks, known also as the curse of dimensionality problem,[23] we used the CNN and RNN as encoder/decoder stages in two distinct SAEs. Each SAE is trained separately and then, along with a softmax-based classifier network, they are stacked to obtain a unified network.

SAEs are mainly used for learning a compressed representation of a set of data, namely for a dimensionality reduction. Unlike PCA, SAEs are able to learn nonlinear transfer functions. This ability leads to represent the input data in a more meaningful state space, which could better offer the possibility to extract more representative features from data, better suitable for classification purposes. As depicted in Figure 1, the proposed detection architecture makes use of two SAEs, named CNN-SAE and RNN-SAE, respectively. The encoder–decoder function of the former encloses a CNN, and it is trained on status variables values arranged as a two-dimensional (2D) image-like matrix. The latter SAE uses an RNN for the encoder–decoder functions, and it is trained on the data coming from the latent space of the first SAE. Next, the encoder stages of both SAEs are stacked to form a unique network, whose output is fed to a classifying network that will perform final attack detection.

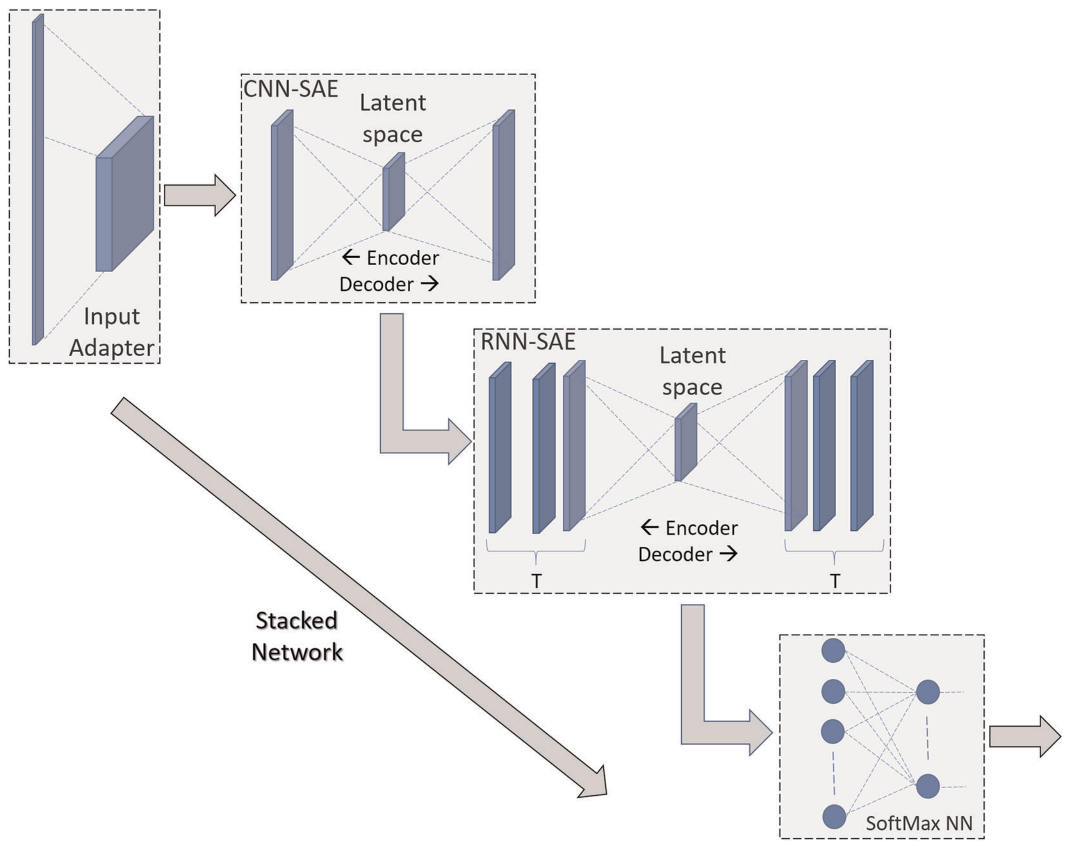

FIGURE 1 The proposed architecture. It includes an input adapter that serves as a dimensional transformer of the input data, two SAEs, and a softMax full-connected neural network. CNN-SAE uses a CNN for its encoder, while RNN-SAE makes usage of an long short term memory network. Both SAEs are trained separately, and then their encoders are being stacked along with the input stage and the softMax network to form the final network. CNN, convolutional neural network; RNN, recurrent neural network; SAE, sparse autoencoder [Color figure can be viewed at wileyonlinelibrary.com]

As it will be shown from the theoretical point of view, the proposed architecture firstly extracts compressed spatial relations from system variables through the CNN-SAE, next, a compact representation of their behavior over time (temporal dependencies) is derived by the RNN-SAE. Finally, a softmax classifier, using all the extracted elements as its input features, concludes the network. As reported in Reference [24], such an approach has proven to be effective also in network traffic classification. The mathematical characterization of each component is provided in the following to ease the understanding of the deepest architectural insights, starting from its theoretical bases.

## 3.1 | Input adaptation

Usually, the raw data captured by an industrial power control system are provided as a set of time-series *status variables* arranged as a one-dimensional (1D) vector, each related to a given timestamp. Let $d$ be such a dimension, the $d$-dimensional input needs to be arranged to a 2D matrix for being fed into 2D-CNN. Accordingly, let $x \in \mathcal{R}^d$ be the raw input and let $d' \equiv (N_x \times N_y)$ be a derived 2D-matrix, then the input to CNN can be expressed as follows:

$$x^{[\gamma]} \in \mathcal{R}^{d'} \tag{1}$$

where the superscript notation $*^{[\gamma]}$ refers to all the specific parameters related to the CNN network.

## 3.2 | Sparse autoencoder

With reference to the general autoencoder (AE) architecture, ENotice that here we have replacedquation (2) expresses the overall input–output transfer function. As it can be observed, the input ($x^{[\alpha]} \in \mathcal{R}^d$) is fed to the hidden layer, whose output ($h$, the latent space), is used to reconstruct ($\hat{x}^{[\alpha]}$) the input through the output layer ( $y$). We use the same superscript notation $*^{[\alpha]}$ referring to all the specific parameters related to the AE network.

$$\hat{x}^{[\alpha]} = y_{(W',b')}(h_{(W,b)}(x^{[\alpha]})) \equiv x^{[\alpha]}, \tag{2}$$

here $(W, b)$ and $(W', b')$ represent the weighting and biasing vectors associated with the encoder and decoder functions, respectively. For $s$ hidden neurons, then $W \in \mathcal{R}^{s\times d}$ and $b \in \mathcal{R}^s$, and the output of the latent space ($h_{(W,b)}$) can be expressed as follows:

$$h_{(W,b)}(x^{[\alpha]}) = \sigma(Wx^{[\alpha]} + b) \tag{3}$$

where $\sigma$ is the activation function.

As it is known, to obtain a compact representation from data, SAEs are used.[25] The sparsity property is obtained by training the network so that only a small number of hidden neurons are being simultaneously activated. In the extreme case, in which only one neuron is active at a time, the activation of a hidden neuron is caused by a specific content (features) in the input. Consequently, it is possible to capture, in theory, as many numbers of content-features as the number of hidden neurons.

Indeed, according to Equation (3), the output of the $l$th hidden neuron ($h_l$) of an SAE is given by:

$$h_l = \sigma\left(\sum_{k=1}^{d} w_{l,k} x_k^{[\vartheta]} + b_l\right) \tag{4}$$

with $w_{l,k} \in W$ and $x_k^{[\vartheta]}$ the $k$th component of $x^{[\vartheta]}$. Notice that here we have replaced the superscript notation $*^{[\alpha]}$ with $*^{[\vartheta]}$ to refer to the SAE.

If $||x^{[\vartheta]}||^2 \leq 1$, then the $k$th component ($x_k^{[\vartheta]}$) of the input ($x^{[\vartheta]}$) capable to activate the $l$th hidden neuron can be expressed by:

$$x_k^{[\vartheta]} = \frac{w_{l,k}}{\sum_{m=1}^{d} (w_{l,m})^2}, \quad \forall\ k = 1 \dots d. \tag{5}$$

That is, without considering the constant term associated with the denominator, the weights ($w_{l,k}$), related to the $k$th hidden neuron, identify a specific features-vector representing a specific content enclosed into raw data. In other words, the set of $w_{l,k}$ for a given $k$ corresponds to the features that we were looking for.

Nevertheless, as it is shown below, several transformations can be applied to these weights by using the CNN and RNN as encoder–decoder functions. Such derived weights are able to extract more complex content from data, and in particular to mine spatial and temporal dependencies to be used as detection features.

## 3.3 | CNN-SAE

The convolution operation between an input matrix ($x^{[\gamma]} \in \mathcal{R}^{N_x \times N_y}$) and $N_f$ bidimensional filters, whose dimensions are $P$ and $Q$ ($K^f \in \mathcal{R}^{P \times Q}$), is defined as follows:

$$Y_{i,j} = \sum_{f=1}^{N_f} \sum_{p=1}^{P} \sum_{q=1}^{Q} K^f_{p,q} \quad x^{[\gamma]}_{i+p-1,j+q-1} \tag{6}$$

with $Y_{i,j}$ being the components of the convolutional operation.

Note that the components of $x^{[\gamma]}$, depicted in Equation (6), are directly related to the components of the 1D vector $x$, according to the following relation:

$$x_k \equiv x^{[\gamma]}_{i,j} \tag{7}$$

with $k = 1 \dots d$, $i = 1 \dots N_x$, $j = 1 \dots N_y$, and $d = N_x \times N_y$.

Let $\varphi(k) = (i + p - 1, j + q - 1)$ be an analytic relation associated to Equation (7), and replacing Equation (6) into Equation (4), we have:

$$h_l = \sigma\left(\sum_{k=1}^{d} w'_{l,k} x_{\varphi(k)} + b_l\right) \tag{8}$$

with:

$$w'_{l,k} = \sum_{f=1}^{N_f} \sum_{p=1}^{P} \sum_{q=1}^{Q} K^f_{p,q} w_{l,k} \tag{9}$$

As depicted, the weights derived by using a CNN as encoder–decoder function in the SAE express a more complex relation involving the feature extracted by the latent space of a pure SAE. Indeed, as it can be observed from Equation (9), the new features $w'_{l,k}$ are expressed as a linear combination of the basic ones, $w_{l,k}$ of Equation (4). As a consequence, we can assert that features of Equation (9) are able to capture the aforementioned spatial dependencies from the input data.

## 3.4 | RNN-SAE

RNNs are networks that are widely used for mining temporal dependencies among data by maintaining the memory of the information over time. Nevertheless, as already mentioned, due to the vanishing gradient problem, the training of RNNs could be difficult, and in some cases impossible. LSTM and gated recurrent unit (GRU) are powerful alternatives capable to overcome this issue.[26] A single LSTM unit includes a cell of memory maintaining information for sufficiently time intervals. A set of gates are used to control the information flow and to make decisions about what data should be put into memory, and when they need to be forgotten. GRUs are similar to LSTMs but they use a simplified structure comprising fewer gates to control the flow of information. As depicted in Figure 2, a LSTM unit includes different gates which can be described by the following equations:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{10}$$

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{11}$$

$$\widehat{C}_t = \tanh(W_c[h_{t-1}, x_t] + b_c) \tag{12}$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{13}$$

$$C_t = f_t \otimes C_{t-1} + i_t \otimes \widehat{C}_t \tag{14}$$

$$h_t = o_t \otimes \tanh(C_t) \tag{15}$$

where $\otimes$ indicates the element-wise product, $W_f, W_i, W_c, W_o$ are matrices that act as linear transformations on data, $C_t$ and $h_t$ are, respectively, the state of the memory and the output at a given timestamp $t$. Note that the output $(h_t)$ is time-dependent and is related to the state that the cell assumes in the previous timestamps.

LSTM accepts as input a time-series of data of a given length $T$, and the final output is produced only after all the input data have been examined. Consequently, the equations from (10) to (15) are continuously and recursively updated for $T$ timestamps.

Let $t \in \{1, ..., T\}$ be the timestamps at which the input data are being considered, and by indicating with $x^{(t)}$ the input at the timestamp $t$, then the output $(y^T)$ at the timestamp $T$ can be expressed by:

$$y^T = \Psi^T(x^{(1)}, x^{(2)}, ..., x^{(T)}) \tag{16}$$

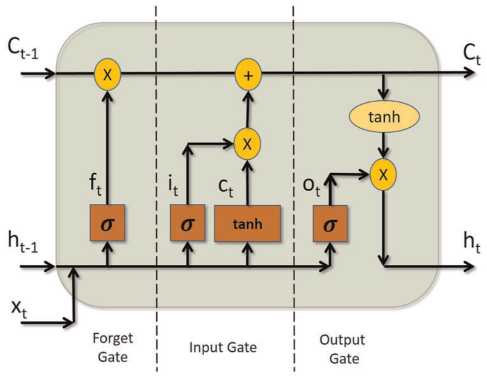

FIGURE 2 Long short-term memory unit. A set of gates are used to control the information flow [Color figure can be viewed at wileyonlinelibrary.com]

where $\Psi^T$ expresses the updating of equations from (10) to (15) of the unit for $T$ timestamps as a nonlinear multivariable vector-valued function.

Supposing $y^T$ be a $(r \times 1)$-dimensional vector, by replacing $x_k^{[\vartheta]}$ of Equation (4) with the $k$th component of $y^T$ (i.e., $y_k^T$), the output of the hidden neuron $l$ of the SAE is given by:

$$h_l = \sigma\left(\sum_{k=1}^{r} w_{l,k} y_k^T + b_l\right) \tag{17}$$

Comparing Equation (17) with Equation (4), it can be noted that the derived features gather the behavior of the input over time ($T$ timestamps) in a compact form.

A first-order Taylor approximation of $\Psi^T$ can help to better understand how the derived features are related to the basic ones provided by the pure SAE of Equation (4).

To accomplish this, let $\Psi_k^T(X)$ be the $k$th component of $\Psi^T$, with $X$ the set of the input $x^{(t)}$ for $t = 1, \ldots, T$, then according to Taylor's theorem, the first-order expansion, at null initial hypothesis (i.e., $\bar{X} = \{0\}^{T\times d}$), of $\Psi_k^T$ is given by:

$$\Psi_k^T(X) \approx \Psi_k^T(\bar{X}) + \nabla\Psi_k^T(\bar{X})[X - \bar{X}] \tag{18}$$

Note that at the initial stage of operation, the cell outputs expressed by Equations (14) and (15) are null, then $\Psi_k^T(\bar{X}) = 0$, and by considering that $\Psi_k^T(X)$ is a multivariable scalar-valued function, we have:

$$\nabla\Psi_k^T(\bar{X}) = \left[\frac{\partial\Psi_k^T(\bar{X})}{\partial x^{(1)}}, \frac{\partial\Psi_k^T(\bar{X})}{\partial x^{(2)}}, \ldots, \frac{\partial\Psi_k^T(\bar{X})}{\partial x^{(T)}}\right] \tag{19}$$

which replaced in Equation (18) it yields:

$$\Psi_k^T(X) \approx \sum_{t=1}^{T} \frac{\partial\Psi_k^T(\bar{X})}{\partial x^{(t)}} x^{(t)} \tag{20}$$

Besides, remarking that $x^{(t)}$ is a $d$-dimensional column vector, then:

$$y_k^T = \Psi_k^T(X) \approx \sum_{t=1}^{T}\sum_{u=1}^{d} \frac{\partial\Psi_k^T(\bar{X})}{\partial x_u^{(t)}} x_u^{(t)} \tag{21}$$

By replacing $x_k^{[\alpha]}$ of Equation (4) with $y_k^T$, it yields:

$$h_l = \sigma\left(\sum_{k=1}^{r} w_{l,k} \sum_{t=1}^{T}\sum_{u=1}^{d} \frac{\partial\Psi_k^T(\bar{X})}{\partial x_u^{(t)}} x_u^{(t)} + b_l\right) \tag{22}$$

which can be rewritten as follows:

$$h_l = \sigma\left(\sum_{k=1}^{r}\left[\sum_{u=1}^{d} \underbrace{\frac{\partial\Psi_k^T(\bar{X})}{\partial x_u^{(1)}} w_{l,k}}_{w_{l,u}^{(1)}} x_u^{(1)} + \cdots + \sum_{u=1}^{d} \underbrace{\frac{\partial\Psi_k^T(\bar{X})}{\partial x_u^{(T)}} w_{l,k}}_{w_{l,u}^{(T)}} x_u^{(T)}\right] + b_l\right) \tag{23}$$

As depicted, the derived features ($w_{l,u}^{(t)}$) express a complex form of content enclosed in data observed at different timestamps $t$. Also, their summation (on $k$) is able to group in a compact result all the behaviors of the input over $T$ timestamps. Ultimately, Equation (23) provides the aforementioned temporal dependency features.

## 3.5 | Stacked CNN-RNN-SAE

Once the two SAEs are being separately trained, each on the input provided of the previous stage, the input adapter (see Section 3.1) and the encoder functions of both CNN-SAE (Section 3.3) and RNN-SAE (see Section 3.4) are stacked to form a unique network.

With reference to Equations (1), (6), and (7), any component ($Y_{i,j}^{(t)}$) of the CNN output ($Y^{(t)}$) at time $t$ can be expressed as:

$$Y_u^{(t)} \equiv Y_{i,j}^{(t)} = \sum_{f=1}^{N_f}\sum_{p=1}^{P}\sum_{q=1}^{Q} K_{p,q}^{f} x_{\varphi(u)}^{(t)} \tag{24}$$

with $t \in \{1, \dots, T\}$, $u \in \{1, \dots, d'\}$, and $d' = N_x \times N_y$.

Consequently, according to Equation (8), the $l$th component of the output ($h^{[\gamma]\,(t)}$) of the encoder associated with CNN-SAE at the time $t \in \{1, \dots, T\}$ is:

$$h_l^{[\gamma](t)} = \sigma\left(\sum_{u=1}^{d'} w_{l,u}^{[\gamma]} Y_u^{(t)} + b_l^{[\gamma]\,(t)}\right), \quad l \in \{1, \dots, s\} \tag{25}$$

Next, according to Equation (16), any component $k$ of the LSTM-cell output after $T$ iterations can be expressed as:

$$y_k^T = \Psi_k^T(h^{[\gamma](1)}, \dots, h^{[\gamma](T)}), \quad k \in \{1, \dots, r\} \tag{26}$$

Lastly, the $l$th output component of the encoder ($h^{[\varrho]}$) of the RNN-SAE at the time $t \in \{1, \dots, T\}$ is:

$$h_l^{[\varrho]} = \sigma\left(\sum_{k=1}^{r} w_{l,k}^{[\varrho]} y_k^T + b_l^{[\varrho]}\right), \quad l \in \{1, \dots, r'\} \tag{27}$$

Also in this case we use the superscript notation $*^{[\varrho]}$ referring to all the specific parameters related to the LSTM network.

As depicted from these equations, the derived features are produced by a combination of the weights associated with the encoders of both the CNN and RNN-SAEs, that is ($w_{l,k}^{[\gamma]}$) and ($w_{l,k}^{[\varrho]}$), respectively.

As above, a first-order Taylor expansion can help to understand the meaning of Equation (27).

By substituting $Y_u^{(t)}$ of Equation (24) into Equation (25), for a sigmoid activation function, it yields:

$$h_l^{[\gamma]\,(t)} = \frac{1}{1 + \exp-\left(\sum_{u=1}^{d'} w'_{l,u} x_{\varphi(u)}^{(t)} + b_l^{[\gamma]\quad(t)}\right)} \tag{28}$$

with

$$w'_{l,u} = \sum_{f=1}^{N_f}\sum_{p=1}^{P}\sum_{q=1}^{Q} K_{p,q}^{f} w_{l,u}^{[\gamma]} \tag{29}$$

Thus, if we assume $\bar{X} = (0, ..., 0)'$, $b_l^{[\gamma]\,(t)} = 0 \forall \ t \in \{1, ..., T\}$, the Taylor series expansion of $h_l^{[\gamma]\,(t)}$ is given by:

$$\begin{aligned} h_l^{[\gamma]\,(t)} &\approx h_l^{[\gamma]\,(t)}(\bar{X}) + \nabla h_l^{[\gamma]\,(t)} X^{(t)} \\ &= \frac{1}{2} + \sum_{u=1}^{d'} \frac{\partial h_l^{[\gamma]\,(t)}(\bar{X})}{\partial x_{\varphi(u)}^{(t)}} x_{\varphi(u)}^{(t)} \\ &= \frac{1}{2} + \sum_{u=1}^{d'} w'_{l,u} x_{\varphi(u)}^{(t)} \end{aligned} \tag{30}$$

with $w'_{l,u}$ given by Equation (29), and $X^{(t)} = \left(x_{\varphi(1)}^{(t)}, ..., x_{\varphi(d')}^{(t)}\right)'$ being a column vector of the input at time $t$.

Let $H = (h^{[\gamma]\,(1)}, ..., h^{[\gamma]\,(T)})'$, for $X = \bar{X}$, then:

$$\bar{H} = H(\bar{X}) = \left\{\left\{\frac{1}{2}\right\}^s, ..., \left\{\frac{1}{2}\right\}^s\right\}^T \tag{31}$$

where $s$ is the number of hidden neurons, first defined in 3.2.

Therefore, the Taylor series expansion of $\Psi_k^T$ can be expressed as follows:

$$\begin{aligned} y_k^T = \quad & \Psi_k^T(H) \approx \Psi_k^T(\bar{H})(H - \bar{H}) \\ &= \Psi_k^T(\bar{H}) + \sum_{t=1}^{T}\sum_{u=1}^{s} \frac{\partial \Psi_k^T(\bar{H})}{\partial h_u^{[\gamma](t)}}\left(h_u^{[\gamma]\,(t)} - \frac{1}{2}\right) \end{aligned} \tag{32}$$

By replacing $h_u^{[\gamma]\,(t)}$ of Equation (32) with $h_l^{[\gamma]\,(t)}$ of Equation (30), it yields:

$$\Psi_k^T(H) \approx \Psi_k^T(\bar{H}) + \frac{1}{4}\sum_{t=1}^{T}\sum_{u=1}^{s}\sum_{\nu=1}^{d'} \frac{\partial \Psi_k^T(\bar{H})}{\partial h_u^{[\gamma]\quad(t)}} w'_{l,\nu} x_{\varphi(\nu)}^{(t)} \tag{33}$$

Finally, by substituting Equation (33) into Equation (26), and then in Equation (27), it yields:

$$h_l^{[\varrho]} = \sigma\left(\sum_{k=1}^{r} w_{l,k}^{[\varrho]}\left[\Psi_k^T(\bar{H}) + \frac{1}{4}\sum_{t=1}^{T}\sum_{u=1}^{s}\sum_{\nu=1}^{d'} \frac{\partial \Psi_k^T(\bar{H})}{\partial h_u^{[\gamma]\quad(t)}} w'_{l,\nu} x_{\varphi(\nu)}^{(t)}\right] + b_l^{[\varrho]}\right) \tag{34}$$

which can be rewritten as follows:

$$\begin{aligned} h_l^{[\varrho]} = \sigma\Bigg( & \sum_{k=1}^{r} w_{l,k}^{[\varrho]} \Psi_k^T(\bar{H}) \\ & + \frac{1}{4}\sum_{k=1}^{r}\left(\sum_{\nu=1}^{d'} \underbrace{\sum_{u=1}^{s} \frac{\partial \Psi_k^T(\bar{H})}{\partial h_u^{[\gamma]\,(1)}} w'_{l,\nu} w_{l,k}^{[\varrho]}}_{w''_{l,\nu}{}^{(1)}} x_{\varphi(\nu)}^{(1)} + , ... \right. \\ & \left. \cdot\cdot + \sum_{\nu=1}^{d'} \underbrace{\sum_{u=1}^{s} \frac{\partial \Psi_k^T(\bar{H})}{\partial h_u^{[\gamma](T)}} w'_{l,\nu} w_{l,k}^{[\varrho]}}_{w''_{l,\nu}{}^{(T)}} x_{\varphi(\nu)}^{(T)} \right) + b_l^{[\varrho]} \Bigg) \end{aligned} \tag{35}$$

Substituting $w'^{(t)}_{l,u}$ of Equation (29) into Equation (35), it is possible to observe that the temporal dependency features at timestamp $t$ are given by:

$$w''^{(t)}_{l,\nu} = \sum_{u=1}^{s}\sum_{f=1}^{N_f}\sum_{p=1}^{P}\sum_{q=1}^{Q} \frac{\partial \Psi_k^T(\bar{H})}{\partial h_u^{[\gamma](t)}} K_{p,q}^{f} w_{l,\nu}^{[\gamma]} w_{l,k}^{[\varrho]} \tag{36}$$

As depicted, the derived features ($w''^{(t)}_{l,\nu}$), through the summations on the indexes $f$, $p$, and $q$, are able to combine the features ($w_{l,\nu}^{[\gamma]}$ and $w_{l,k}^{[\varrho]}$) extracted from both SAEs, and provide a compact representation of the input over the time. We remark that $w_{l,\nu}^{[\gamma]}$ and $w_{l,k}^{[\varrho]}$ are derived by two separate training processes, that is the training of CNN-SAE and RNN-SAE, respectively.

## 4 | EXPERIMENTAL RESULTS

To evaluate the effectiveness of the proposed approach, in this section, the results obtained over a realistic testbed are shown and compared with the state-of-the-art.

### 4.1 | Power system testbed and attack scenarios

The data used in our experiment refer to a Power system testbed framework provided by Mississippi State University and Oak Ridge National Laboratory,[27] enclosing several real-world operating scenarios. As depicted in Figure 3, it includes two power generators (i.e., G1 and G2), four IEDs, that is, R1 through R4, each capable to control an assigned breaker (BR1 through BR4). IEDs are able to detect faults and consequently trips the breakers, but they are not equipped with intelligence allowing them to distinguish if the faults are actually valid or fake. The control panel includes all the tools need for monitoring and control the power system.

The provided framework generates 37 total scenarios, which include 8 Natural-events, 1 No-event, and 28 Attack-events. Each event comprises thousand of monitoring observations captured by four PMUs working at 120 samples/s, one control panel logging system, and a Snort intrusion detection system (IDS). Any PMU provides 29 status variables, thus the set of all PMUs provides 116 ($29 \times 4$) variables that become the features that directly describe the power system behavior, while the Snort IDS and the logs provide other 12 additional elements to be used for spotting anomalous events in the overall power control system. This, results into 128 basic elements describing the system activity, to be used for detecting and recognizing attacks, plus the class (or supervisory signal) consisting in a label reporting the involved attack type or the absence of anomalies (normal activity). For a more detailed description of these items, we remand to Reference.[27] These basic elements will be the starting point for mining the more meaningful space or time-related features that will be useful in detecting attacks.

All data are arranged into three schemes:

- *Binary*. Only two classes of operations are considered, that is Attack and Normal which include 28 and 9 events, respectively. Table 1 shows the numeric distribution of the items for each class.
- *Three-Class*. All the scenario are grouped into three classes, that is Attack (28 events), Natural (8 events), and “No events” (1 event). Table 2 shows the details of the numeric distribution.

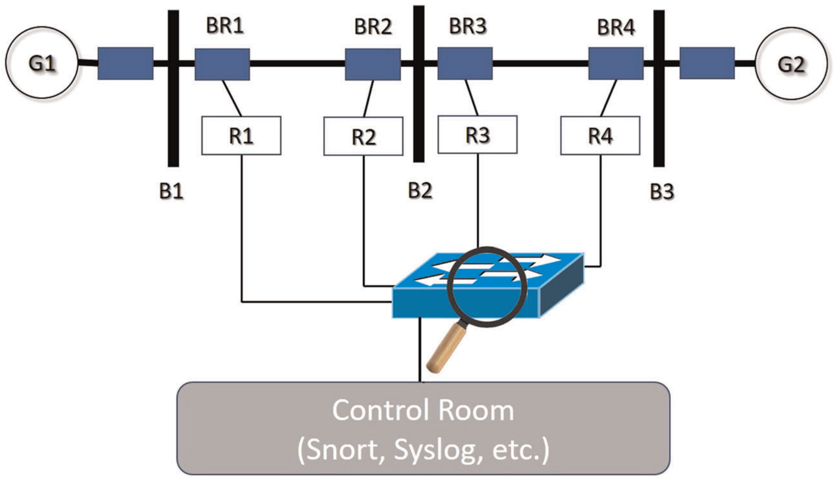

**FIGURE 3** Power system testbed framework provided by Mississippi State University and Oak Ridge National Laboratory[27] [Color figure can be viewed at wileyonlinelibrary.com]

**TABLE 1** Numeric distribution of the items for each class—Binary schema

| | Normal | Attack |
|---|---|---|
| **Data1** | 320 | 1150 |
| **Data2** | 453 | 1047 |
| **Data3** | 471 | 1133 |
| **Data4** | 530 | 1016 |
| **Data5** | 434 | 1094 |
| **Data6** | 433 | 1037 |
| **Data7** | 388 | 1163 |
| **Data8** | 453 | 1121 |
| **Data9** | 521 | 1061 |
| **Data10** | 484 | 1166 |
| **Data11** | 375 | 1181 |
| **Data12** | 521 | 1026 |
| **Data13** | 336 | 1225 |
| **Data14** | 396 | 1119 |
| **Data15** | 543 | 1015 |

- *Multi-Class*. Each of the 37 scenarios is considered as a different class. Due to its large size, the table of the numeric distribution is omitted.

Each schema is divided into 15 different data sets, each one comprising a different number of time-series items. Any item includes 129 columns reporting the 128 basic elements plus the

**TABLE 2** Numeric distribution of the items for each class—Three-Class schema

| | No-Event | Attack | Natural |
|---|---|---|---|
| **Data1** | 42 | 1150 | 268 |
| **Data2** | 87 | 1047 | 357 |
| **Data3** | 96 | 1133 | 365 |
| **Data4** | 111 | 1011 | 409 |
| **Data5** | 71 | 1094 | 353 |
| **Data6** | 247 | 1037 | 376 |
| **Data7** | 52 | 1163 | 325 |
| **Data8** | 97 | 1121 | 346 |
| **Data9** | 133 | 1061 | 378 |
| **Data10** | 88 | 1166 | 387 |
| **Data11** | 33 | 1181 | 331 |
| **Data12** | 105 | 1026 | 406 |
| **Data13** | 51 | 1225 | 275 |
| **Data14** | 14 | 111 | 372 |
| **Data15** | 144 | 1015 | 394 |

class. As shown below, we tested our approach for each of these schemes and compared it with some ML-based solutions characterizing the state-of-the-art.

## 4.2 | Experimental setting and network training

The proposed solution and all the experiments were implemented by using the Python language and the Keras-Tensorflow library running on a Desktop PC equipped with an Intel i7 CPU operating at 2.00 GHz frequency, 16 GB RAM, and an NVIDIA GeForce MX150 GPU with 4 GB of memory.

According to Equation (7), the 128 basic elements to be used for detection were arranged in a $(16 \times 8)$-dimensional matrix.

The CNN-SAE was implemented by using 2 Conv2D layers including 14 and 23 $(6 \times 3)$-dimensional filters, respectively. A *padding* = *same* and a *Relu* activation function were used. We remark that the inverse layer configuration, that is, 23 and 14, was used for the decoder. An Adam optimizer and a MSE loss function were used for training.

From the CNN-SAE, a new net (CNN-SAE-Encoder) was extracted comprising the input adapter and the encoder function of CNN-SAE. The resulting features, representing spatial relations among status variables, were fed into this network and its output (cf. Equation 8) was used to train the second SAE, that is, the RNN-SAE that will extract temporal dependencies.

The RNN-SAE was implemented by using two LSTM layers of 16 and 36 cells, respectively. A *timesteps* of 10 with *stride* of 1 was used, which corresponds to around 80 ms for PMU working at 120 samples/s. The reconstruction over time of the input was obtained by using a *TimeDistributed* layer. Also, in this case, an Adam optimizer and an MSE loss function were used for the training.

The encoder function of RNN-SAE was stacked with CNN-SAE-Encoder and its output (cf. Equations 27 and 35) was fed into a fully connected softmax neural network. The softmax network was used to classify the scenarios, and it includes 2 dense layers of 64 and 24 neurons, respectively. Also, to avoid overfitting, a dropout layer with a rate of 0.5 was added after each dense layer of the softmax network. As it is known, dropout is a regularization process that probabilistically removes some inputs during training by yielding the effect of making layers with a different number of nodes, thus avoiding overfitting. A *Relu* activation function was used, and the training was performed by using the Adam optimizer and the *categorical_crossentropy* loss function. Finally, the whole final stacked network was further trained for fine-tuning. Note that all the networks were trained on a very low number of epochs, that is 100. This means that the time spent on the training was around 1 min and 20 s.

## 4.3 | Evaluation metrics and results

To test the performance of the proposed stacked network architecture, all data sets associated to the aforementioned schemes were first divided by classes and then, the data of each class were divided into two parts, namely the *training* (70%) and *testing* (30%) sets. As usual, all data sets were rescaled through the min–max normalization to avoid biases. Besides, the exploratory data analysis (EDA) showed no outliers and missing values. Therefore, no further preprocessing was required. Well-known metrics derived from the multiclass confusion matrix[28] were used for the evaluation of the results obtained on testing sets, that is: Accuracy (Acc), Sensitivity (Sens), Specificity (Spec), Precision (Prec), Area under the ROC curve (AUC), and F_Measure (Fmea).

More precisely:

$$\text{Accuracy} \quad (\text{Acc}) = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{37}$$

$$\text{Sensitivity} \quad (\text{Sens}) = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{38}$$

$$\text{Specificity} \quad (\text{Spec}) = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{39}$$

$$\text{Precision} \quad (\text{Prec}) = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{40}$$

$$\text{AUC} = \frac{\text{Sens} + \text{Spec}}{2} \tag{41}$$

$$\text{F\_Measure} \quad (\text{Fmea}) = \frac{2{*}\text{Sens}{*}\text{Prec}}{\text{Sens} + \text{Prec}} \tag{42}$$

where TPs denote true positives and are the scenarios correctly classified, FPs denotes false positives and are the scenarios incorrectly classified, FNs denotes false negatives are the scenarios incorrectly rejected, and TNs denotes true negatives and are the scenarios correctly rejected.

For each metric, the results are reported as the average evaluated on the different classes. All the metrics were computed by using the *scikit-learn* Python library.[29] The results for any schema are provided in the following sections.

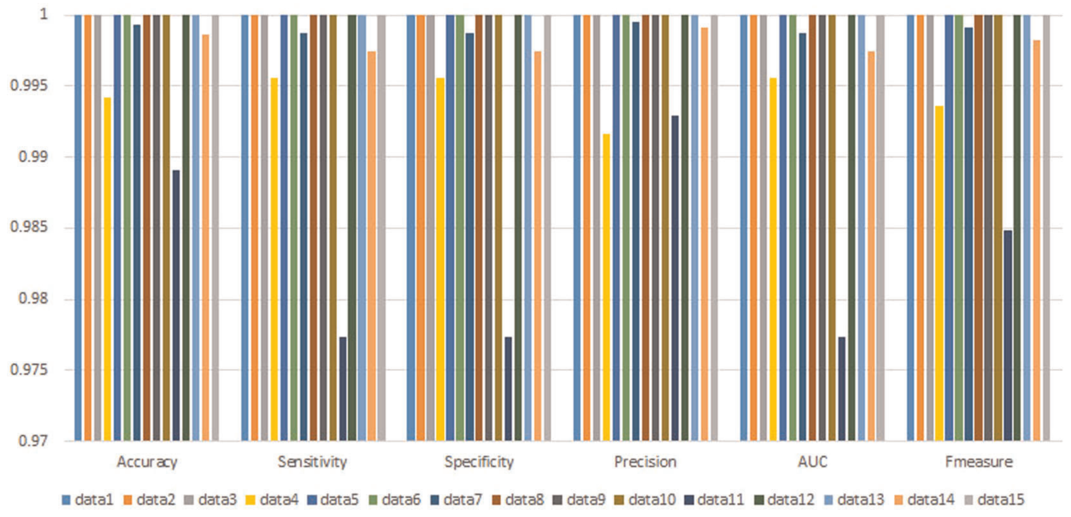

**FIGURE 4** Results for the Binary schema. The results are reported as average evaluated on the different classes. The worst case is represented by data11, which achieved 98.91%, 97.73%, 99.29%, 97.73%, and 98.48% for Accuracy, Sensitivity, Specificity, Precision, AUC, and F_measure, respectively [Color figure can be viewed at wileyonlinelibrary.com]

### 4.3.1 | Binary

Figure 4 depicts, for any data set (data1–data15), the performance metrics averaged on all the classes. As it is shown, for most of the data sets the metrics achieved 100%. Although the metrics associated with some data sets achieved lower values, it should be noted that they correspond to values just below 100%. Indeed, the worst case is represented by *data*11, which achieved 98.91%, 97.73%, 99.29%, 97.73%, and 98.48% for Accuracy, Sensitivity, Specificity, Precision, AUC, and F_measure, respectively.

### 4.3.2 | Three-class

Also in this case, for most of the data sets (9 out of 15, i.e., data 1, 5, 7, 8, 9, 10, 11, 13, and 15) the metrics achieved the maximum value of 100%, while for the remaining data sets the metrics achieved values just below the maximum. Indeed, as depicted in Figure 5, both data2 and data6 achieved the 99.91% of Accuracy and 99.87% for F_measure. Slightly lower values were achieved with data12 and data14 data sets, with an Accuracy of 99.74% and 99.46%, respectively. The worst cases are associated to data4 and data3 with an Accuracy of 99.17% and 99.08%, respectively.

### 4.3.3 | Multiclass

As shown in Figure 6, in the multiclass scenario, even though the metrics achieved high (near perfect) values overall, only data15 achieved the 100% of performance. Nevertheless, the metrics associated with the data sets from 3 to 14 achieved values very close to the ones of data15, that

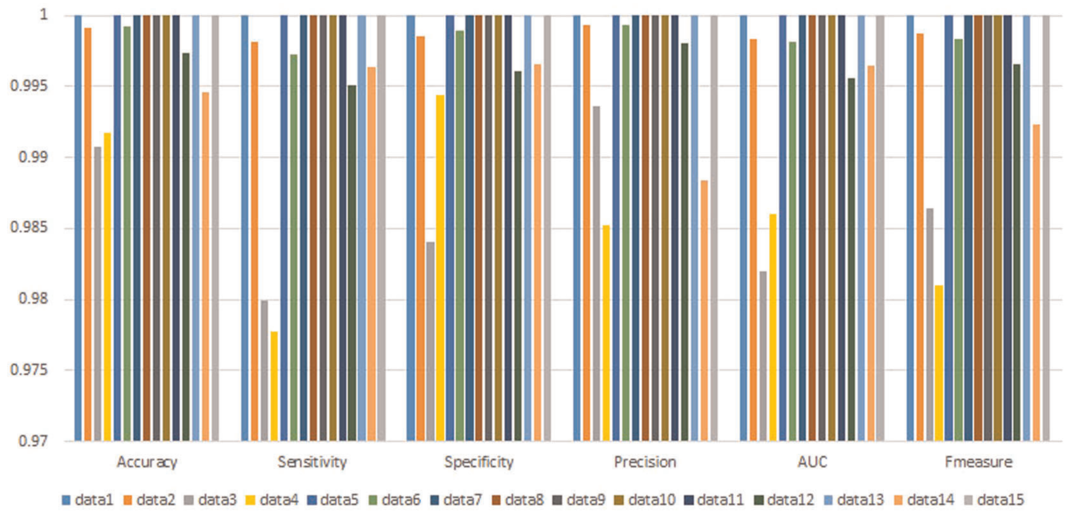

**FIGURE 5** Results for the three-class schema. The results are reported as average evaluated on the different classes. The worst cases are associated with data4 and data3 with an Accuracy of 99.17% and 99.08%, respectively [Color figure can be viewed at wileyonlinelibrary.com]

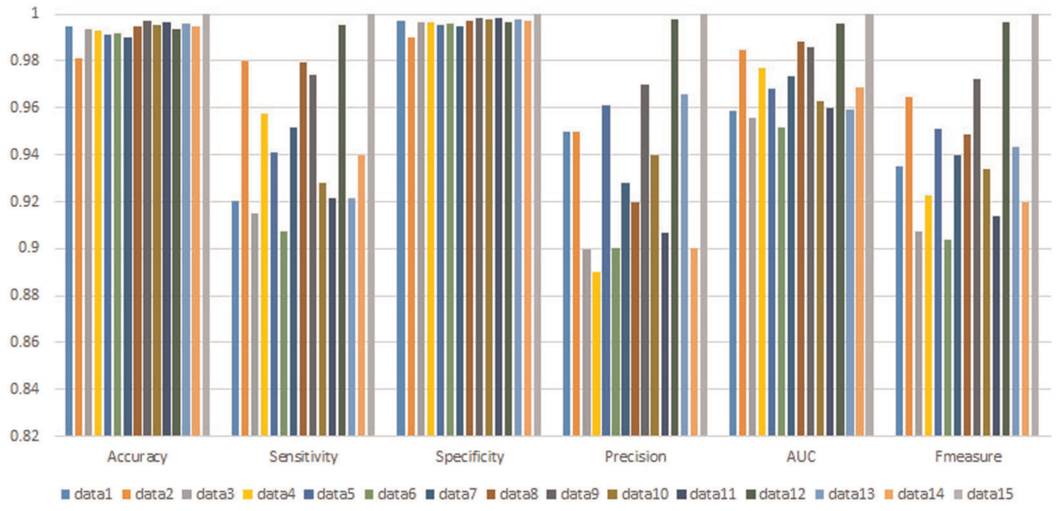

**FIGURE 6** Results for the multiclass schema. The results are reported as average evaluated on the different classes. The worst-case corresponds to data6, which achieved performance higher than 90% [Color figure can be viewed at wileyonlinelibrary.com]

is greater than 99%. Besides, unlike data15, the sensitivity associated with other data sets achieved lower values with respect to both Binary and three-class schemes. However, it should be noted that all the metrics assumed high values. Indeed, the worst case, corresponding to data6, achieved performance higher than 90%.

## 4.4 | Comparison and discussion

As shown in the previous sections, our approach achieved very high performance in all the scenarios. In addition, we want to remark that the training procedure was very fast (80 s on average, despite operating with limited computing resources) and for all the experiments the accuracy evaluated on the training sets achieved 100%. To give further proof of the effectiveness of the proposed architecture, Figures 7–9 show a comparison with the state-of-the-art based on ML. In particular, we compared our approach with seven different algorithms presented in Reference [27], that is, OneR,[30] NNge,[31] Random Forests,[32] Naive Bayes,[33] SVM,[34] JRipper,[35] and Adaboost.[36] The results refer to the average value of the Accuracy computed by using the

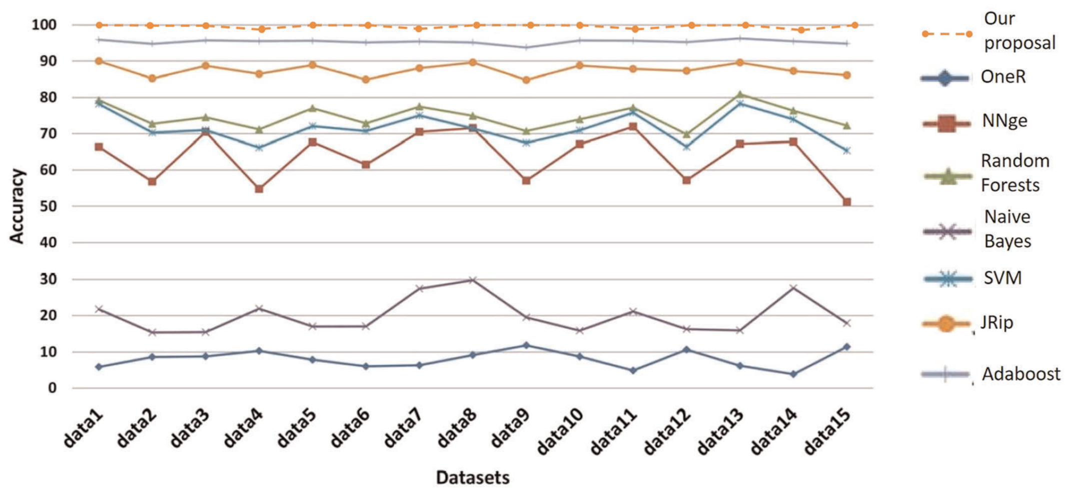

FIGURE 7 Comparison with the state-of-the-art—Binary-Class schema. Our approach outperformed the best state-of-the-art solution (i.e., Adaboost) by around 5% on average [Color figure can be viewed at wileyonlinelibrary.com]

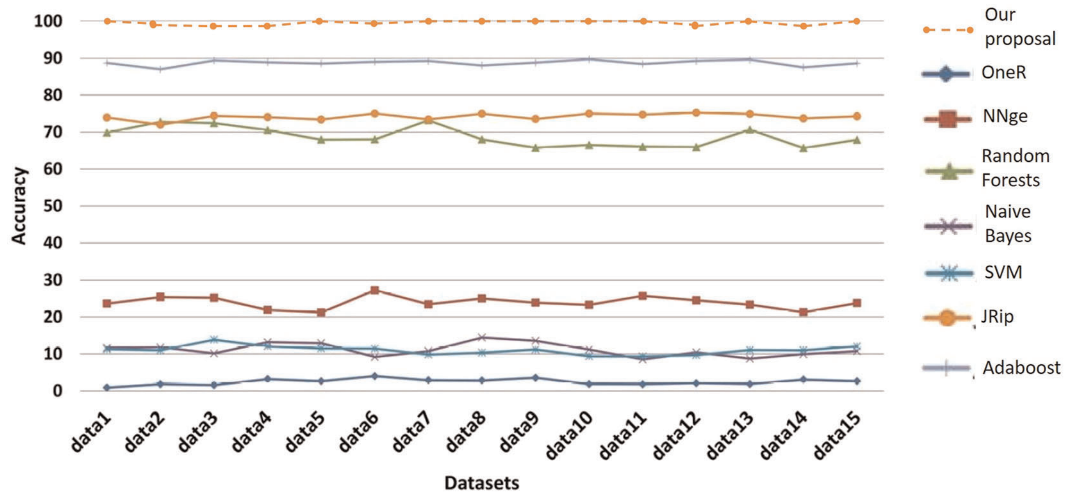

FIGURE 8 Comparison with the state-of-the-art—Three-Class schema. Our proposal outperformed the best state-of-the-art solution (i.e. Adaboost) by around 10% on average [Color figure can be viewed at wileyonlinelibrary.com]

10-fold cross-validation method. Accordingly, for any schema (i.e., binary, three, and multi-class), each data set was partitioned into 10 subsets including random instances coming from each category. Any subset was employed as a testing set and used for the performance evaluation, while the remaining sets were used as training set. All the considered approaches were tested on the same data set. No other particular setting was used in the comparison.[27]

As depicted, our approach outperformed the others in all the testbed evaluation schemes. In particular, for both the Three-Class and Multi-Class schemes, our proposal outperformed the best ML-based algorithms by around 10% on average. While, for Binary-Class, although the performance achieved by some ML-based algorithms are very high, our approach outperformed the best state-of-the-art solution (i.e., Adaboost) by around 5% on average. Finally, it should be noted that, differently from most of the available state-of-the-art solutions, the results of our approach remain steady with respect to the specific evaluation schemes and data sets used. Besides, to highlight how the proposed neural network configuration, and in particular the usage of the SAEs, influences the classification performance, we compared our findings with ones obtained by using only a CNN or an LSTM network followed by a fully connected soft-max network (referred to as CNN-NN and LSTM-NN, respectively). To ensure a fair comparison, all the configurations have been compared starting from the same operating conditions, that is same training/testing data set pair for each fold of the 10-fold cross-validation approach. Tables 3 and 4 reports the evaluation metrics averaged across all data sets (i.e., Data1–Data15) and classes. As it is depicted, for both the network configurations, the metrics achieved lower values than our proposal (see Table 5). The best performance was achieved

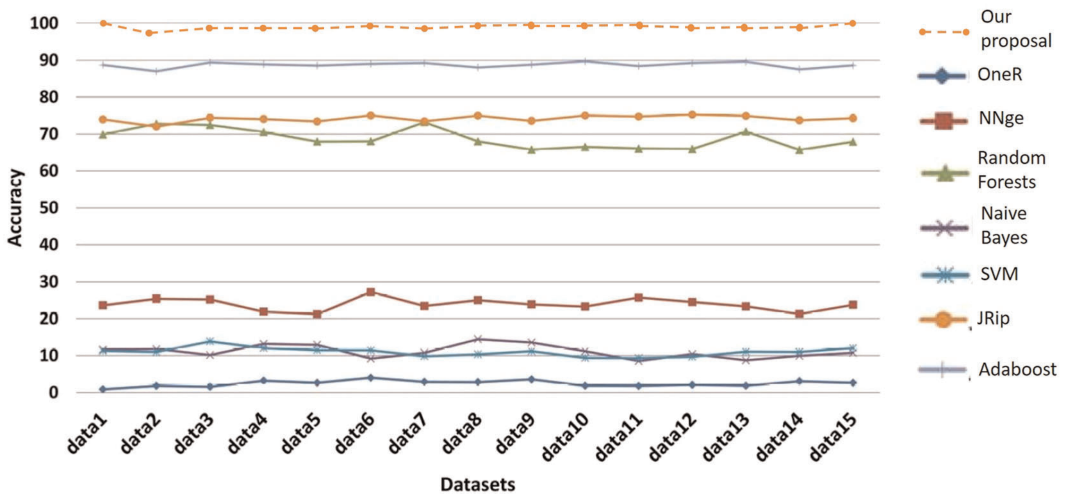

FIGURE 9 Comparison with the state-of-the-art—Multi-Class schema. Our proposal outperformed the best state-of-the-art solution (i.e., Adaboost) by around 10% on average [Color figure can be viewed at wileyonlinelibrary.com]

TABLE 3 Evaluation metrics averaged across all data sets and classes for the CNN-NN configuration

| | Acc | Sens | Spec | Prec | AUC | Fmea |
|---|---|---|---|---|---|---|
| Binary-Class | 0.92 | 0.89 | 0.89 | 0.92 | 0.89 | 0.90 |
| Three-Class | 0.93 | 0.76 | 0.91 | 0.88 | 0.84 | 0.79 |
| Multi-Class | 0.96 | 0.30 | 0.98 | 0.33 | 0.64 | NaN |

**TABLE 4** Evaluation metrics averaged across all data sets and classes for the LSTM-NN configuration

| | Acc | Sens | Spec | Prec | AUC | Fmea |
|---|---|---|---|---|---|---|
| Binary-Class | 0.84 | 0.77 | 0.77 | 0.83 | 0.77 | 0.79 |
| Three-Class | 0.88 | 0.88 | 0.88 | 0.88 | 0.88 | 0.88 |
| Multi-Class | 0.95 | 0.09 | 0.97 | NaN | 0.53 | NaN |

**TABLE 5** Evaluation metrics averaged across all data sets and classes for the proposed network configuration

| | Acc | Sens | Spec | Prec | AUC | Fmea |
|---|---|---|---|---|---|---|
| Binary-Class | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| Three-Class | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| Multi-Class | 0.99 | 0.95 | 1.00 | 0.94 | 0.97 | 0.94 |

for the CNN-NN configuration when applied to the Binary-Class scheme (cf. Table 3). Although both Sensitivity and Specificity achieved a good score of 89%, they were around 10% lower than ones obtained by our proposal, which achieved 100% performance.

## 5 | CONCLUSIONS AND FUTURE WORKS

In this paper, a stacked autoencoder-based convolutional and recurrent DNN is presented for detecting cyber-attacks in power control systems interconnected over the IoT. More specifically, the proposed architecture leverages two SAEs whose encoder-decoder functions are replaced with a convolutional (CNN-SAE) and a recurrent neural network (RNN-SAE), respectively. The encoders' functions are combined and fed into a softmax-based classifier for inferring the operation status of the control system.

Such detection architecture can be particularly useful for supporting early alerting and automatic reaction systems involved in situations in which the reactivity time and precision are crucial. Indeed, extensive performance evaluation experiments on realistic power system data resulted in almost perfect classification outcomes in both binary- and multi-class scenarios. The proposed architectural framework also exhibits good adaptivity to any kind of ICS applications without requiring preliminary field knowledge, since autoencoders with convolutional and recurrent encoding demonstrated to be effective in autonomously mining new more meaningful features from data. In particular, they have allowed the extraction of hard-to-detect correlations among data (spatial dependencies) and among data over time (temporal dependencies). Besides, because each autoencoder has been separately trained, it is evident that the architecture does not suffer from the vanishing gradient problem that affects large neural networks. Also, the compact representation (latent space) offered by autoencoders has allowed overcoming the training problem related to unbalanced data sets. Finally, the proposed architecture has proven to have a very low training latency. Indeed, the training procedure required only 80 s on average despite using very limited computing resources. Nevertheless, for all the experiments the accuracy evaluated on the training sets has always achieved 100%. The excellent results, obtained from the proposed approach without using preliminary knowledge of the applicant field, suggest us to continue the investigation in this direction. In this regard, in the future, we intend to evaluate the performance of the proposal on a number of scenarios that include any type of application field, and not necessarily related to power systems. As a consequence, we intend to generalize the proposed architecture and then implement a web-based service capable to offer an anomaly detection framework usable online through the Internet network.

## CONFLICT OF INTERESTS

The authors declare that there are no conflict of interests.

## AUTHOR CONTRIBUTIONS

**Gianni D'Angelo**: Idea, investigation, development of the theory, computation, software, experiments, and writing. **Francesco Palmieri**: Supervision, theory verification, methodology, writing, and editing.

## ORCID

*Gianni D'Angelo* https://orcid.org/0000-0001-7164-5736
*Francesco Palmieri* https://orcid.org/0000-0003-1760-5527

## REFERENCES

1. Ntalampiras S. Detection of integrity attacks in cyber-physical critical infrastructures using ensemble modeling. *IEEE Trans Industr Inform*. 2014;11(1):104-111.
2. Kolen JF, Kremer SC. Gradient flowin recurrentnets: the diffculty of learning long term dependencies. Wiley-IEEE Press; 2001:237-243.
3. Hadeli H, Schierholz R, Braendle M, Tuduce C. Leveraging determinism in industrial control systems for advanced anomaly detection and reliable security configuration. In: *2009 IEEE Conference on Emerging Technologies & Factory Automation*. IEEE; 2009:1-8.
4. Valenzuela J, Wang J, Bissinger N. Real-time intrusion detection in power system operations. *IEEE Trans Power Syst*. 2012;28(2):1052-1062.
5. Talebi M, Wang J, Qu Z. Secure power systems against malicious cyber-physical data attacks: protection and identification. In: *International Conference on Power Systems Engineering*; 2012:11-12.
6. Mohammadpourfard M, Sami A, Weng Y. Identification of false data injection attacks with considering the impact of wind generation and topology reconfigurations. *IEEE Trans Sustain Energy*. 2017;9(3):1349-1364.
7. Ekstrand MD, Riedl JT, Konstan JA. Anomaly detection for cybersecurity of the substations. *IEEE Trans Smart Grid*. 2011;2(4):865-873.
8. Chen Y, Luo B. S2a: secure smart household appliances. In: *Proceedings of the Second ACM Conference on Data and Application Security and Privacy*; 2012:217-228.
9. Pan S, Morris TH, Adhikari U A specification-based intrusion detection framework for cyber-physical environment in electric power system. *IJ Network Security*. 2015;17(2):174-188.
10. Pan S, Morris T, Adhikari U. Developing a hybrid intrusion detection system using data mining for power systems. *IEEE Trans Smart Grid*. 2015;6(6):3104-3113.
11. Pan S, Morris T, Adhikari U. Classification of disturbances and cyber-attacks in power systems using heterogeneous time-synchronized data. *IEEE Trans Industr Inform*. 2015;11(3):650-662.
12. ElChamie M, Lore KG, Shila DM, Surana A. Physics-based features for anomaly detection in power grids with micropmus. In: *2018 IEEE International Conference on Communications (ICC)*. IEEE; 2018:1-7.
13. Hink RCB, Beaver JM, Buckner MA, Morris T, Adhikari U, Pan S. Machine learning for power system disturbance and cyber-attack discrimination. In: *2014 7th International Symposium On Resilient Control Systems (ISRCS)*. IEEE; 2014:1-8.
14. Nader P, Honeine P, Beauseroy P. lp-norms in one-class classification for intrusion detection in SCADA systems. *IEEE Trans Industr Inform*. 2014;10(4):2308-2317.
15. Wang J, Shi D, Li Y, Chen J, Ding H, Duan X. Distributed framework for detecting PMU data manipulation attacks with deep autoencoders. *IEEE Trans Smart Grid*. 2018;10(4):4401-4410.
16. Esmalifalak M, Liu L, Nguyen N, Zheng R, Han Z. Detecting stealthy false data injection using machine learning in smart grid. *IEEE Syst J*. 2014;11(3):1644-1652.

17. Martinelli M, Tronci E, Dipoppa G, Balducelli C. Electric power system anomaly detection using neural networks. In: *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*. Springer; 2004:1242-1248.
18. Li S, Han Y, Yao X, Yingchen S, Wang J, Zhao Q. Electricity theft detection in power grids with deep learning and random forests. *J Electr Comput Eng*. 2019:2019.
19. Feng C, Li T, Chana D. Multi-level anomaly detection in industrial control systems via package signatures and LSTM networks. In: *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE; 2017:261-272.
20. Sharma N, Jain V, Mishra A. An analysis of convolutional neural networks for image classification. *Procedia Comput Sci*. 2018;132:377-384. https://www.sciencedirect.com/science/article/pii/S1877050918309335. International Conference on Computational Intelligence and Data Science.
21. Quax SC, D'Asaro M, van Gerven MAJ. Adaptive time scales in recurrent neural networks. *Sci Reports*. 2020;10(1):11360. https://doi.org/10.1038/s41598-020-68169-x
22. Hu Y, Lu X. Learning spatial-temporal features for video copy detection by the combination of CNN and RNN. *J Visual Commun Image Representation*. 2018;55:21-29. https://www.sciencedirect.com/science/article/pii/S1047320318301081
23. Chen L. *Curse of Dimensionality*. Boston, MA: Springer US; 2009:545-546. https://doi.org/10.1007/978-0-387-39940-9133
24. D'Angelo G, Palmieri F. Network traffic classification using deep convolutional recurrent autoencoder neural networks for spatial-temporal features extraction. *J Network Comput Appl*. 2021;173:102890. https://www.sciencedirect.com/science/article/pii/S1084804520303519
25. Makhzani A, Frey BJ. k-sparse autoencoders. In: *2nd International Conference on Learning Representations*, Conference Track Proceedings, ICLR 2014, Banff, AB, Canada, April 14–16, 2014; 2014:1-9. http://arxiv.org/abs/1312.5663
26. Yang S, Yu X, Zhou Y. LSTM and GRU neural network performance comparison study: taking yelp review dataset as an example. In: *2020 International Workshop on Electronic Communication and Artificial Intelligence* (IWECAI); 2020:98-101.
27. BorgesHink RC, Beaver JM, Buckner MA, Morris T, Adhikari U, Pan S. Machine learning for power system disturbance and cyber-attack discrimination. In: *2014 7th International Symposium on Resilient Control Systems (ISRCS)*; 2014:1-8.
28. Diez P. Chapter 1—Introduction. In: Diez P, ed. *Smart Wheel chairs and Brain-Computer Interfaces*. Academic Press; 2018:1-21. https://www.sciencedirect.com/science/article/pii/B9780128128923000017
29. Cournapeau D. scikit-learn; 2017. https://scikit-learn.org/stable/
30. Holte RC. Very simple classification rules perform well on most commonly used datasets. *Mach Learn*. 1993;11(1):63-90. https://doi.org/10.1023/A:1022631118932
31. Martin B. *Instance-Based Learning: Nearest Neighbor With Generalization*. University of Waikato; 1995.
32. Breiman L. Random forests. *Mach Learn*. 2001;45(1):5-32. https://doi.org/10.1023/A:1010933404324
33. Webb GI. *Naïve Bayes*. Boston, MA: Springer US; 2010:713-714. https://doi.org/10.1007/978-0-387-30164-8576
34. Yue S, Li P, Hao P. SVM classification: its contents and challenges. *Appl Math—A J Chinese Univ*. 2003;18(3):332-342.
35. Cohen WW. Fast effective rule induction. In: Prieditis A, Russell S, eds. *Machine Learning Proceedings*. San Francisco, CA: Morgan Kaufmann; 1995:115-123. https://www.sciencedirect.com/science/article/pii/B9781558603776500232
36. Wang R. AdaBoost for feature selection, classification and its relation with SVM, a review. *Phys Procedia*. https://www.sciencedirect.com/science/article/pii/S1875389212005767. International Conference on Solid State Devices and Materials Science, April 1–2, 2012, Macao.

**How to cite this article:** D'Angelo G, Palmieri F.. A stacked autoencoder-based convolutional and recurrent deep neural network for detecting cyberattacks in interconnected power control systems. *Int J Intell Syst*. 2021;1-23. https://doi.org/10.1002/int.22581