01 Jan 2022

A Novel Echo State Network Autoencoder for Anomaly Detection in Industrial Iot Systems

Fabrizio De Vita

Giorgio Nocera

Dario Bruneo

Sajal K. Das
Missouri University of Science and Technology, sdas@mst.edu

Recommended Citation

F. De Vita et al., "A Novel Echo State Network Autoencoder for Anomaly Detection in Industrial Iot Systems," *IEEE Transactions on Industrial Informatics*, Institute of Electrical and Electronics Engineers, Jan 2022.
The definitive version is available at https://doi.org/10.1109/TII.2022.3224981

A Novel Echo State Network Autoencoder for Anomaly Detection in Industrial IoT Systems

Fabrizio De Vita†, Giorgio Nocera†, Dario Bruneo†, Sajal K. Das‡

†*Department of Engineering, University of Messina, Italy*, {fdevita, dbruneo}@unime.it, giorgio.nocera@studenti.unime.it
‡*Department of Computer Science, Missouri University of Science and Technology, USA, sdas@mst.edu*

***Abstract*—The Industrial Internet of Things (IIoT) technology had a very strong impact on the realization of smart frameworks for detecting anomalous behaviors that could be potentially dangerous to a system. In this regard, most of the existing solutions involve the use of Artificial Intelligence (AI) models running on Edge devices, such as Intelligent Cyber Physical Systems (ICPS) typically equipped with sensing and actuating capabilities. However, the hardware restrictions of these devices make the implementation of an effective anomaly detection algorithm quite challenging. Considering an industrial scenario, where signals in the form of multivariate time-series should be analyzed to perform a diagnosis, Echo State Networks (ESNs) are a valid solution to bring the power of neural networks into low complexity models meeting the resource constraints. On the other hand, the use of such a technique has some limitations when applied in unsupervised contexts. In this paper, we propose a novel model that combines ESNs and autoencoders (ESN-AE) for the detection of anomalies in industrial systems. Unlike the ESN-AE models presented in the literature, our approach decouples the encoding and decoding steps and allows the optimization of both the processes while performing the dimensionality reduction. Experiments demonstrate that our solution outperforms other machine learning approaches and techniques we found in the literature resulting also in the best trade-off in terms of memory footprint and inference time.**

***Index Terms*—Industry 4.0, ESN, anomaly detection, edge computing, intelligent CPS**

I. Introduction

With the advent of Industry 4.0 paradigm, the realization of effective diagnosis systems is becoming extremely important to detect in advance abnormal conditions that can lead to unwanted behaviors, or even more severe consequences, such as complete breakdowns [1]. In the recent period, anomaly detection became a very popular approach to understand the health of a system and avoid potentially dangerous events that could affect its normal operation [2]. However, most of the challenges that emerge when trying to elaborate a diagnosis for an industrial plant find their roots in the complexity of these systems, where signals (typically in the form of multivariate time series) have to be properly analyzed. Moreover, if we consider that the events where a system exhibits an anomalous behavior can be rare, this makes even more challenging to perform an accurate diagnosis [3].

Until recently, Cloud computing has been a central component in the realization of anomaly detection frameworks by providing the infrastructure, the storage, and the high computing power necessary to run complex algorithms. On the other hand, if we consider an industrial scenario application, it poses requirements in terms of real-time/low latency response times, data privacy, and stable Internet connections, that make the use of this paradigm ineffective. To this end, modern solutions involve the Edge computing paradigm to move the computational resources close to where data is generated, thus improving the security, application latencies, and resources utilization [4].

Undoubtedly, the Industrial Internet of Things (IIoT) widespreading had a very strong impact on the Industry 4.0 scenario by pushing towards the realization of smart Edge frameworks capable of detecting anomalous conditions. Together with the Artificial Intelligence (AI), these technologies are two of the most important building blocks for the implementation of Intelligent Cyber Physical Systems (ICPS); in such a context, machine learning brings reasoning and learning capabilities through the exploitation of the sensing/actuating features of these devices. However, if on the one hand AI introduces new smart applications that can support the human operator, on the other, the hardware limitations of Edge devices make very complex the effective execution of onerous algorithms (e.g., neural networks) [5].

The majority of anomaly detection solutions today involves Deep Neural Network (DNN) architectures to accomplish this task, but they usually produce a large number of parameters which make them unsuitable to run on constrained devices. Such a condition, becomes even more evident when working with multivariate time series data, where resource demanding techniques such as Long Short Term Memories (LSTMs) are typically adopted to catch the temporal dynamics of a system. In order to mitigate this problem, Reservoir Computing (RC) is a framework where a state transition container (called reservoir) remains fixed during the training phase and has the task to capture complex input dynamics [6]. Echo State Networks (ESNs) [7] are families of neural networks belonging to RC which exploit the power of Recurrent Neural Networks (RNNs), while keeping the number of model trainable parameters low. As a consequence, these architectures exhibit a reduction of the memory footprint, model complexity and training/inference times, thus meeting the hardware requirements of a constrained Edge device. However, the use of such a technique has some limitations when applied in unsupervised settings.

Starting from the vanilla ESN Autoencoder (ESN-AE) ar-

chitecture [8], in this paper we propose a novel model which decouples the encoder and decoder networks and introduces a trainable layer to project input points in a new latent space. Using this network, we implemented an anomaly detection algorithm on a real scale replica industrial testbed and compared the performance of the proposed ESN-AE with some machine/deep learning techniques and other approaches we found in the literature. Experimental results demonstrate the effectiveness of the proposed technique and its feasibility for an industrial scenario.

The paper contributions can be summarized as follows. *i)* We propose a novel ESN-AE architecture which addresses some of the typical problems that emerge when working with the vanilla version of this model. To the best of our knowledge this is the first time a topology like this one is presented for an ESN-AE. *ii)* We develop an anomaly detection application to assess the working conditions of a real industrial testbed. *iii)* We present an extensive experimental results section where we compare several models in terms of predictive performance and memory occupation.

The rest of the paper is organized as follows. Section II reviews the related works. Section III provides a description about ESNs. Section IV presents the proposed ESN-AE architecture for the anomaly detection. Section V reports the anomaly detection algorithm for the assessment of an industrial plant working conditions. Section VI summarizes the experimental results obtained from testing and comparing the proposed technique with others found in the literature. Finally, Section VII concludes the paper providing also future research directions.

II. Related works

Modern industrial plants tend to be large systems where many components cooperate and interact for the execution of several tasks. In such a context, anomaly detection rapidly became a key element, supporting the human operator to locate the fault and activate a maintenance process [9]. Hardware based methods that exploit Physical Unclonable Functions (PUF) can be considered another viable solution for the detection of threats in industrial scenarios. This has been widely proved in [10] where authors investigate several AI approaches to improve the performance and privacy in e-Cash schemes. In the following, we summarize existing works found in the literature highlighting their differences from our approach.

When working in the area of time series anomaly detection, traditional RNNs exhibit the vanishing gradient problem that limits the learning sequences having long term dependencies. To overcome this drawback, new solutions involve the use of LSTM networks and its variants such as Gated Recurrent Units (GRUs) have been proposed. For example, in [11] is described an improved LSTM based anomaly detection scheme in rail transit devices. A LSTM and a Gaussian process model are used in [12] to detect outliers in IIoT by utilizing the predictive error. In [13] a Variational LSTM (VLSTM) model is adopted for intelligent anomaly detection in industrial systems. An enhanced Bi-directional LSTM version integrating Generative Adversarial Networks and Attention Mechanism (AMBi-GAN) is used in [14] to detect anomalies in industrial data by combining discrimination and reconstruction losses. Although the good results achieved in these works, LSTMs tend to be resource demanding and produce a large number of parameters which could make them unaffordable for an Edge device. In this paper we address these problems, proposing the use of an ESN-AE as a recurrent model that exploits its sparse structure to reduce the amount of parameters and network complexity.

The use of ESN-AEs as an anomaly detection method is not new in the literature. In [15] and [16] authors demonstrated the effectiveness of this technique for feature extraction and classification tasks. In [17] is presented a Multi Objective Particle Swarm Optimization Multi-Layered Echo State Network Autoencoder (MOPSO-MLESNAE) model as a method to perform anomaly detection. However, the architecture suffers from a strong unbalance between the encoding and decoding phase due to the impossibility to train the reservoir weights. In [18] authors propose a solution to mitigate this effect by introducing a training algorithm which iteratively replaces the encoding weights with the decoding ones. Although different, these solutions share the problem of not being able to perform dimensionality reduction; in this work, we were able to address it by introducing a trainable layer which enables the generation of an optimized latent representation while performing the dimensionality reduction. In [19] is described a model connecting the features extracted from several reservoirs that are passed to the output. The architecture contains encoder blocks that wrap two different types of dimensionality reduction algorithms such as: Principal Component Analysis (PCA) or Extreme Learning Machine Autoencoder (ELM-AE). A substantial difference with our approach is that our architecture is already an autoencoder (i.e., it does not require encoder blocks) which converts into a network with a lower number of layers. Such a choice derives also from our will to run the ESN-AE at the Edge where device hardware resources can be quite limited. Moreover, the model in [19] is not presented as an anomaly detection technique, but as a solution to visualize multiscale dynamics in time series.

The works described in [20] and [21] combine ESNs with a Sparse Autoencoder (SAE) and a Convolutional Autoencoder (CAE), respectively, for temporal feature extraction and anomaly detection; in both cases, we can observe two disjoint systems whose building blocks do not interact during the learning phase, a problem that could affect the quality of the prediction. In this sense, the proposed ESN-AE architecture performs the feature extraction and detection tasks using a single model and thanks to the use of a code layer immediately after the encoding reservoir it is able to optimize the prediction, the encoding, and decoding processes at the same time during the training procedure, with a consequent improvement of the performance while maintaining a reduced number of parameters.

III. Echo State Networks

For a better reading and understanding we report in Table I the main notation used in the paper.

TABLE I: Notation used in the paper.

Paper symbols	
K	ESN input units
L	ESN output units
M	Code layer units
N	Reservoir units
T	Input sequence length
α	Reservoir connectivity
W_{in}	ESN input weights
W_{res}	ESN reservoir weights
W_{back}	ESN feedback weights
W_{out}	ESN output weights
$W_{\mathcal{E}}$	Encoder weights
$W_{\mathcal{D}}$	Decoder weights
$W_{res}^{\mathcal{E}}$	Encoding reservoir weights
$W_{res}^{\mathcal{D}}$	Decoding reservoir weights
$W_{C_{in}}$	Code layer input weights
$W_{C_{out}}$	Code layer output weights
$b_{\mathcal{E}}$	Encoder bias
$b_{\mathcal{D}}$	Decoder bias
u	Input vector
$\hat{u}$	Reconstructed input vector
x	State vector
$\hat{x}$	Reconstructed state vector
z	Encoded latent vector
y	Output vector
$\mathcal{L}$	Autoencoder reconstruction error

Originally introduced in [7], ESNs belong to the RC framework and are typically adopted to process and analyze time series data. Architecturally speaking, an ESN is a RNN characterized by a sparse, randomly connected recurrent structure (the reservoir) consisting of non trainable weights, and a trainable part called *readout* representing the output. Fig. 1 depicts an ESN architecture where the dashed and the solid arrows are respectively the trainable and not-trainable weights, while the gray lines indicate valid, but not required links.

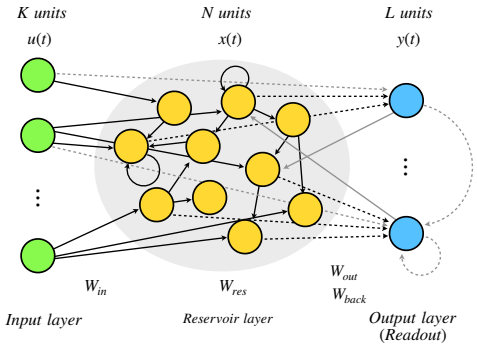


Fig. 1: ESN architecture.

Given an ESN with K input units, N reservoir units, and L output units, we define its weights matrices as: $W_{in} \in \mathbb{R}^{N\times K}$, $W_{res} \in \mathbb{R}^{N\times N}$, $W_{out} \in \mathbb{R}^{L\times(K+N+L)}$, and $W_{back} \in \mathbb{R}^{N\times L}$ containing in the given order, the weights between the inputs and the reservoir, the weights inside the reservoir, the weights from the input and reservoir layers to the outputs, and the feedback weights between the readout and the reservoir. Remembering that ESNs derive from RNNs, we introduce the equations describing the network behavior as follows [6], [7]:

$$x(t+1) = f(W_{in}\cdot u(t+1) + W_{res}\cdot x(t) + W_{back}\cdot y(t)), \quad (1)$$

$$y(t+1) = g(W_{out}\cdot [u(t+1), x(t+1), y(t)]), \quad (2)$$

where $x(t+1)$ is the new computed reservoir state acting as a "memory" component, storing the temporal evolution (just like in RNNs) and depending on the new input $u(t+1)$, the state $x(t)$ and the output $y(t)$, while $f(\cdot)$ is the activation function.

On the other hand, the new output equation is derived from the concatenation of the new state $x(t+1)$, the input $u(t+1)$ and the previous output $y(t)$ which makes it depending on the "history" of the network itself. Also in this case the entire expression is wrapped around an activation function $g(\cdot)$ that produces the actual output of the network.

Unlike traditional RNNs, in ESNs the weights of W_{in} and W_{res} are sparse, randomly sampled, and remain fixed (also during the training process). Such a feature allows to decrease the model complexity because of the lower number of trainable weights. Moreover, since these weights are contained only in the W_{out} matrix, it is possible to speed-up the training phase which can be achieved (in some cases) solving a linear regression problem [7]. On the other hand, the heavy stochastic nature of ESNs requires a proper hyperparameters setting in order to obtain a stability in the network outputs and not making them only subject to the "luck" of the random initialization. A typical solution is to use reservoirs with a large number of internal units to "encourage" the formation of independent sparsely connected subnetworks capable of catching a wide range of input dynamics [7].

In this sense, the ESN core idea consists in using the reservoir as a temporal feature extractor. Thanks to its large dimensionality, such a layer is able to extract high level features (or meta features) that allow to capture the temporal structure of the input data [22]. However, in order to properly work, the reservoir must satisfy the *separation* and *echo state* properties [7], [23] (hence the name of these networks). With respect to the first one, it guarantees that two separate inputs will produce separate states; this is fundamental to avoid the "collapse" of the reservoir that could cause the extraction of almost the same features for any provided input. To achieve this condition, it is sufficient to adopt sparse and large reservoirs in order to stimulate the generation of different connections with the inputs. The echo state property tells that the effects of the previous input $u(t)$ and state $x(t)$ on a future state $x(t+1)$ should gradually vanish as the time passes [24]. In most cases, empirical tests demonstrated that a W_{res} weight matrix whose spectral radius (i.e., largest eigenvalue in absolute value) is lower than one is a necessary (but not sufficient) requisite to satisfy this property. Although weights initialization methods (e.g., random normal or Glorot uniform), if properly scaled, allow the generation of matrices respecting this condition,

we should point out that for some inputs it could be neither necessary nor sufficient, which makes the echo state property still an object of research [25].

IV. Proposed ESN Autoencoder architecture

In the previous section, we provided a description about ESNs and how they work. In this section, we introduce a novel model architecture for ESN-AEs useful in the analysis of multivariate time series data. For a better understanding, let us define what an autoencoder is. From a topological point of view, an autoencoder is a neural network which has the task of learning an efficient representation of the input in an unsupervised way (i.e., without the corresponding labels). The peculiar aspect that differentiates these type of networks from the others is a specific hidden layer named *code layer* containing a new latent representation of the data passed as input (as shown in Fig. 2).

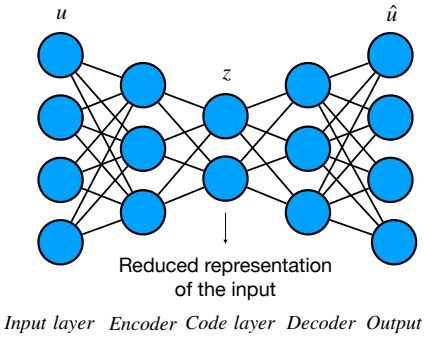


Fig. 2: Autoencoder architecture.

An autoencoder can be described as the combination of two building blocks, namely: the *encoder* and *decoder* networks. They are defined as: $\mathcal{E} : U \rightarrow Z$ and $\mathcal{D} : Z \rightarrow U$ where the encoder $\mathcal{E}$ is a function mapping a k dimensional (with k the number of features) input $U \in \mathbb{R}^k$ in a new m dimensional latent space (i.e., the code size) $Z \in \mathbb{R}^m$ where typically $m < k$; in particular cases where $m > k$, these architectures are called *overcomplete* autoencoders. On the other hand, the decoder $\mathcal{D}$ is a function that reconstructs the input from the latent space bringing it back to the original space. Obviously, such a process implies a loss of information due to the compression of the dimensions. Given the following system of equations:

$$\begin{cases} z = f_{\mathcal{E}}(W_{\mathcal{E}} u + b_{\mathcal{E}}) \\ \hat{u} = f_{\mathcal{D}}(W_{\mathcal{D}} z + b_{\mathcal{D}}) \\ \mathcal{L} = \|u - \hat{u}\|^2 , \end{cases} \tag{3}$$

the first equation describes the code computation z by the encoder network where $f_{\mathcal{E}}(\cdot)$ is the encoder activation function, and $W_{\mathcal{E}}$ and $b_{\mathcal{E}}$ are the encoder weights matrix and bias vector respectively. With respect to the second equation, $\hat{u}$ is the input reconstructed by the decoder network where, just like the previous case, $f_{\mathcal{D}}(\cdot)$ is the decoder activation function (equal to the one used to encode), whereas $W_{\mathcal{D}}$ and $b_{\mathcal{D}}$ are the decoder weight matrix and bias vector. Finally, the last equation represents the neural network loss function $\mathcal{L}$ (or reconstruction error) computed as the norm of the original input u and the reconstructed one $\hat{u}$. The autoencoder is trained to find the best set of parameters which minimize the loss function (and consequently, the reconstruction error); in particular, the satisfaction of this condition ensures also the minimization of information loss.

A vanilla ESN-AE is similar to an autoencoder [8]. It can be considered as an ESN where the numbers of input and output units coincide (i.e., $K = L$) and the input at the generic t^{th} time instant is equal to the output at the same instant (i.e., $u(t) = y(t)$) as shown in Fig. 3. In this specific case, we can observe that the vanilla ESN-AE does not adopt some of the connections which are typical of ESNs such as: the ones between the input layer and the readout, those between the output units, and the backwards connections. That said, we can rewrite the vanilla ESN-AE equations as follows:

$$x(t+1) = f(W_{in} \cdot u(t+1) + W_{res} \cdot x(t)), \tag{4}$$

$$y(t+1) = g(W_{out} \cdot x(t+1)). \tag{5}$$

Given this setting, the network is trained to reconstruct the sequences passed as input with the great advantage of keeping fixed the majority of the weights inside the network. However, when using such a topology, the ESN-AE loses some of the peculiar features of "traditional" autoencoders.

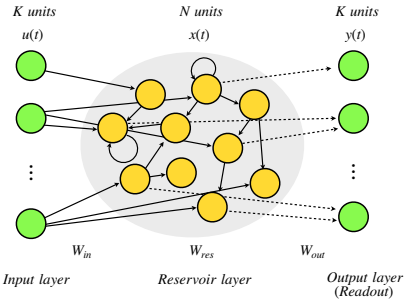


Fig. 3: Vanilla ESN-AE architecture.

By observing the vanilla ESN-AE architecture the encoding and decoding processes are somehow "mimicked" by the W_{in}, W_{res}, and W_{out} weights. However, due to the nature of this technique, such that the only trainable weights are those related to the output (i.e., W_{out}), we obtain a network where only the decoding phase is really optimized. Another problem is represented by the absence of a code layer (one of the core elements in autoencoders) to achieve dimensionality reduction, clustering, and anomaly detection tasks. We can think that the latent code lives inside the reservoir, but unfortunately such a layer is not suitable to play as a bottleneck element inside a network, because "rich" reservoirs are required to fully catch the dynamics of a system [7].

Hence, to address these issues, we propose a novel ESN-AE architecture (shown in Fig. 4) with the purpose of making

This article has been accepted for publication in IEEE Transactions on Industrial Informatics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TII.2022.3224981

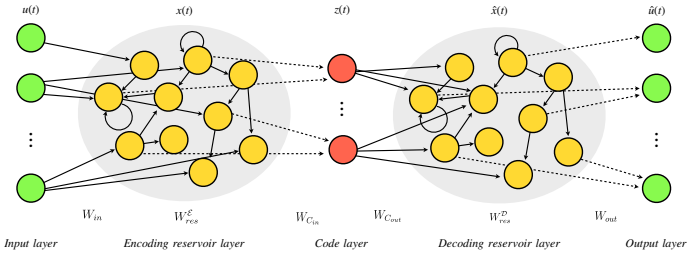


Fig. 4: Proposed ESN-AE architecture.

it more similar to the original model definition of traditional autoencoders. The first change we made towards this direction consists in the use of two separated reservoirs (i.e., one for the encoding and one for the decoding). Such an addition has the main advantage of decoupling these two steps, thus making the $W_{res}^{\mathcal{E}}$ weights responsible for the learning of the input dynamics and its encoding, and the $W_{res}^{\mathcal{D}}$ weights for the decoding and input reconstruction. Compared to vanilla ESN-AE, where the single reservoir is involved in both the processes, in this network model we were able to add a new degree of freedom which results in a better learning process.

The second change we made is the introduction of a trainable code layer acting as a connecting bridge between the encoding and the decoding reservoirs. Such a layer plays a key role in our model since it enables the optimization of the encoding process. In particular, this has been made possible by means of the trainable $W_{C_{in}}$ weights which force the encoder to generate an optimized encoded representation. More specifically, the code is a fully connected layer whose units fix the latent space dimension m with a consequent effect on the number of encoded features. Given this new configuration, let us define the equations describing the proposed ESN-AE as follows:

$$\begin{cases} x(t+1) = f(W_{in} \cdot u(t+1) + W_{res}^{\mathcal{E}} \cdot x(t) + b_{\mathcal{E}}) \\ z(t+1) = f(W_{C_{in}} \cdot x(t+1)) \\ \hat{x}(t+1) = f(W_{C_{out}} \cdot z(t+1) + W_{res}^{\mathcal{D}} \cdot \hat{x}(t) + b_{\mathcal{D}}) \\ \hat{u}(t+1) = f(W_{out} \cdot \hat{x}(t+1)) \end{cases} \tag{6}$$

where the obtained system is a combination of eqs. (3), (4), and (5). The first equation is equivalent to the one shown for the vanilla ESN-AE with the new computed state still depending on the input $u(t+1)$ and on the state $x(t)$ which is multiplied by the encoding weights $W_{res}^{\mathcal{E}}$. The second equation describes the output (i.e., the code layer) of the first reservoir; mathematically speaking, it is equivalent to the ESN-AE output equation (see eq. (5)), with the substantial difference that the encoded features $z(t+1)$ are a function of the computed model state and the input code weights $W_{C_{in}}$. The rest of the equations define the decoding network whose task is to reconstruct the model state $\hat{x}(t+1)$ from the latent representation $z(t+1)$. Unlike vanilla ESN-AEs, we can observe that in this case the reconstruction task is split into two phases: the first one is achieved by means of the decoding reservoir weights $W_{res}^{\mathcal{D}}$ for the state reconstruction. Finally, the second phase is responsible for the actual input reconstruction $\hat{u}(t+1)$ by combining the reconstructed state $\hat{x}(t+1)$ computed at previous step and the trained weights W_{out}. Hence, we were able to obtain a novel architecture where the training process is balanced between the encoding and decoding parts. Compared to the vanilla ESN-AE, our approach requires a larger number of weights necessary for the encoding optimization. However, if we consider the benefits derived from their addition, the proposed architecture still represents a good trade-off in terms of trainable parameters and performance.

A. Computational complexity analysis

This sub-suction analyzes the computational complexity of the proposed ESN-AE. Assuming input sequences of length T, K input neurons and N reservoir neurons, it has been demonstrated that the time complexity for the reservoir connected to the input layer is computed as follows [19]:

$$\mathcal{C}_{res}^{\mathcal{E}} = O(2\alpha TN^2 + TNK), \tag{7}$$

where α is the reservoir connectivity which usually is very small (in the order of 0.1 or even smaller). The output of the encoding reservoir is then passed to a fully connected layer (i.e., the code) with M units (with $M < K$) for which the time complexity depends on $W_{res}^{\mathcal{E}}$ and $W_{C_{in}}$ matrices, whose sizes are $N \times N$ and $N \times M$, respectively, and can be computed:

$$\begin{aligned} \mathcal{C}_{code} &= O(N \cdot N \cdot M) = \\ &= O(N^2 \cdot M). \end{aligned} \tag{8}$$

The last part of the model is another reservoir (i.e., the decoding one) which in this case receives the input from the code layer having M units. Hence, we can compute the time computational complexity for the decoder part as already did in eq. (7):

$$\mathcal{C}_{res}^{\mathcal{D}} = O(2\alpha TN^2 + TNM). \tag{9}$$

Combining these three components the overall time complexity of the proposed ESN-AE is the following:

$$\begin{aligned} \mathcal{C}_{ESN-AE} = \mathcal{C}_{res}^{\mathcal{E}} + \mathcal{C}_{code} + \mathcal{C}_{res}^{\mathcal{D}} = \\ = O(2\alpha TN^2 + TNK) + O(N^2 \cdot M) + O(2\alpha TN^2 + TNM). \end{aligned} \tag{10}$$

Due to the very small connectivity values, the time complexity of the encoding and decoding reservoirs can be reduced to $O(TNK)$ and $O(TNM)$, respectively. Moreover, if we consider that $M < K$ and in general $TK < NM$, we can rewrite the time complexity equation as:

$$\begin{aligned} \mathcal{C}_{ESN-AE} &= O(TNK) + O(N^2 \cdot M) = \\ &= O(N^2 \cdot M), \end{aligned} \tag{11}$$

which results to be quadratic with respect to N.

This article has been accepted for publication in IEEE Transactions on Industrial Informatics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TII.2022.3224981

V. Anomaly detection

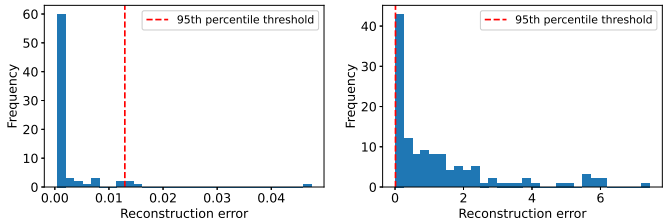
Over the years, understanding systems "health" state and the consequent fault prevention have been two of the biggest challenges in the industry business. Until recently, most of the proposed approaches tackled this problem by fixing a maintenance schedule to prevent the occurrence of faults. Evidently, when working with complex systems, the use of such an approach becomes unfeasible in terms of time, money, and resources. Moreover, the large number of signals and intricate non-linear relationships describing an industrial process make very difficult the elaboration of a precise diagnosis using traditional methods. In such a context, anomaly detection can be considered a valid solution for the recognition of events that can be potentially dangerous for an industrial plant. Indeed, a timely detection becomes critical to avoid the emergence of conditions that can be harmful for a system.

Although anomaly detection is gaining a lot of interest in the recent period, the lack of anomalous data (especially in the industrial environment) is another challenge that makes very hard the validation and implementation of this kind of algorithms. The literature demonstrated that the use of unsupervised approaches can address this problem through the definition of intelligent models that can learn from unlabeled data. Specifically, if we consider an industrial scenario where systems typically work in a "normal" operative setting, the use of unsupervised techniques is a viable way to design algorithms that should extract and learn patterns describing a working plant and then exploit this knowledge to identify anomalous behaviors and promptly start fault-response mechanisms.

The ESN-AE proposed in this work embodies the above mentioned characteristics necessary to properly operate in an industrial environment. Thanks to its unsupervised nature, our algorithm does not require labeled data for the training process; in fact, as described in Section IV, an autoencoder is a model that is asked to learn how to reconstruct the input samples by using as output the input itself. On the other hand, the use of ESNs as encoder and decoder allows to reduce the network weights, thus resulting into a model whose training and inference process can be sustainable even for a constrained Edge device.

The methodology we adopted to detect anomalies using the proposed ESN-AE model is the following. We first trained the ESN-AE in an unsupervised way passing as input the sequences associated to the normal working condition. By doing this, the model is able to learn those patterns and features that characterize this condition with the aim of obtaining a low reconstruction error. On the contrary, for an anomaly that by definition is an event that deviates from the trend determined by the data to which it belongs, the reconstruction error can be very high. Hence, we can exploit this metric as a discriminant element between a normal and an anomalous behavior.

Fig. 5 depicts a histogram of the reconstruction errors for the normal and anomalous sequences, where the Mean Squared Error (MSE) has been used as loss function to compute the



(a) Normal reconstructions (b) Anomalous reconstructions

Fig. 5: Reconstruction errors for normal and anomalous sequences.

"distance" between the original input sequences and the corresponding reconstructions. As we would expect, the majority of the errors relative to the normal condition are very low and clustered around zero (see Fig. 5a) with a range of values sensibly lower than the ones shown in Fig. 5b where the reconstruction errors exhibit a significant increase. In general, the choice of a threshold value represents a core step that heavily affects the performance of an anomaly detection algorithm; this process becomes even more complex when working in an industrial scenario, where the selection should be carefully done in order to avoid the generation of too many false positives and false negatives. In such a context, we introduce the anomaly threshold τ as the $95th$ percentile over the training reconstruction errors such that the input sequence $u(t)$ is anomalous if $\mathcal{L}(u(t), \hat{u}(t)) \geq \tau$. In this sense, to select this threshold, we conducted a series of experiments from which the $95th$ percentile resulted the best in terms of performance. Using this definition, we obtained a threshold which changes every time the model is updated with a new set of training data. Such a property is fundamental to achieve a dynamic threshold that can evolve and adapt to the occurrence of new normal and anomalous patterns.

VI. Experimental results

In this section, we present the results derived from the experimentation we conducted to test the proposed ESN-AE. We demonstrate the feasibility of our approach showing its performance in detecting the anomalies and the advantages of using it compared to other machine and deep learning techniques. Specifically, we considered a real assembly plant (shown in Fig. 6) used in the automotive sector for the transportation of car pieces. The system is equipped with two gear motors, six belts and a cart that is able to move back and forth by means of a limit switch that, when activated, reverses the motion direction.

In such a context, we were interested in the detection of anomalous vibrations along the testbed frame structure that usually denote the presence of a mechanical fault. To this aim we instrumented the plant with a VTV-122 sensor produced by IFM electronics to measure the vibrations and detect anomalies due to the brake system or cart limit switch malfunctioning. In this sense, a very important feature of our testbed is the possibility to manually inject mechanical faults. For example,

This article has been accepted for publication in IEEE Transactions on Industrial Informatics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TII.2022.3224981

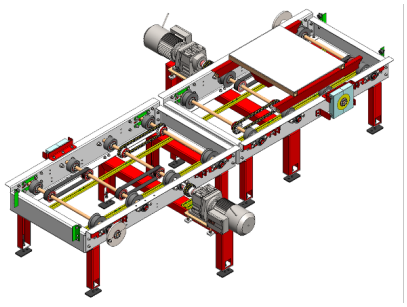
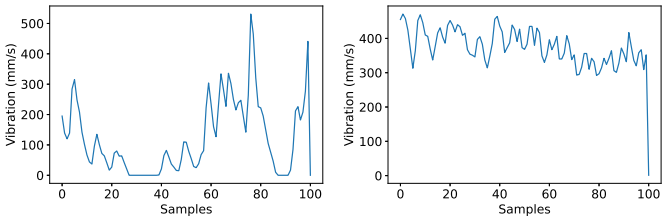


Fig. 6: CAD representation of the scale replica industrial plant.

when acting on the plant brake system, it is possible to increment the motors gears friction causing a cart movement subject to mechanical strains, which converts into the generation of large vibrations. On the other hand, the fault injection in the limit switch causes an immediate stop of the cart movement once it reaches one of the plant extremities, with a consequent reduction of the overall structural vibrations. In general, being able to inject faults in the tested has been fundamental for our studies allowing the collection of a labeled dataset, the validation of our approach and a better understanding of the plant behavior in a normal and anomalous operative setting.

Using the above mentioned testbed we were able to collect samples in correspondence of two conditions, namely: a normal setting where the cart is able to move back and forth, and an anomalous one where a mechanical fault is injected, negatively affecting the cart motion. Figure 7 depicts two examples of vibration signals that have been obtained slicing the raw data into smaller sequences by means of a sliding window whose width of 100 samples has been set experimentally.



(a) Normal (b) Anomalous

Fig. 7: Vibration raw signals collected from the industrial plant in correspondence of a normal and an anomalous condition.

In Table II, we report a possible view of the dataset structure collected using the above described scale replica plant, where each row represents a vibration sequence. The dataset is provided with binary labels such that each record is associated with a normal event (labelled as 0), or an anomaly (labelled as 1). With respect to the bottom part of the table, we computed a set of statistical indicators (i.e., mean, standard deviation, quartiles, and max value) for each column to provide a better understanding of the data. We decided to tackle this problem using an unsupervised approach; as already mentioned in Section V, the use of such a technique can be beneficial when working in industrial scenarios. Moreover, the use of an unsupervised model makes the proposed algorithm more generic and suitable even for those applications where the data labelling is not available or possible. To this aim, the normal data has been divided into train, validation and test sets following a 80%, 10%, and 10% split ratio. We then added the remaining anomalous data to the test set, so that the training and validation processes are performed only on normal data to learn its patterns and tune the model hyperparameters, whereas the test set contains both normal and anomalous data to evaluate the performance of the proposed approach. In this sense, due to the unsupervised nature of the proposed approach, the labels have been used exclusively during the testing phase in order to compute the $precision$, $recall$, $F1$-$score$, accuracy, and Matthews Correlation Coefficient (MCC) metrics. As a final pre-processing step, we normalized the data using a min-max scaling approach so that the features were in the same range of values.

TABLE II: Dataset structure and statistical information.

t	$vibr(t+1)$	$vibr(t+2)$	...	$vibr(t+100)$	$Label$
0	222	333	...	274	0
...	...	...	...	...	...
10	5	34	...	79	0
11	195	398	...	598	1
...	...	...	...	...	...
50	1100	304	...	489	1
...	...	...	...	...	...
$Mean\ value$	507.07	494.09	...	514.83	–
$Std.$	522.56	507.53	...	514.83	–
$1^{st}\ quartile$	139	128.75	...	122.5	–
$2^{nd}\ quartile$	294	279.50	...	307.5	–
$3^{rd}\ quartile$	778.35	732	...	752.25	–
$Max.\ value$	2473	2489	...	2497	–

Given this setting, we started with the implementation of our ESN-AE by selecting the best hyperparameters according to the performance achieved on the validation set. With respect to the number of units, we set the encoding and decoding reservoirs to 100 and the code layer to 50. Such a configuration has been obtained by means of *Keras tuner*, an optimization framework that helps finding the best hyperparameters for a given neural network model. The connectivity parameter α, which refers to the percentage of connections inside the reservoirs was set to 0.1 (thus implying a sparsity of 90%) to ensure the separation property discussed in Section III whereas the spectral radius was set to 0.9 to achieve the echo state property. The model implemented in [26] involves the backpropagation algorithm for the training process, hence we used *Adam* as optimizer with a *learning rate* of 0.001. Finally, we set the training epochs to 2500, but to avoid the model overfitting, we used an early stopping approach to terminate the learning procedure if the model is not able to reduce the validation loss for a consecutive number of epochs. Such a value is defined by the *patience* term that in our specific implementation was set to 10 epochs.

To demonstrate the advantages of the proposed ESN-AE, we compared it against several approaches. In particular, we

considered four different models, namely: a One Class Support Vector Machine (OCSVM), a LSTM autoencoder (LSTM-AE), a GRU autoencoder (GRU-AE), and the vanilla version of ESN-AE. The results are summarized in Table III, moreover, it is worth to mention that for each of the aforementioned techniques, we performed 100 different experiments in order to provide a more precise evaluation of their performance by computing the mean and 95% confidence interval for each index.

For the OCSVM model, we set the *radial basis* kernel, a *gamma* equal to $\frac{1}{N_f \cdot \sigma^2(U)}$ where N_f is the number of features of the dataset and $\sigma^2(U)$ is the variance of the input passed for the training. Finally, a tolerance value of 0.001 has been set as stopping criterion for the solver. The OCSVM returned good results with a precision of 0.93, a recall of 0.92, a $F1\text{-}score$ of 0.93, and a MCC of 0.34.

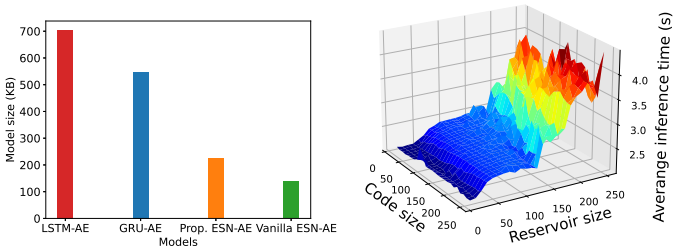
The LSTM-AE topology has been set via Keras tuner, which returned the same results to the ones obtained for our ESN-AE with 100 neurons for the LSTM encoder and decoder cells. The model generated roughly 171 K parameters, which make it very resource demanding in terms of memory and inference time, on the other hand, it reached a precision of 0.95, a recall of 0.96, a $F1\text{-}score$ of 0.96 and a MCC of 0.66, thus outperforming the OCSVM algorithm.

For the GRU-AE we used the same topology adopted in the LSTM-AE resulting in a reduced amount of parameters (i.e., 131 K). In general, the use of GRU cells as encoder and decoders resulted in model with a level of performance in between the OCVSM and the LSTM-AE with a precision of 0.94, a recall of 0.90, a $F1\text{-}score$ of 0.93, and a MCC score of 0.42.

Regarding the vanilla ESN-AE, we considered a topology like the one shown in Fig. 3 with a reservoir of 100 neurons. Despite the simplicity of the topology, the model performed well and obtained a precision of 0.95, a recall of 0.90 a $F1\text{-}score$ of 0.92, and a MCC of 0.42; such results prove the effectiveness of RC models and their ability to extract complex time series dynamics using a much lower amount of trainable parameters than deep learning models like LSTM-AE.

The proposed model architecture uses a larger amount of parameters if compared with the vanilla ESN-AE; on the other hand, the addition of a trainable code layer had a positive impact on the overall performance which reached a very good precision (i.e., 0.95) and the highest values of recall, $F1\text{-}score$, and MCC: 0.97, 0.96, and 0.61, respectively. For the sake of completeness, we report in Table IV the confusion matrix extracted from the proposed ESN-AE in the best case scenario on set of 100 experiments.

With reference to Fig. 8a, it shows a bar plot of models memory footprint. As we would expect, the LSTM-AE which resulted to be the model with the largest number of parameters has also the largest size of 700 KB, followed by the GRU-AE with a memory footprint of 547 KB. With respect to the proposed ESN-AE, we were able to obtain a much smaller model with a size of 224 KB by exploiting the RC framework that allows to effectively reduce the network size thanks to its weights sparsity. The vanilla ESN-AE model reached the lowest memory size with 138 KB. In this sense, the additional encoding weights in our model generated a little increase of the memory occupation which is still suitable to fit the hardware of an Edge device.



(a) Memory footprint (b) Time complexity

Fig. 8: Memory footprint and computational complexity of the proposed ESN-AE.

Fig. 8b depicts the inference time computational complexity of the proposed ESN-AE when varying the reservoir neurons (i.e., N) and code neurons (i.e., M) sizes. As we would expect from the analysis conducted in Section IV-A, the time complexity exhibits a quadratic trend as the reservoir neurons increase. In this sense, also the code has an impact on the model performance due to the fully connected nature of this layer. However, if we look at the worst case scenario, the model has an average inference time of $4.43s$ which is still a good response time.

To further prove the effectiveness of our approach, we compared it against some of state of the art approaches we found in the literature, where, for a fair comparison, we considered the same ECG200 dataset adopted in these works [27]. Such a dataset is a very popular anomaly detection benchmark used for the recognition of anomalous heartbeats, where each of the 200 signal samples is a sequence of 96 timesteps. As in the previous case, the dataset has been randomly split into train, validation, and test sets (using only normal data during the training and validation phases), and employing the labels in the test set only to extract the performance metrics. Specifically we considered four different approaches: two versions of MOPSO-MLESNAE proposed in [17], the Global Reversible Autoencoder Echo State Network (GRAE-ESN) presented in [18], and a model combining a Temporal Feature Network (TFN) and LSTM-based Attention Network (LSTMaN) [28].

Table V reports the results derived from this comparison. Following the same methodology discussed above, we found the best hyperparameters setup for our ESN-AE using Keras tuner, which returned a topology with 150 reservoir neurons for both the encoder and decoder, and 50 neurons for the code layer. Also in this case, our solution reached good results with a $F1\text{-}score$ close to 1 (i.e., 0.953) and an overall accuracy that outperforms the MOPSO-MLESNAE. In general, the best performance are achieved by the GRAE-ESN and LSTMaN,

TABLE III: Models performance comparison on our industrial dataset.

Model	*Precision*	*Recall*	*F1-score*	*MCC*	*Params*
OCSVM	0.93 ± 0.003	0.92 ± 0.001	0.93 ± 0.001	0.34 ± 0.023	–
LSTM-AE	0.95 ± 0.003	0.96 ± 0.007	0.96 ± 0.003	0.60 ± 0.023	$171\ K$
GRU-AE	0.94 ± 0.004	0.90 ± 0.008	0.93 ± 0.003	0.42 ± 0.020	$131\ K$
Vanilla ESN-AE	0.95 ± 0.004	0.90 ± 0.006	0.92 ± 0.002	0.43 ± 0.020	$30.2\ K$
Prop. ESN-AE	0.95 ± 0.004	0.97 ± 0.004	0.96 ± 0.001	0.61 ± 0.020	$50.3\ K$

TABLE IV: Confusion matrix of the proposed ESN-AE.

		Actual	
		Anomaly (1)	Normal (0)
Pred.	Anomaly (1)	1964	112
	Normal (0)	24	225

TABLE V: Models performance comparison on ECG200 dataset.

Model	*F1-score*	*Accuracy*	*Params*
MLESNAE (bi-objective) [17]	-	0.905	-
MLESNAE (tri-objective) [17]	-	0.900	-
GRAE-ESN [18]	-	0.920	$1.1M$
TFN + LSTMaN [28]	-	0.920	-
Prop. ESN-AE	0.953	0.912	$89.2K$

however it is worth to mention that the techniques adopted in [18] and [28] make use of a supervised approach. In this sense, one of the main advantages of the proposed ESN-AE is the possibility to train it in an unsupervised way without the need of external labels. Moreover, the GRAE-ESN requires a massive amount of parameters which makes it unsuitable to be deployed on a constrained Edge device; given these results our ESN-AE can be considered the best trade-off in terms of memory footprint and predictive performance.

VII. Conclusions

In this paper, we proposed a novel architecture which extends the vanilla ESN-AE by decoupling the encoding and decoding steps using two separate reservoirs, and introduces a trainable code layer for the generation of an optimized latent representation of the input. Our algorithm has been able to outperform other approaches and techniques we found in the literature resulting in the best trade-off in term of memory occupation and predictive performance. Such features make our model a viable solution to be executed on hardware constrained devices (e.g., microcontrollers) and suitable for an industrial application scenario.

Future works will be devoted to the improvement of the anomaly detection algorithm performance, to the study of new solutions that allow a further reduction of trainable weights and memory footprint (e.g., model quantization), and to the deployment on a Edge device to enable the possibility of an on-device training and inference.

References

[1] E. Kozłowski, D. Mazurkiewicz, T. Żabiński, S. Prucnal, and J. Sep, "Machining sensor data management for operation-level predictive model," *Expert Systems with Applications*, vol. 159, p. 113600, 2020.

[2] F. De Vita, D. Bruneo, and S. K. Das, "A novel data collection framework for telemetry and anomaly detection in industrial iot systems," in *2020 IEEE/ACM Fifth International Conference on Internet-of-Things Design and Implementation (IoTDI)*, 2020, pp. 245–251.

[3] G. Pang, C. Shen, L. Cao, and A. V. D. Hengel, "Deep learning for anomaly detection: A review," *ACM Comput. Surv.*, vol. 54, no. 2, mar 2021.

[4] L. Cui, Y. Qu, G. Xie, D. Zeng, R. Li, S. Shen, and S. Yu, "Security and privacy-enhanced federated learning for anomaly detection in iot infrastructures," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 5, pp. 3492–3500, 2022.

[5] F. De Vita, G. Nocera, D. Bruneo, V. Tomaselli, D. Giacalone, and S. K. Das, "Quantitative analysis of deep leaf: a plant disease detector on the smart edge," in *2020 IEEE International Conference on Smart Computing (SMARTCOMP)*, 2020, pp. 49–56.

[6] A. Rodan and P. Tino, "Minimum complexity echo state network," *IEEE Transactions on Neural Networks*, vol. 22, no. 1, pp. 131–144, 2011.

[7] H. Jaeger, "The "echo state" approach to analysing and training recurrent neural networks-with an erratum note," *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, vol. 148, no. 34, p. 13, 2001.

[8] S. Zhang, X. Lin, L. Wu, Y. Song, N. Liao, and Z. Liang, "Network traffic anomaly detection based on ml-esn for power metering system," *Mathematical Problems in Engineering*, vol. 2020, 2020.

[9] Y. Zhang, Z. Y. Dong, W. Kong, and K. Meng, "A composite anomaly detection system for data-driven power plant condition monitoring," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 7, pp. 4390–4402, 2020.

[10] G. Fragkos, C. Minwalla, J. Plusquellic, and E. E. Tsiropoulou, "Artificially intelligent electronic money," *IEEE Consumer Electronics Magazine*, vol. 10, no. 4, pp. 81–89, 2021.

[11] Y. Wang, X. Du, Z. Lu, Q. Duan, and J. Wu, "Improved lstm-based time-series anomaly detection in rail transit operation environments," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2022.

[12] D. Wu, Z. Jiang, X. Xie, X. Wei, W. Yu, and R. Li, "Lstm learning with bayesian and gaussian processing for anomaly detection in industrial iot," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 8, pp. 5244–5253, 2020.

[13] X. Zhou, Y. Hu, W. Liang, J. Ma, and Q. Jin, "Variational lstm enhanced anomaly detection for industrial big data," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 5, pp. 3469–3477, 2021.

[14] F. Kong, J. Li, B. Jiang, H. Wang, and H. Song, "Integrated generative model for industrial anomaly detection via bi-directional lstm and attention mechanism," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2021.

[15] L. Sun, B. Jin, H. Yang, J. Tong, C. Liu, and H. Xiong, "Unsupervised eeg feature extraction based on echo state network," *Information Sciences*, vol. 475, pp. 1–17, 2019.

[16] J. Long, S. Zhang, and C. Li, "Evolving deep echo state networks for intelligent fault diagnosis," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 7, pp. 4928–4937, 2020.

[17] N. Chouikhi, B. Ammar, A. Hussain, and A. M. Alimi, "Bi-level multi-objective evolution of a multi-layered echo-state network autoencoder for data representations," *Neurocomputing*, vol. 341, pp. 195–211, 2019.

[18] H. Wang, Q. J. Wu, D. Wang, J. Xin, Y. Yang, and K. Yu, "Echo state network with a global reversible autoencoder for time series classification," *Information Sciences*, vol. 570, pp. 744–768, 2021.

[19] Q. Ma, L. Shen, and G. W. Cottrell, "Deepr-esn: A deep projection-encoding echo-state network," *Information Sciences*, vol. 511, pp. 152–171, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025519309053

[20] J. Long, Z. Sun, C. Li, Y. Hong, Y. Bai, and S. Zhang, "A novel sparse echo autoencoder network for data-driven fault diagnosis of delta 3-

d printers," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 3, pp. 683–692, 2020.

[21] R. Liu, B. Reimer, S. Song, B. Mehler, and E. Solovey, "Unsupervised fNIRS feature extraction with CAE and ESN autoencoder for driver cognitive load classification," *Journal of Neural Engineering*, vol. 18, no. 3, p. 036002, mar 2021.

[22] C. Gallicchio, A. Micheli, and L. Pedrelli, "Deep reservoir computing: A critical experimental analysis," *Neurocomputing*, vol. 268, pp. 87–99, 2017, advances in artificial neural networks, machine learning and computational intelligence. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231217307567

[23] W. Maass, T. Natschläger, and H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," *Neural Computation*, vol. 14, no. 11, pp. 2531–2560, 2002.

[24] M. Lukoševičius and H. Jaeger, "Reservoir computing approaches to recurrent neural network training," *Computer Science Review*, vol. 3, no. 3, pp. 127–149, 2009.

[25] C. Gallicchio, "Chasing the echo state property," in *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, April 2019.

[26] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. [Online]. Available: https://www.tensorflow.org/

[27] R. T. Olszewski, *Generalized feature extraction for structural pattern recognition in time-series data*. Carnegie Mellon University, 2001.

[28] Z. Xiao, X. Xu, H. Xing, S. Luo, P. Dai, and D. Zhan, "Rtfn: A robust temporal feature network for time series classification," *Information Sciences*, vol. 571, pp. 65–86, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025521003820

Fabrizio De Vita is a Research Fellow at the University of Messina, Italy. The research activity of Fabrizio De Vita focuses on Intelligent Cyber Physical Systems and implementation of machine/deep learning algorithms on Smart Environments. His research topics also include the Internet of Things applied in smart city contexts, where he has investigated the problem of improving the applications Quality of Service (QoS) by proposing solutions based on Federated Learning and Deep Reinforcement learning approaches. Regarding the Smart Industry area, his research activity aims to create smart systems to diagnose the health state of an industrial plant via the fault prediction, anomaly detection and predictive maintenance techniques.

Giorgio Nocera received the bachelor degree (summa cum laude) in Electrical and Electronical Engineering from the Engineering Department of the University of Messina (Italy) in 2019. In 2021, he received the master degree (summa cum laude) in Engineering and Computer Science from the Engineering Department of the University of Messina (Italy). His research activity is focused on the Internet of Things, embedded systems, and on the study of machine learning techniques (Deep Neural Networks, Recurrent Neural Networks, and Deep/Reinforcement Learning) implementation on Intelligent Cyber Physical Systems with applications on Smart Agriculture and Smart Industry contexts.

Dario Bruneo is an Associate Professor of Computer Engineering at the University of Messina, Italy. The research activity of Dario Bruneo has been focused on the study of distributed systems with particular regards to the management of advanced service provisioning, to the system modeling and performance evaluation. His current research topics include Internet of Things (and its application in Smart City scenarios), performance and reliability of complex systems, Machine Learning techniques for Cyber Physical Systems. He has published over 100 papers in international journal and conferences and he is co-editor of the book "Quantitative Assessments of Distributed Systems - Methodologies and Techniques" - Scrivener/Wiley. He has been involved in several European projects and he is co-founder of the startup SmartMe.io, a University spin-off that develops innovative IoT solutions for smart environments.

Sajal K. Das (Fellow, IEEE) is currently a professor of computer science and Daniel St. Clair Endowed Chair at Missouri University of Science and Technology, Rolla, USA. Prior to 2013, he was a University Distinguished Scholar Professor at the University of Texas at Arlington. His research interests include cyber-physical systems and IoTs, smart environments, cyber security, wireless and sensor networks, mobile and pervasive computing, cloud and fog computing, and social networks. He published extensively in these areas and received ten best paper awards. His h-index is 86 with more than 32,000 citations. He serves as the founding Editor-in-Chief of Elsevier's Pervasive and Mobile Computing Journal, and as Associate Editor of several journals including the IEEE Transactions of Mobile Computing, IEEE Transactions on Dependable and Secure Computing, and ACM Transactions on Sensor Networks.