Efficient Anomaly Detection on Temporal Data via Echo State Networks and Dynamic Thresholding

Antonio Carta, Giacomo Carfì, Valerio De Caro and Claudio Gallicchio

Abstract

Embedded devices are frequently used to deploy adaptive learning systems for several applications, such as anomaly detection models in automotive or aerospace domains. These models can detect anomalous data from the sensors to predict hazardous situations ahead of time. However, training on-the-edge requires the use of efficient learning algorithms, able to run on low-powered devices while keeping a high accuracy. In this paper, we propose the use of Echo State Networks (ESN), a randomized family of efficiently trainable recurrent networks, for anomaly detection on-the-edge in aerospace applications. The anomaly detection method uses a nonparametric dynamic threshold to detect anomalous behaviours from the observed data by comparing it to the model's predictions. The proposed model is empirically assessed on aerospace data against state-of-the-art LSTM networks. The results show that the proposed method grants a *6x speedup in training time*, while also improving the outlier detection performance.

Keywords

anomaly detection, echo state networks, semi-supervised learning, reservoir computing

1. Introduction

On-the-edge applications of machine learning (ML) systems are becoming ever more frequent, and especially crucial to the success of human-centric cyber-physical applications [1, 2, 3]. Deploying ML models on edge devices requires solving a number of challenges, among which stands out the ability of performing training and inference on low powered devices with a limited memory and computational budget. Anomaly detection is a popular application for embedded devices, such as in the automotive or aerospace domains. The objective is to train a model that can detect anomalous data from the sensors and predict hazardous situations ahead of time.

One usual approach for performing anomaly detection consists in training a model to predict the data under a non-anomalous behaviour, and measure the deviation of the observed values with respect to the prediction. This approach typically leverages on the expressiveness of the Recurrent Neural Networks (RNNs) [4] to model the non-anomalous sequences. In RNNs, the parameters are typically learned by gradient descent, with the gradient computed by backpropagation through time. Unfortunately, a well-known issue of this learning approach is the vanishing gradient problem [5], which results in difficulty in converging towards good

Proceedings of the 1st International Workshop on Computational Intelligence for Process Mining (CI4PM) and the 1st International Workshop on Pervasive Artificial Intelligence (PAI), co-located with the IEEE World Congress on Computational Intelligence (WCCI), Padua, Italy, 18–23 July 2022

antonio.carta@unipi.it (A. Carta); g.carfi1@studenti.unipi.it (G. Carfì); valerio.decaro@phd.unipi.it (V. De Caro); claudio.gallicchio@unipi.it (C. Gallicchio)

solutions. Several alternatives have been proposed to mitigate this issue [6, 7, 8], and in particular, gated architectures such as LSTMs [6] and GRUs [9] are among the most popular solutions. Due to the effectiveness of LSTMs to model multidimensional time series without the need for fixed windows size, they were used in several anomaly detection applications [10, 11, 12].

However, these models are typically characterized by a high computational cost for the training phase, and their expressiveness may be lost in the attempt of reaching a trade-off between the model's complexity and the computational cost. These two issues make them unsuitable for their use on embedded devices and on-the-edge applications. For solving this kind of limitations, an orthogonal approach is provided by Echo State Networks (ESNs) [13], a family of recurrent networks where the recurrent parameters are initialized randomly and are not updated during training. The only parameters that are learned during training process are those of the readout, the model which computes the transformation from the recurrent hidden state to the output. ESNs are widely applied in several applications [14, 15, 16], and they are especially competitive in embedded applications thanks to their low computational requirements.

In this paper, we address the problem of performing *efficient anomaly detection on temporal data* while training on-the-edge. In particular, we focus on the task of detecting *contextual anomalies* on multivariate telemetry data: single values within the time-series which do not fall in a low density region of the data distribution, while exhibiting an anomalous behaviour with respect to the context of observation (i.e., their surrounding subsequence) [17]. The work in [18] addressed such problem by proposing a semi-supervised framework which leverages Long Short-Term Memory networks for performing value predictions of the telemetry data (*supervised*), along with a Nonparametric Dynamic Thresholding technique for detecting anomalous behaviours from the observed data with respect to the predicted values (*unsupervised*). Our proposed solution consists in a more efficient version of such framework, which allows to overcome the computational limitations of LSTMs by replacing them with Echo State Networks (ESNs), and perform both training and anomaly detection directly on-the-edge. Experiments show that, at parity of memory resources exploited by the models, ESNs can achieve a better trade-off between expressiveness and computational cost with respect to LSTMs, confirming their validity for their use on on-the-edge applications.

The rest of this paper is organized as follows. Section 2 describes the proposed anomaly detection method. Section 3 describes the experimental setup. Section 4 shows the results of the experiments. Finally, in Section 5 we draw the conclusion of the work.

2. Efficient Anomaly Detection via Nonparametric Dynamic Thresholding

In this section, we provide the details of the anomaly detection method used in this work, which is composed of two parts: an Echo State Network for predicting the data under a normal behaviour (Section 2.1), and the dynamic thresholding technique (Section 2.2) for detecting anomalies with respect to the prediction of the ESN.

2.1. Echo State Networks

Echo State Networks (ESNs) [19, 20] are an efficient ML approach for temporal data. They belong to the family of Recurrent Neural Networks (RNNs), but are based on the exploitation of the network activations from the standpoint of a discrete-time dynamical system. Broadening the perspective, leveraging on the evolution of a recurrent network as a dynamical system represents the core idea of the so-called Reservoir Computing paradigm [21, 22].

The architecture of ESNs is made up by two main components: the first is the recurrent network that holds an internal state which evolves over time, which is called *reservoir*; the second, called *readout*, is a linear layer that takes as input a state of the reservoir and emits a prediction. Formally, let $\boldsymbol{x}(t) \in \mathbb{R}^{N_R}$ denote the state of a reservoir with N_R recurrent units at a given time step t. Then, the evolution of the state for an input sequence of vectors $\mathbf{u}(1), \dots, \mathbf{u}(t) \in \mathbb{R}^{N_U}$ in a reservoir of leaky-integrator neurons [23] can be described as

$$\begin{aligned} \boldsymbol{x}(t) &= (1-a)\,\boldsymbol{x}(t-1) + a \tanh\left(\mathbf{W}_{in}\mathbf{u}(t) + \theta + \hat{\mathbf{W}}\boldsymbol{x}(t-1)\right) \\ \boldsymbol{y}(t) &= \mathbf{W}\boldsymbol{x}(t) + \theta_{out} \quad . \end{aligned} \tag{1}$$

where $\mathbf{W}_{in} \in \mathbb{R}^{N_R \times N_U}$ is the input-to-reservoir weight matrix, $\hat{\mathbf{W}} \in \mathbb{R}^{N_R \times N_R}$ is the recurrent reservoir-to-reservoir weight matrix, $\theta \in \mathbb{R}^{N_R}$ is the reservoir bias term, $\mathbf{W} \in \mathbb{R}^{N_Y \times N_R}$ is the readout weight matrix, θ_{out} is the output bias term, $a \in (0, 1]$ is the leaking rate and $\boldsymbol{x}(0) = \mathbf{0}$, y(t) is the output of the model at timestep t.. ESNs were generalized to deep architectures with Deep Echo State Networks (DeepESN) [24], in which multiple reservoir layers are stacked one on top of each other. With such generalization, the state transition function of reservoir becomes layer-dependent, i.e. at layer l, the input of the reservoir is the input sequence itself for $l = 1$, and the sequence of hidden states of the layer $l - 1$ for $l > 1$.

The main advantage of shallow and deep ESNs is that, instead of backpropagating the error signal through time [25], the input-to-reservoir $\mathbf{W}_{in}$ and the reservoir-to-reservoir $\hat{\mathbf{W}}$ matrices are left *fixed*, and only the readout parameters $\mathbf{W}$ are trained. This comes at the single cost of constraining the spectral radius[1] of the reservoir-to-reservoir matrix $\rho(\hat{\mathbf{W}})$ to a value smaller than 1 to meet the necessary condition for ensuring stability of the reservoir dynamics [19, 26]. This constraint implies a commonly used naïve approach for reservoir initialization, consisting in a random generation of the weight values in $\hat{\mathbf{W}}$, followed by a re-scaling by a term $\rho_{desired} \,/\, \rho(\hat{\mathbf{W}})$, to finally achieve a randomized $\hat{\mathbf{W}}$ with a desired spectral radius $\rho(\hat{\mathbf{W}}) = \rho_{desired}$. This approach requires the computation of eigenspectrum of the recurrent reservoir matrix $\hat{\mathbf{W}}$, which might become a costly operation and a computational bottleneck in real-world complex applications requiring large reservoirs. To overcome this difficulty, the authors of [27] proposed a fast spectral radius initialization rooted in random matrix theory and based on the *circular law*. Specifically, the reservoir initialization strategy proposed in [27] consists in randomly drawing the weights in $\hat{\mathbf{W}}$ from a uniform distribution over $[-\rho_{desired}\sqrt{3 \,/\, N_R}, \rho_{desired}\sqrt{3 \,/\, N_R}]$, which ensures a value of $\rho(\hat{\mathbf{W}}) \to \rho_{desired}$ in the $N_R \to \infty$ limit. In this paper, we take advantage of this simplification and concretely show the computational benefit in the tackled application scenario. Furthermore, notice that the forward computation efficiency is provided by the fact

[1] The maximum of the eigenvalues in modulus.

that both $\mathbf{W}_{in}$ and $\hat{\mathbf{W}}$ are typically sparse matrices, which allows performing efficient matrix operations.

Finally, both ESN and DeepESN have proven to be much more efficient than the popular, end-to-end trainable recurrent models, often with the gain of better performance on the task at hand [28]. For this reason, they represent our models of choice for providing a more efficient, yet effective, prediction on the telemetry data.

2.2. Anomaly Detection via Nonparametric Dynamic Thresholding

In this section, we describe the dynamic thresholding technique proposed by [18] and used in this paper. The algorithm detects anomalous behaviours of spacecrafts by processing a stream of multivariate telemetry data. The algorithm for performing outlier detection consists in three major steps: (1) telemetry value prediction; (2) dynamic error thresholding; (3) false positives mitigation.

Telemetry Value Prediction. This step consists in a supervised learning task for predicting the telemetry data. Formally, the telemetry data is modelled as a set of C channels. Each channel $c \in [1, \dots, C]$ is a time series $X^c = \{\boldsymbol{x}(1), \boldsymbol{x}(2), \dots, \boldsymbol{x}(n^c)\}$ where each step $\boldsymbol{x}(t) \in \mathbb{R}^{m^c}$ is an m^c-dimensional vector $[x_1(t), x_2(t), \dots, x_{m^c}(t)]$, whose elements correspond to scalar input variables. When performing a prediction task in the temporal domain, we must consider three main parameters: the length of the sliding window l_s for the input subsequence $X^c_{l_s} = \{\boldsymbol{x}(t-1-l_s), \boldsymbol{x}(t-l_s), \dots, \boldsymbol{x}(t-1)\}$; the temporal horizon l_p for denoting the length of the subsequence $X^c_{l_p} = \{\boldsymbol{x}(t), \boldsymbol{x}(t+1), \dots, \boldsymbol{x}(t+d-1)\}$ to be predicted; the number of dimensions $d \in \{1, \dots, m^c\}$ to be predicted for each time step.

In our scenario, we employed Echo State Networks for predicting values under a normal behaviour, which are then used as a proxy to perform contextual outlier detection. As already pointed out by [18], recurrent models typically struggle when dealing with the prediction of high-dimensional outputs, implying that it is not feasible to model all the C channels with a single model. For this reason, we model each channel of the telemetry data independently by instantiating one ESN for each of them. As a result, our sequence prediction model will be instantiated as an ensemble of C ESNs, each with $N^c_R = m^c$. This setting provides a more granular view of spacecraft anomaly patterns, and allows to model all the C learning tasks independently. For all the models, the value of d is fixed to 1 for predicting only the actual telemetry value, l_p is fixed to 1 for performing next-step prediction, and l_p is fixed to 250.

Dynamic Error Thresholding. The second step is an unsupervised learning task based on a nonparametric method, which employs the cumulative error between predictions and the observed values for performing outlier detection. In particular, considered a single channel and given a predicted value $\hat{\boldsymbol{y}}(t)$, the prediction error at time t is computed as $e(t) = |\boldsymbol{y}(t) - \hat{\boldsymbol{y}}(t)|$, where $\boldsymbol{y}(t) = \boldsymbol{x}_i(t+1)$ with i corresponding to observation of the i-th dimension of the telemetry data. Errors $e(t)$ are accumulated in a vector of errors, i.e., $\boldsymbol{e} = [e(t-h), \dots, e(t-l_s), \dots, e(t-1), e(t)]$ where $h \geq l_s$ is the number of historical error values used to evaluate the current errors. In order to ensure the consistency of the error evaluation technique in the face of sharp spikes

in error values, we apply an exponentially-weighted moving average (EWMA) to produce a smoothed vector of errors $\boldsymbol{e}_s = [e_s(t-h), \dots, e_s(t-l_s), \dots, e_s(t-1), e_s(t)]$.

The thresholding technique works under the assumption that the errors, under either anomalous or non-anomalous behaviour, follow two distinct normal distributions respectively. Formally, a set of possible thresholds $\epsilon = \{\epsilon(z_1), \epsilon(z_2), \dots, \epsilon(z_k)\}$ is defined. Each threshold $\epsilon(z_i) \in \epsilon$ is formulated as $\epsilon(z_i) = \mu(\boldsymbol{e}_s) + z_i\sigma(\boldsymbol{e}_s)$, where z_i represents the tolerance of the threshold, i.e., the number of standard deviations above $\mu(\boldsymbol{e}_s)$. The different values z_i allow to scale the tolerance of the threshold to potentially anomalous observations: lower values make the threshold more restrictive, while larger values make it more tolerant. In [18], they found that an optimal range of values of z_i for their task was $[2, 10]$ (i.e., values outside the range led to many false positive or false negatives). Then, the current optimal threshold is determined by the following formula:

$$\epsilon^* = \arg\max_{z_i} \frac{\Delta_\mu(\boldsymbol{e}_s, z_i)/\mu(\boldsymbol{e}_s) + \Delta_\sigma(\boldsymbol{e}_s, z_i)/\sigma(\boldsymbol{e}_s)}{|\boldsymbol{e}_a(z_i)| + |\mathbf{E}_{seq}(z_i)|^2} \tag{2}$$

such that

$$\begin{aligned}
&\Delta_\mu(\boldsymbol{e}_s, z_i) = \mu(\boldsymbol{e}_s) - \mu(\{e_s \in \boldsymbol{e}_s \,|\, e_s < \epsilon(z_i)\}) \\
&\Delta_\sigma(\boldsymbol{e}_s, z_i) = \sigma(\boldsymbol{e}_s) - \sigma(\{e_s \in \boldsymbol{e}_s \,|\, e_s < \epsilon(z_i)\}) \\
&\boldsymbol{e}_a(z_i) = \{e_s \in \boldsymbol{e}_s \,|\, e_s > \epsilon(z_i)\} \\
&\mathbf{E}_{seq}(z_i) = \text{number of continuous sequences of } e_a \in \boldsymbol{e}_a(z_i)
\end{aligned}$$

The optimal threshold ϵ^* is the one providing the best separation between the distributions of non-anomalous and anomalous data (numerator), while penalizing an excessive skewing of the data on the distribution of outliers (denominator) to prevent an overly greedy behaviour. Once ϵ^* is determined, each resulting anomalous sequence of smoothed errors $\boldsymbol{e}_{seq} \in \mathbf{E}_{seq}$ is given an anomaly score $s^i = (\max(\boldsymbol{e}^i_{seq}) - \epsilon^*)/(\mu(\boldsymbol{e}_s) + \sigma(\boldsymbol{e}_s))$, indicating the severity of the anomaly.

Pruning False Positives. The purpose of this last step is to trade-off between (1) the number of historical errors h to query from for keeping a good efficacy in prediction, and (2) a feasible false positive rate. In particular, we introduce an additional vector $\boldsymbol{e}_{max}$ which contains the maximum value of each anomalous subsequence sorted in descending order, i.e., $\boldsymbol{e}_{max} = [\max(\boldsymbol{e}_s) \text{ for all } \boldsymbol{e}_s \in \mathbf{E}_{seq}]$ such that $\boldsymbol{e}_{max}[i] \geq \boldsymbol{e}_{max}[j]$ if $i > j$. At the end of the vector, the maximum smoothed error which is not anomalous is appended, i.e., $\max(\{e_s \in \boldsymbol{e}_s \in \mathbf{E}_{seq} \,|\, e_s \notin \boldsymbol{e}_a\})$. Then, the sequence is stepped through incrementally, and for each step $i \in \{1, 2, \dots, (|\mathbf{E}_{seq}| + 1)\}$, we compute $d^i = e^{i-1}_{max} - e^i_{max}/e^{i-1}_{max}$, which denotes the percentage decrease between subsequent errors in $\boldsymbol{e}_{max}$. We define another threshold p as the minimum percentage decrease expected. If, at some step i, p is exceeded by d^i, then all the errors $e^j_{max} \in \boldsymbol{e}_{max} \,|\, j < i$ and their corresponding anomaly sequences are confirmed to be anomalies. Otherwise, if $d^i < p$ and the same condition holds for all $d^{i+1}, \dots, d^{|\mathbf{E}_{seq}|+1}$, the corresponding error sequences are reclassified as nominal. A correct choice of the threshold p provides a good distinction between errors which are caused by regular noise within the stream, and the actual anomalies which occurred in the system.

3. Experimental Setup

In this work, we addressed the problem of performing anomaly detection in on-the-edge applications, which typically leverage low-powered devices with limited memory and computational budget. For this reason, we experimented with LSTMs, one of the most popular choices in the literature, and ESNs, which are known to be much more efficient to train. The objective is to evaluate the trade-off between the performance on the task and computational cost achieved by the two models in the face of computational constraints.

The experimental setup was tailored to fit the same workflow of the methodology described in Section 2. In particular, we split the experimental assessment in two main phases. In the *Telemetry Value Prediction* phase, we select the best model for performing the supervised task of telemetry value prediction on the training set. We evaluate LSTMs and ESNs in both shallow (1-layer) and deep (2-layer) versions, and for the four resulting architectures we set an upper bound to the number of free parameters to simulate hardware constraints.

In the *Anomaly Detection* phase, we evaluate the performance of the method on the anomaly detection task on the test set by following the same workflow of [18]: after a retraining phase, we apply the nonparametric method with different values of p for mitigating false positives.

3.1. Dataset Description

To provide a fair comparison with the existing method based on LSTMs [18], we based our experiments on the dataset derived from *Incident Surprise, Anomaly* (ISA) reports from NASA, in which experts annotated unexpected events (anomalies) which could put at risk the operations of the spacecrafts. In particular, we consider the reports for the Soil Moisture Active Passive (SMAP) satellite, and the Mars Science Laboratory (MSL) rover (known as Curiosity)[2].

The dataset consists in 82 telemetry channels identified with an alphanumerical ID which determines the channel type (e.g., P-1= First Power channel). Each channel is a matrix where the first dimension indicates the timesteps and the second dimension indicates the input features. In particular, the first feature represents the values of the telemetry, while the others are commands sent or received by the module delegated for gathering such data. All the numerical values are normalized to lie in the range $(-1, 1)$, while commands are one-hot encoded. There is neither information about the sampling frequency of the channel, nor the nature of the commands.

The dataset is provided with a training and test split, where the training set contains only non-anomalous sequence, while the test set contains anomalous subsequences for evaluating the method on the anomaly detection task. At the time when we performed the experiments, anomalies for the channel $T - 10$ were not provided. Therefore, we decided to exclude it from the following evaluation. In Table 1 we provide a summary of the main statistics of the dataset.

3.2. Phase I: Telemetry Value Prediction

In this phase, we select the best model for performing the supervised task of telemetry value prediction on the training set. As previously mentioned, we assessed a shallow (1-layer)

[2]More detailed descriptions of the operations are available at https://smap.jpl.nasa.gov/mission/description/ and https://mars.nasa.gov/msl/mission/overview/.

Table 1
Dataset information summary.

	SMAP	MSL	Total
Total anomaly sequences	69	36	105
Point anomalies (% tot.)	43 (62%)	19 (53%)	62 (59%)
Contextual anomalies (% tot.)	26 (38%)	17 (47%)	43 (41%)
Unique telemetry channels	55	27	82
Unique ISAs	28	19	47
Telemetry values evaluated	429,735	66,709	494,444

Table 2
Hyperparameter employed in the model selection

LSTM		
Hidden layers	1	2
Hidden layer units	10 - 67	10 - 42
Dropout	0.1 - 0.5	
Learning rate	0.01, 0.001, 0.0001	

ESN	
Hidden layers	1,2
Hidden layer units	100 - 2550
Input Scaling	0.01 - 1.0
Spectral Radius	0.1 - 0.99
Learning rate	0.01, 0.001, 0.0001

configuration of LSTMs and ESNs, as well as a deep one (2-layer), resulting on four configurations. All the hyperparameters were chosen under a simulated hardware constraint. In particular, the maximum number of free parameters for each model, set to 25 000, is equal to provide a fair comparison. As a result, for ESN networks the maximum number of units was set to $N_R = 2550$, for networks with one or two layers, since having multiple hidden layers does not change the number of free parameters (i.e., the readout is applied to the last reservoir only). Instead, LSTMs with a single layer use up to 67 units, while LSTMs with two hidden layers have a maximum of 42 units. The values of hyperparameters are summarized in Table 2. We optimized all the models by applying Adam [29] on the Mean Squared Error (MSE) as loss function with the use of early stopping, for a maximum of 150 epochs.

As mentioned in section 2.2, we istantiate one model for each channel to fit its own dynamics. Thus, we performed a model selection for each channel, consisting in a Random Search [30] with 50 random configurations on the hyperparameters' space. The model selection was implemented and performed with Keras Tuner [31].

The data was preprocessed as in [18]. We processed each channel with sliding windows of length 250 to create a training and test set consisting of n sub-sequences extracted from the original channels.

3.3. Phase II: Anomaly Detection Assessment

In this phase, for each channel, we first retrained the configurations chosen upon model selection for 5 times to average the performance in the face of bad initializations[3], and fed the test data

[3]To guarantee the reproducibility of this phase, all the random seeds for initializing the models' weights were stored

to the resulting models for producing the sequence of predicted values. Then, we applied the thresholding technique on the resulting sequence of errors. For this purpose, the method requires the definition of a set of threshold parameters $\mathbf{z}$, which, informally, determine the tolerance of error between the predicted and the observed values. As reported in [18], values of $z_i \in \mathbf{z}$ less than 2 are too restrictive, resulting in a high number of false positives. In our study the possible values for z_i are within a range $[2.5, 12]$. Finally, we applied the false positives mitigation technique, which requires the definition of a threshold parameter p. We evaluated the performance in the anomaly detection task with respect to values of p in the range $[0.01, 1]$.

4. Experimental Results

In this section, we show and compare the results obtained for both the telemetry value prediction and anomaly detection tasks (Section 4.1), as well as the computation efficiency (Section 4.2) of the four architectures.

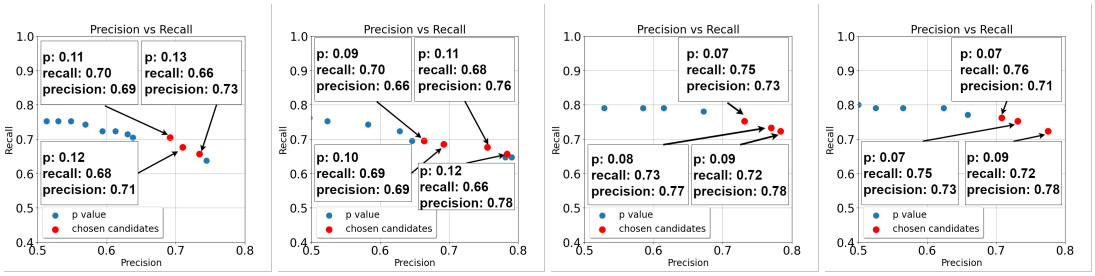
4.1. Prediction Performance

All the results concerning the performance on the predictions of the telemetry values, as well as anomalies, are summarized in Table 3. In particular, we want to stress that the aim of the experiments is to compare the performance of the different models under the same computational resource and memory constraints. This is highlighted by the second column of the table, in which we show the number of parameters for each of the models.

Telemetry Value Prediction. In Table 3 we report the average and standard deviation of the Mean Squared Error on the validation set over 5 training processes of the best configuration for each architecture. We can observe that, at parity of free parameters, all the architectures can reach a good quality in the prediction of the telemetry data values. Also, we can observe that the performance of the ESNs is almost the same, independently from the number of layers.

Table 3
Summary of the results. For the telemetry value prediction task, we report the MSE on the validation set averaged over 5 training processes of the selected configuration. For the anomaly detection task, we report the number of True Positives (TP), False Positives (FP), False Negatives (FN), the precision, recall and F_1 score on the final run, which is performed with the best value of p and the best configuration for the corresponding architecture.

Model	Max Params	Telemetry Value Prediction	Anomaly Detection						
		MSE×10^3	p	TP	FP	FN	Precision	Recall	F_1 score
LSTM 1-layer	25 604	6.91 ± 0.55	0.11	72	34	33	0.68	0.69	0.68
LSTM 2-layer	26 134	5.78 ± 0.27	0.09	79	42	26	0.65	0.75	0.69
ESN 1-layer	25 510	8.41 ± 0.26	0.07	78	34	27	0.70	0.74	0.71
ESN 2-layer	25 510	8.53 ± 0.31	0.07	78	23	27	0.77	0.74	0.76



(a) LSTM 1 layer. (b) LSTM 2 layers. (c) ESN 1 layer. (d) ESN 2 layers.

Figure 1: Precision and recall for different values of p. Candidates values of p which present the best precision-recall trade-off are highlighted in red.

Anomaly Detection. We evaluated the anomaly detection performance with respect of different threshold parameters p, which, as mentioned in Section 2.2, represents the minimum decrease in percentage that is expected between two points marked as anomalous. Changing this threshold adjusts the ratio of false positives and false negatives and influences the model's precision and recall. For this application, it is preferable to use a threshold value which balances the trade-off such that as many anomalies as possible are identified, therefore preferring recall, but trying to keep precision high at the same time so as not to have too many false positives. Therefore, we set a threshold for precision and recall of 70% when possible, at 65% otherwise. Figure 1 shows the precision/recall ratio for the first 20 threshold values which provide the best trade-off for our scenario. For each of the architectures, we chose a threshold p for re-evaluating the anomaly detection pipeline. In particular, for LSTM networks with one layer is 0.11, while the value chosen for LSTM networks with a two layers is 0.09. For ESNs, we chose a threshold of 0.07 for both the shallow and deep architectures. The chosen values of p are summarized in Table 3.

Pipeline re-evaluation. After conducting the model selection for the networks and finding the ideal threshold parameter we proceed with a final run that includes the training, prediction, and anomaly detection phases. The results are summarized in Table 3, which includes false positives, false negatives and true positives, precision, recall and F_1. From these results, we can observe that the similar behaviour of the four architectures in the telemetry value prediction is not supported by the same behaviour in the anomaly detection task. In particular, we can observe that, at parity of free parameters, the ESN with two reservoir provides the best performance, with precision 0.77, recall 0.74 and F_1 0.76. The differences between the models highlight that ESNs, with constrained hardware resources and model complexity, are more capable of modelling the dynamics of the channels under a non-anomalous behaviour of the input sequence.

4.2. Computational Efficiency

For each phase of the anomaly detection pipeline, we measured the corresponding elapsed time for each architecture, and are summarized in Table 4.

Initialization time. The use of the circular law allows to initialize the reservoir efficiently (Section 2.1). The results show large computational savings, with the build time going from more than half an hour to less than one second when using the circular law. This is due to the computation of the eigenspectrum of the reservoir weight matrix, which is needed to rescale the matrix to the desired spectral radius value. This is an expensive operation that dominates the network's initialization cost.

Training time. Table 4 shows a summary of the training times for each architecture. Notice that the times shown in the table sums of the training times for all the channels. In particular, we measured that a single SGD step costs ~3ms for ESNs with two stacked reservoirs, which *is almost an order of magnitude faster* than the ~20ms of LSTMs with two recurrent layers. This was possible by feeding all the sequences to the reservoirs once, and, training only the readout by taking in input the pre-processed sequences.

Table 4
Computational efficiency of the four models. For the reservoir initialization and training times, we report the median time. For model selection and re-evaluation we report the corresponding times for their single execution.

Model	Training	Model Selection	Re-evaluation
LSTM 1 layer	0h:17m	18h:25m	23m:23s
LSTM 2 layer	0h:29m	33h:32m	36m:39s
ESN 1 layer	0h:03m	2h:49m	14m:43s
ESN 2 layer	0h:04m	3h:40m	23m:04s

Reservoir matrix build times	
Units	2550
Layers	1
No Circular Law	32m:20s.435ms
Circular Law	00m:00s.146ms

5. Conclusion

In this paper, we showed how to perform anomaly detection efficiently by exploiting ESNs to model the normal behavior of temporal data, and dynamic thresholding to detect anomalies. In particular, we showed that ESNs are 6x faster to train compared to similar LSTMs in anomaly detection applications. More importantly, despite the lower computational cost, the model also achieves better precision and recall. These results enable embedded application with training on-the-edge, which may be impractical with more expensive models such as LSTMs. Even outside the embedded domain, more efficient training algorithms can enable novel application of anomaly detection systems. Such models can enable human-centric applications, such as suggestions for pilots during flights, or stress monitoring of passengers for autonomous driving vehicles [1, 2].

In the future, we plan to extend this work to different cyber-physical systems applications, such as those in the automotive and avionics domain. We also plan to integrate the ESN-based anomaly detection system into an AI-as-a-Service system, where the computational advantages can be exploited on distributed hardware such as the NVIDIA Jetson Nano and the NXP i.MX 8 boards, both carrying System on a Chip (SoC) processors.

Acknowledgments

This work is supported by the TEACHING project funded by the EU Horizon 2020 under GA n. 871385.

References

[1] D. Bacciu, S. Akarmazyan, E. Armengaud, M. Bacco, G. Bravos, C. Calandra, E. Carlini, A. Carta, P. Cassara, M. Coppola, C. Davalas, P. Dazzi, M. C. Degennaro, D. Di Sarli, J. Dobaj, C. Gallicchio, S. Girbal, A. Gotta, R. Groppo, V. Lomonaco, G. Macher, D. Mazzei, G. Mencagli, D. Michail, A. Micheli, R. Peroglio, S. Petroni, R. Potenza, F. Pourdanesh, C. Sardianos, K. Tserpes, F. Tagliabò, J. Valtl, I. Varlamis, O. Veledar, TEACHING – Trustworthy autonomous cyber-physical applications through human-centred intelligence, arXiv:2107.06543 [cs] (2021). arXiv:2107.06543.

[2] V. De Caro, S. Bano, A. Machumilane, A. Gotta, P. Cassará, A. Carta, R. Semola, C. Sardianos, C. Chronis, I. Varlamis, K. Tserpes, V. Lomonaco, C. Gallicchio, D. Bacciu, AI-as-a-Service Toolkit for Human-Centered Intelligence in Autonomous Driving, arXiv:2202.01645 [cs] (2022). arXiv:2202.01645.

[3] D. Bacciu, A. Carta, D. D. Sarli, C. Gallicchio, V. Lomonaco, S. Petroni, Towards Functional Safety Compliance of Recurrent Neural Networks, in: Proceedings of the 1st International Conference on AI for People: Towards Sustainable AI, CAIP 2021, 20-24 November 2021, Bologna, Italy, 2021.

[4] J. L. Elman, Finding structure in time, Cognitive science 14 (1990) 179–211.

[5] S. Hochreiter, The vanishing gradient problem during learning recurrent neural nets and problem solutions, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 6 (1998) 107–116.

[6] S. Hochreiter, J. Schmidhuber, Long Short-Term Memory, Neural Computation 9 (1997) 1–32. doi:10.1144/GSL.MEM.1999.018.01.02.

[7] A. Carta, A. Sperduti, D. Bacciu, Encoding-based memory for recurrent neural networks, Neurocomputing (2021). doi:10.1016/j.neucom.2021.04.051.

[8] M. Arjovsky, A. Shah, Y. Bengio, Unitary Evolution Recurrent Neural Networks, in: ICML, 2015.

[9] X. Wang, Y. Jin, K. Hao, A Gated Recurrent Unit based Echo State Network (2020) 7.

[10] P. Malhotra, L. Vig, G. M. Shroff, P. Agarwal, Long Short Term Memory Networks for Anomaly Detection in Time Series, in: ESANN, 2015.

[11] L. Bontemps, V. L. Cao, J. McDermott, N.-A. Le-Khac, Collective Anomaly Detection based on Long Short Term Memory Recurrent Neural Network, arXiv:1703.09752 [cs] (2017). arXiv:1703.09752.

[12] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, G. Shroff, LSTM-based Encoder-Decoder for Multi-sensor Anomaly Detection, arXiv:1607.00148 [cs, stat] (2016). arXiv:1607.00148.

[13] H. Jaeger, Short Term Memory in Echo State Networks, volume 5, GMD-Forschungszentrum Informationstechnik, 2001.

[14] A. Cossu, D. Bacciu, A. Carta, C. Gallicchio, V. Lomonaco, Continual Learning with Echo State Networks, in: ESANN 2021, 2021. arXiv:2105.07674.
[15] C. Gallicchio, A. Micheli, L. Pedrelli, Design of deep echo state networks, Neural Networks 108 (2018) 33–47. doi:10.1016/j.neunet.2018.08.002.
[16] D. Bacciu, D. Di Sarli, P. Faraji, C. Gallicchio, A. Micheli, Federated Reservoir Computing Neural Networks, in: 2021 International Joint Conference on Neural Networks (IJCNN), 2021, pp. 1–7. doi:10.1109/IJCNN52387.2021.9534035.
[17] A. Blázquez-García, A. Conde, U. Mori, J. A. Lozano, A review on outlier/anomaly detection in time series data, arXiv:2002.04236 [cs, stat] (2020). arXiv:2002.04236.
[18] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, T. Soderstrom, Detecting Spacecraft Anomalies Using LSTMs and Nonparametric Dynamic Thresholding, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, ACM, 2018, pp. 387–395.
[19] 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 148 (2001).
[20] H. Jaeger, H. Haas, Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication, science 304 (2004) 78–80.
[21] M. Lukoševičius, H. Jaeger, Reservoir computing approaches to recurrent neural network training, Computer Science Review 3 (2009) 127–149.
[22] D. Verstraeten, B. Schrauwen, M. D'Haene, D. Stroobandt, An experimental unification of reservoir computing methods, Neural Networks 20 (2007) 391–403.
[23] H. Jaeger, M. Lukoševičius, D. Popovici, U. Siewert, Optimization and applications of echo state networks with leaky-integrator neurons, Neural networks 20 (2007) 335–352.
[24] C. Gallicchio, A. Micheli, L. Pedrelli, Deep reservoir computing: A critical experimental analysis, Neurocomputing 268 (2017) 87–99. doi:10.1016/J.NEUCOM.2016.12.089.
[25] Y. Bengio, P. Simard, P. Frasconi, Learning long-term dependencies with gradient descent is difficult, IEEE Transactions on Neural Networks 5 (1994) 157–166.
[26] M. Lukoševičius, A practical guide to applying echo state networks, in: Neural networks: Tricks of the trade, Springer, 2012, pp. 659–686.
[27] C. Gallicchio, A. Micheli, L. Pedrelli, Fast Spectral Radius Initialization for Recurrent Neural Networks, in: Recent Advances in Big Data and Deep Learning, volume 1, Springer International Publishing, 2020, pp. 380–390.
[28] C. Gallicchio, A. Micheli, L. Pedrelli, Comparison between DeepESNs and gated RNNs on multivariate time-series prediction, arXiv:1812.11527 [cs, stat] (2019). arXiv:1812.11527.
[29] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).
[30] J. Bergstra, Y. Bengio, Random search for hyper-parameter optimization, Journal of Machine Learning Research 13 (2012) 281–305. URL: http://jmlr.org/papers/v13/bergstra12a.html.
[31] T. O'Malley, E. Bursztein, J. Long, F. Chollet, H. Jin, L. Invernizzi, et al., Keras Tuner, https://github.com/keras-team/keras-tuner, 2019.