

Online Extreme Learning on Fixed-point Sensor Networks

H.H.W.J. Bosman^{*†}, A. Liotta^{*}

^{*}Department of Electrical Engineering

Eindhoven University of Technology

P.O. Box 513, 5600MB, Eindhoven, The Netherlands

h.h.w.j.bosman, a.liotta @ tue.nl

G. Iacca[†], H.J. Wörtche[†]

[†]INCAS³

P.O. Box 797, 9400AT, Assen, The Netherlands

heddebosman, giovanniacca, heinrichwoertche @ incas3.eu

Abstract—Anomaly detection is a key factor in the processing of large amounts of sensor data from Wireless Sensor Networks (WSN). Efficient anomaly detection algorithms can be devised performing online node-local computations and reducing communication overhead, thus improving the use of the limited hardware resources. This work introduces a fixed-point embedded implementation of Online Sequential Extreme Learning Machine (OS-ELM), an online learning algorithm for Single Layer Feedforward Neural Networks (SLFN). To overcome the stability issues introduced by the fixed precision, we apply correction mechanisms previously proposed for Recursive Least Squares (RLS).

The proposed implementation is tested extensively with generated and real-world datasets, and compared with RLS, Linear Least Squares Estimation, and a rule-based method as benchmarks. The methods are evaluated on the prediction accuracy and on the detection of anomalies. The experimental results demonstrate that fixed-point OS-ELM can be successfully implemented on resource-limited embedded systems, with guarantees of numerical stability. Furthermore, the detection accuracy of fixed-point OS-ELM shows better generalization properties in comparison with, for instance, fixed-point RLS.

Keywords—Anomaly detection; Embedded Systems; Single Layer Feedforward Neural Networks; Extreme Learning Machine;

I. INTRODUCTION

Technological advances of the past decades have enabled the abundant and ubiquitous presence of electronic systems embedded in our daily lives. Researchers are driven by the vision of the Internet of Things, a world-wide network of such systems. The advances provide more capable embedded devices that allow some online processing, but resources, such as energy, are still limited.

Example of such systems are Wireless Sensor Networks (WSN) and Body Sensor Networks (BSN) [1]. These are embedded devices with limited resources that are tightly coupled to the environment. In the last decade, the WSN community has shifted its focus from systems to applications and deployments. The resulting long-lived deployments and increasing possibilities of sensors have made it possible to generate, log and analyze large amounts of data, as demonstrated in, e.g., GreenOrbs [2]. Online (distributed) processing of this data promises to reduce resource usage,

for instance by reducing communication only to the most interesting or necessary data.

One such way of information selection is anomaly detection, i.e., the detection of unexpected behavior, or patterns, in data. Incoming data are compared to models of the environment, created with either formal methods or from data. There is an increasing interest to detect anomalies [3]. A common example of which is fault detection, where a part of the system displays unexpected behavior. Also, environmental data may only be of interest to domain experts when it deviates from normal behavior.

In recent years, Extreme Learning Machine (ELM), a learning algorithm for Single-Layer Feedforward Neural Networks (SLFN), has emerged as a powerful nonlinear mapping method used in classification and regression [4]. A sequential modification based on Recursive Least Squares (RLS), called Online Sequential ELM (OS-ELM), provides the possibility for online processing with little memory usage and only slight computational overhead [5]. Implementation of this method on embedded systems provides a powerful online processing tool, for instance for anomaly detection. However, we find that due to its limited precision, the fixed-point arithmetic introduces stability issues, such as divergence.

In this work, we introduce and evaluate a fixed-point version of the original OS-ELM learning method for SLFN, showing its viability for embedded systems. We apply correction mechanisms, previously proposed for RLS, to overcome these stability issues. Furthermore, we assess the usage of OS-ELM for online anomaly detection in WSN with an extensive evaluation campaign. To keep our algorithm generic and application agnostic we use a common categorization of anomalies, as identified in literature [6]–[8], as types *spike*, *constant*, *noise* and *drift* (Fig. 1). *Spikes*

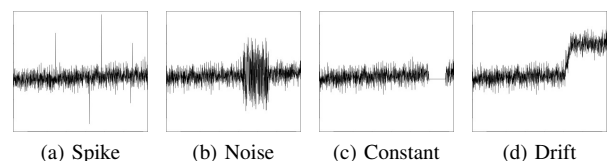


Figure 1. Injected anomalies in noisy linear data

are short-duration peaks in the measurement, for instance caused by buffer overruns. *Noise* is an unexpected change in variance which may, for example, be caused by electromagnetic interference. When changes and noise in the original signal do not lead to any changes in the measured signal, the anomaly is called *Constant*. This can, for instance, occur when a sensor is physically stuck. Finally, *drift* occurs when a measurement has a static or changing offset, relative to the original signal, e.g. by degradation of the sensor.

The following section reviews related work, after which Section III explains the proposed OS-ELM stabilization mechanisms and anomaly detection. Section IV shows that fixed-point OS-ELM estimation is similar in performance to the benchmark methods in terms of average prediction error and detecting anomalies. We elaborate this conclusion and projections of future possibilities in section V.

II. RELATED WORK

Anomaly detection, also known as fault detection or outlier detection, is of increasing interest to researchers and developers of a wide range of applications, from fraud detection for credit cards [9], intrusion detection in security [10] or military applications [11], to (sensor) fault detection in safety critical systems [12]. Traditionally, such anomaly detection systems process the data at a centralized processing facility, where all current and historical data are available. Even the anomaly detection techniques for distributed sensor systems, as surveyed in [3], [13], [14], are mostly centralized, or trained offline. Due to increasing computational power and larger volumes of sensor data, there is an increasing interest in anomaly detection on distributed embedded systems, such as WSN.

Such embedded systems have severe resource limitations, and thus online learning of models is difficult. However, online embedded machine learning online is still feasible. One of the most used techniques is Reinforcement Learning, for instance suitable for routing [15]. Another category of techniques are the biologically inspired algorithms called Swarm Intelligence. An example of which is ants that leave a trail of pheromones, strengthening edges in a graph [16]. There are also techniques based on statistics, such as a piecewise linear regression technique by Yao et al. [8]. The authors model a time series in linear segments and compare them to the segments of the previous day cycle. But, most anomaly detection systems have a training phase before deployment, such that data driven models can be trained. For instance, one can train and tune the parameters of an ensemble of classifiers, e.g., Neural Networks, offline [17]. Another example of offline training and online detection is given by Chang et al. [18], who train Echo State Networks to detect anomalies in time-series.

A Neural Network is most commonly trained using the backpropagation method, a gradient descent method [19]. The major drawback of backpropagation is that the training

set must be processed multiple times, thus resulting in slow learning. Furthermore, learning parameters have to be tuned iteratively for a specific dataset.

In 2004, Huang et al. proposed a new learning algorithm specific for SLFN, which randomly sets input weights and determines the output weights analytically. The authors called this algorithm Extreme Learning Machine (ELM), because “it makes learning extremely fast” [20]. Their research showed that ELM is efficient for continuous, non-differentiable and piecewise continuous activation functions [21]. Furthermore, it produces good generalization performance in artificial and real-world problems, without user tuning [22].

Liang et al. have proposed a version of ELM learning that processes the data online and sequentially, called Online Sequential ELM (OS-ELM) [5]. Where the original ELM method uses Ordinary Least Squares Estimation techniques to learn the output weights, the OS-ELM method uses Recursive Least Squares, as further described in Section III. OS-ELM permits online updating of the SLFN weights when, for instance, not all data are available offline. Both ELM learning methods, however, assume floating point precision for their algorithms to work, but floating point precision commonly cannot be used on lightweight embedded systems. Our work addresses this shortcoming.

III. METHOD

A. OS-ELM foundation

Let us consider a SLFN. A SLFN has the following structure, depicted graphically in Fig. 2:

$$y = f_{\tilde{N}}(\mathbf{x}) = \sum_{i=1}^{\tilde{N}} \beta_i G(\mathbf{a}_i, b_i, \mathbf{x}), \quad \mathbf{x}, \mathbf{a}_i \in \mathbb{R}^d, \quad b_i \in \mathbb{R} \quad (1)$$

where $\mathbf{a}_i$ and b_i are the input node weights and bias of the i th hidden node, $G(\mathbf{a}_i, b_i, \mathbf{x})$ is the output of that hidden node, and $\tilde{N}$ is the number of hidden nodes. Furthermore, $\mathbf{x} \in \mathbb{R}^d$ is a vector of input values of dimension d and β_i is the weight connecting the i th hidden node to the output node.

One can use several activation functions $g(x) : \mathbb{R} \rightarrow \mathbb{R}$ to determine the output of a hidden node. For example, for sigmoid functions (such as $\tanh(x)$) the hidden-node output is determined by:

$$G(\mathbf{a}_i, b_i, \mathbf{x}) = g(\mathbf{a}_i \cdot \mathbf{x} + b_i) \quad (2)$$

On the other hand, for radial basis functions (e.g. the Gaussian function), the hidden-node output is determined by:

$$G(\mathbf{a}_i, b_i, \mathbf{x}) = g(b_i \|\mathbf{x} - \mathbf{a}_i\|), \quad b_i \in \mathbb{R}^+ \quad (3)$$

Now, when we set the input weights and biases randomly and consider N training samples for equation 1, we can aggregate all hidden node outputs per training example in a

hidden-layer output matrix $\mathbf{H}$ of size $N \times \tilde{N}$. We can then rewrite the formula

$$y_j = f_{\tilde{N}}(\mathbf{x}_j) = \sum_{i=1}^{\tilde{N}} \beta_i G(\mathbf{a}_i, b_i, \mathbf{x}_j), \quad j = 1, \dots, N \quad (4)$$

as

$$\mathbf{Y} = \beta \mathbf{H} \quad (5)$$

Huang et. al showed that this can be considered a linear system [20]. One can find the hidden node weights β_i using the Moore-Penrose generalized inverse of the hidden-layer output matrix $\mathbf{H}$:

$$\hat{\beta} = \mathbf{H}^\dagger \mathbf{Y}$$

which is the least squares solution of equation 5. When $\text{rank}(\mathbf{H}) = \tilde{N}$, $\mathbf{H}^\dagger$ is given by:

$$\mathbf{H}^\dagger = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \quad (6)$$

Liang et al. showed that the least squares solution to $\hat{\beta}$, using equation 6, can also be obtained sequentially, using RLS [5]. The authors showed that the hidden node weights β^{k+1} can be recursively calculated according to the following formula:

$$\begin{aligned} \mathbf{P}_{k+1} &= \mathbf{P}_k - \mathbf{P}_k \mathbf{H}_{k+1}^T (\mathbf{I} + \mathbf{H}_{k+1} \mathbf{P}_k \mathbf{H}_{k+1}^T)^{-1} \mathbf{H}_{k+1} \mathbf{P}_k \\ \beta^{k+1} &= \beta^k + \mathbf{P}_{k+1} \mathbf{H}_{k+1}^T (\mathbf{Y}_{k+1} - \mathbf{H}_{k+1} \beta^k) \end{aligned} \quad (7)$$

Where $\mathbf{P}_k$ is the inverse auto-correlation matrix between the hidden nodes and the output node.

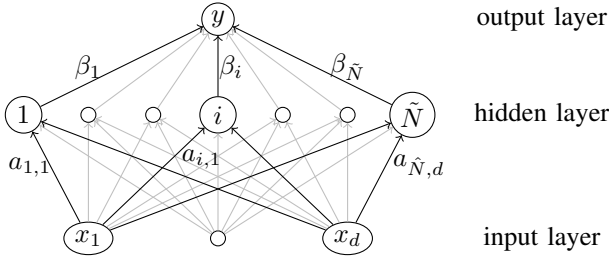


Figure 2. Structure of a Single Layer Feedforward Neural Network.

The above outlined method for learning output weights β_i is called Online Sequential Extreme Learning Machine (OS-ELM) and is described in [5]. In short, it has two phases:

Initialization phase:

- 1) Choose random input weights $\mathbf{a}_i$ and bias b_i (or center $\mathbf{a}_i$ and impact factor b_i in the case of RBF) for hidden nodes $i = 1, \dots, \tilde{N}$,
- 2) Estimate initial output weights β from a small initial number of training data, not less than the number of hidden nodes $\tilde{N}$.

Sequential Learning Phase:

- 1) Calculate $\mathbf{H}_{k+1}$, the partial hidden-layer output matrix,
- 2) Determine the desired output $\mathbf{Y}_{k+1}$,
- 3) Compute the output weight β_{k+1} using equation 7.

B. Stability correction mechanism

Due to its low memory requirements, OS-ELM is a viable candidate learning method to be implemented on embedded systems. Most embedded systems, however, deal only with limited precision, fixed-point variables, and thus, some quantization error is introduced. This quantization error, in turn, can lead to the overflow or underflow of a variable, which may cause the RLS method to become unstable and display, for example, explosive divergence.

We present a stability correction mechanism based on the work of Bottomley [23], who has analyzed the implications of quantizing the computations of RLS, which has been further evaluated by, e.g., Bosman et al. [24] in the context of anomaly detection. By adapting a subset of the correction methods suggested in [23], we obtained stable filters. Specifically, we do not round the results of fixed-point calculations, and we apply the following correction

$$\mathbf{C}(\mathbf{X}) = \alpha \mathbf{X} + \delta \mathbf{I} \quad (8)$$

to the update of the covariance matrix $\mathbf{P}_{k+1}$ in equation 7. Here, $\mathbf{I}$ is the identity matrix and δ is the smallest value the fixed-point implementation can represent. By adding this small number to the diagonals we bias the inverse auto-correlation matrix $\mathbf{P}_{k+1}$ to be positive definite. Contrary to [23], where only non-zero diagonal elements of $\mathbf{P}_{k+1}$ are biased, we bias both zero and nonzero diagonal elements, thus preventing them from underflow and allowing for a correct weight update. Finally, the variable α is a forgetting factor, which reduces explosive divergence.

C. Anomaly detection

With the online learning SLFN we predict sensor measurements, and we use the error of the predictions to detect anomalies. Anomalies are unexpected behavior, or patterns, in data. Using OS-ELM we create a SLFN predicting model for the measurements, and we assume that when an anomaly occurs, the output of a sensor will deviate significantly from its prediction. Furthermore, we assume the prediction error is Gaussian distributed, based around zero. We evaluate two methods to analyze these prediction errors ϵ , namely a static thresholding method, and an adaptive threshold method.

The static threshold is determined from an initial set of measurements, based on the maximum absolute prediction error $\epsilon_{max} = \max(|\epsilon|)$ of this set. The remaining data are classified anomalous when they exceed the threshold by a certain factor. This factor is chosen to be 3.29, such that 99.9% of the prediction error is within the confidence interval (i.e. normal), if we assume ϵ_{max} to be the standard deviation.

The other method is an adaptive threshold based on the average of the prediction error ϵ , using Exponentially Weighted Moving Average (EWMA) filters [25]. What follows is a concise description of this method, as proposed by Bosman et al. [24]. First, we determine the running mean of the

prediction error $\hat{\mu}_\epsilon = \text{EWMA}(\epsilon)$ and running absolute deviation $\delta_\epsilon = \text{EWMA}(|\epsilon - \hat{\mu}_\epsilon|)$ from the running mean. We then define the adaptive threshold as $f\delta_\epsilon$, i.e., running absolute deviation times a factor f , which is determined in an initialization period. The absolute deviation of the mean prediction error, $|\epsilon - \hat{\mu}_\epsilon|$, of any new measurement is then compared to the threshold. When this threshold is crossed, then we have detected an anomaly and flag the measurement as such, otherwise, we update $\hat{\mu}_\epsilon$ and δ_ϵ .

IV. EVALUATION

We evaluate fixed-point OS-ELM on the basis of resource usage, average prediction error, and its accuracy in detecting anomalies, using the datasets described below. For prediction error and anomaly detection performance we use 3 synthetic datasets and traces of a real-world WSN setup as described in the following sections. Furthermore, we compare the algorithm to fixed-point RLS and Linear Least Squares Estimation (LLSE) in the aforementioned respects.

A. Algorithm Complexity

Because we are investigating online learning methods, we analyze the method complexity per new training instance. That is, every time a measurement is taken, we analyze the complexity of updating the model with this new measurement. In contrast, given a fixed dataset of N samples, the overall complexity, in terms of CPU, can be given by multiplying by N . The resulting overall computational complexity is higher compared to an offline non-sequential approach, where all the data is readily available. Nevertheless, the online sequential learning requires only a fixed amount of memory (that of the model), thus allowing for an online embedded approach. The analysis is based upon the pseudo-code of the algorithms, and can be seen in Tab. I.

For RLS the complexity per update mostly stems from a number of covariance matrix operations, where the matrices are of size $d \times d$. For ELM, we see the RLS complexity return in terms of the number of hidden nodes $\hat{N}$ from the update, as per equation 7. Furthermore, the update of the $\mathbf{H}$ matrix is in the order of the number of inputs d times the number of hidden nodes $\hat{N}$. Thus, the quadratic complexity together with the resource limitations of the embedded system do not allow for a very complex SLFN. However, the analysis in section IV-C and Fig. 4 show that the number of hidden nodes does not have to be much bigger than the number of inputs to achieve a small prediction error. The following paragraphs describe an analysis of energy usage, to let application designers decide on a trade-off between communicating to a central processing facility or local computation.

We have used MSPSim [26], a simulator for the MSP430

Method	OS-ELM	RLS	LLSE
CPU	$O(\hat{N}d + \hat{N}^2)$	$O(d^2)$	$O(N_{hist}d^2)$
Memory	$O(\hat{N}d + \hat{N}^2)$	$O(d^2)$	$O(N_{hist}^2)$

Table I

COMPARISON OF ORDER OF COMPLEXITY DERIVED FROM ALGORITHMS. $\hat{N}$ IS THE NUMBER OF HIDDEN NODES, d THE NUMBER OF INPUTS, N_{hist} THE MEASUREMENT HISTORY LENGTH.

MCU found on the TelosB sensor node platform¹ [27], to determine values of a real implementation, as seen in Fig. 3. Our implementation is based on TinyOS, and compiled for the TelosB platform, with compiler flags set to optimize for code size. The plot of Fig. 3a shows that the number of hidden nodes $\hat{N}$ has the most impact on the CPU time used, as it scales exponential, in agreement with the complexity derived in Tab. I. Moreover, the fixed-point calculations are implemented in software and thus, contribute significantly to the overall computational time. The effect of the number of input nodes d on the CPU time is, however, not obvious. This may be due to the random weight initialization, or the iterative algorithm used to solve the inverse of a matrix needed in the update of P_{k+1} in equation 7.

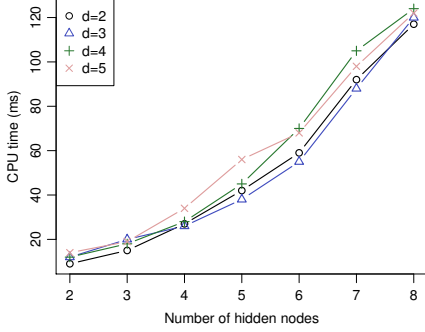
The CPU time can be translated to energy usage, expressed in millijoule (mJ), by determining the power consumption of the CPU used in milliwatts (mW) from the datasheet [28]. In the case of the MSP430, found on the TelosB sensor node, this is 14.4mW per second (s) when the CPU runs at 8MHz. The formula is then $\text{mJ} = \text{mW} * s$. When we take, for example, from Fig. 3a a CPU-time of 10ms for 2 hidden nodes, this results in 0.144mJ. And, if we take the CPU-time for 8 hidden nodes, being around 120ms, this would result in around 1.728mJ. In comparison, the radio chip on the TelosB, a TI CC2420, uses 52.2mW in transmit mode. Transmitting for 5ms would take 0.261mJ, but a message may require multiple nodes to forward the message to reach a central storage/processing facility, depending on the application and factors such as network size.

The memory usage mostly depends on the size of the matrices in memory. Due to implementation details, mainly matrix related [29], [30], this size does not grow proportional to the number of inputs or to the number of hidden nodes, as seen in Fig. 3b. Instead, in some cases a larger matrix size had to be chosen for the matrix algorithms to function. We do see, however, that the number of hidden nodes influences mainly the memory usage, which is largely consistent with Tab. I. But, we only see a difference in memory usage for different inputs when using 2 or 3 hidden nodes.

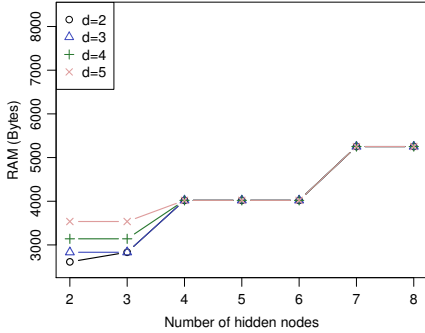
B. Datasets

To evaluate the modeling and prediction capabilities, we use synthetic and real-world datasets. Similar to the work of

¹TelosB, an 8bit MSP430 based platform with 10KB ROM, 48KB RAM and peripherals such as wireless 2.4GHz radio (TI CC2420), temperature, humidity (SHT11) and light sensors (Hamamatsu S1087 series).



(a) CPU Time



(b) RAM usage

Figure 3. CPU time and RAM usage of OS-ELM in a TinyOS program running on an MSP430 (determined with MSPSim).

Bosman et al., we use 3 synthetic datasets containing 28800 data points for 50 nodes, where each node has 3 sensors [24]. We assume that all sensors of a node have some linear correlation with each other. These sets are synthesized in different manners. The first set is composed of lines with different slope and bias, coupled with Gaussian noise with mean 0 and variance 0.1. The second set is similar to the first but also has an added sine-wave component with different amplitude per sensor. The last set is meant to resemble real-world temperature data and is created using several random walks, for e.g. trend, and a sine-wave component.

Furthermore, we use traces of a real-world indoor WSN setup. For this setup, 19 TelosB nodes were distributed inside our office building for 5 months. This dataset contains 42113 data points relating to the 19 nodes, each including 3 sensors: temperature, humidity and light. In this case we try to detect temperature anomalies using the humidity and light as inputs for the SLFN.

C. Prediction residuals

To determine the OS-ELM learning precision we analyze the root-mean-square (RMS) prediction error with different numbers of hidden nodes, and compare these results to RLS and LLSE. For the RLS stabilization parameters we have used forgetting factor $\alpha = 10$, as suggested by Bosman et al. [24]. However, our implementation of RLS for the ELM is different, thus for stabilization we have chosen α

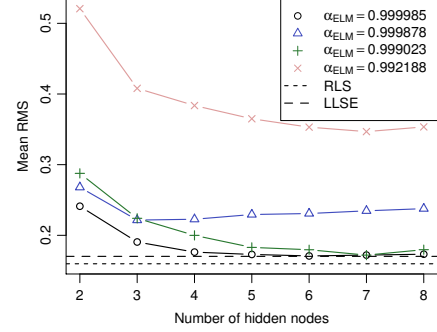


Figure 4. The average RMS error for different numbers of hidden nodes, with number of inputs $d = 3$

from one of 0.999985, 0.999878, 0.999023, to 0.992188, which are factors of power of 2 near a decimal, e.g. $1 - 1/2^{16} = 0.999985 \approx 0.99999$. We have computed the RMS prediction error four times over all 3 synthetic datasets per parameter, obtaining the result shown in Fig. 4. Because the OS-ELM and RLS algorithms need some time to converge, we have taken the second half from the synthesized data without injected anomalies. From Fig. 4 we see that the RLS algorithm has a lower RMS prediction error than LLSE, which might mean that it is slightly over fitting, as suggested by Bosman et al. [24]. Furthermore, we see that the RMS prediction error of ELM can converge to the prediction error of LLSE on the same datasets. However, this is not guaranteed, due to the random initialization of the hidden node weights and bias. This shows, for instance, in $\alpha = 0.999878$, where the RMS prediction error does not seem to converge to the LLSE prediction error, and for $\alpha = 0.992188$ we see a larger prediction error over the whole range of hidden nodes. The latter may be due to over-correcting OS-ELM, causing the algorithm to favor local data instead of generalizing the data.

D. Anomaly detection performance

To evaluate the performance of OS-ELM for anomaly detection, we compare it to RLS, LLSE and a rule-based baseline method. From the evaluation in sections IV-A and IV-C we determine that using an SLFN with $\hat{N} = 4$ hidden nodes should provide a good trade-off between accuracy and speed. For the RLS stabilization we have used parameters suggested by Bosman et al. [24], in particular the RLS forgetting factor $\alpha = 10$. The baseline rules are empirically determined and act on the first-order difference of the incoming data, as described in [24]. Specifically, when the difference between 3 or more consecutive measurements is zero, these data are flagged as *constant* anomaly. Furthermore, when the difference is larger than a determined threshold, they are flagged as *spike* anomaly.

From the results on the synthetic datasets, seen in Fig. 5a to 5d, we observe that the baseline indeed performs well

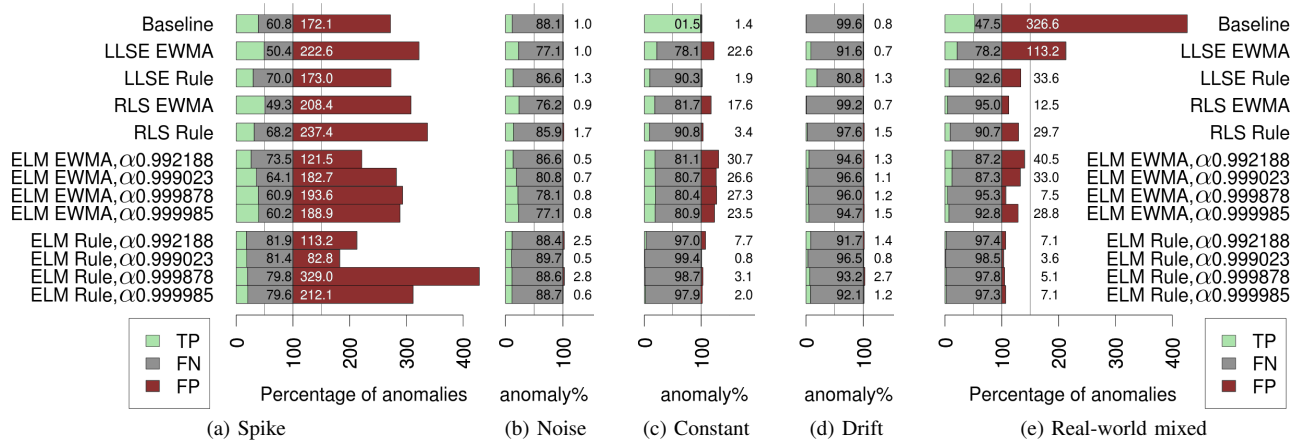


Figure 5. True Positives (TP), False Negatives (FN) and False Positives (FP) as the ratio of number of anomalies in dataset. The adaptive threshold is denoted *EWMA* and the static threshold is denoted *Rule*.

on *constant* anomalies, with 98.5% True Positives (TP), and also has reasonable performance for spike anomalies. All the regression methods, however, have lower performance on the *constant* anomalies, around 20% TP if we use the adaptive threshold. This is because the last known measurement may be relevant for a short time period. Moreover, the online learning will slowly adapt to the constant value, and thus the prediction error will go down over time. Nevertheless, we see that LLSE, RLS and ELM are detecting similar amounts of constant anomalies when using the adaptive threshold. On the other hand, this does not hold for the static threshold method, where ELM performs less.

Detecting *spike* anomalies results in a high False Positive (FP) ratio for all methods, including the baseline method, but we see that LLSE and RLS have higher TP ratios (around 50%) compared to the baseline method (39.2%), when using the adaptive threshold. The ELM with adaptive thresholding detection, however, does not surpass the baseline method with a little less than 40% TP. Again, the static threshold performs less well than its adaptive counterpart in all cases.

We see that the other anomalies have much lower FP ratios, commonly in the order of 3%. *Noise* anomalies, in particular, have FP ratios lower than 2.5%. The TP ratios for *noise* detection are similar to the TP ratio for the baseline when using the static threshold detection with all prediction methods. When using the adaptive threshold, the TP ratio increases to around 22% for all prediction methods. But, we see a decrease in TP ratio when the ELM forgetting factor α decreases, using the adaptive threshold. This decrease is caused by ELM learning more local data with lower α , such that the prediction error increases, and thus the threshold.

Drift anomalies are not detected very well by any method, because we have chosen to model this anomaly with a slow onset and offset, similar to, e.g., a degrading sensor calibration. When the forgetting rate is faster than the drift rate, this anomaly is likely to go undetected. Otherwise, only

the onset or offset of the *drift* are likely to be detected, before the model adapts through the online learning. This shows in the 20% TP ratio for the LLSE predictions with a static threshold, both of which do not adapt to the drift. Moreover, the FP ratio for all methods is very low with less than 3% overall, and in most cases less than 1.5%.

The online learning methods do adapt to *drift*, but, in this case ELM combined with adaptive thresholding has a higher TP rate (around 5%) compared to RLS (less than 1%), regardless of the learning rate α . This may indicate that the chosen forgetting factors for ELM are slower than the drift rate, whereas an earlier study showed a clear effect of the forgetting factor for RLS on the detection of drift [24]. For the anomalies that have a longer stable effect, the *drift* and *constant* anomalies, ELM displays more robustness to changes in the parameters of the correction method than RLS, which shows in more stable results.

The real-world dataset contains largely *constant* anomalies and some *spike* anomalies. The *constant* anomalies are mainly due to the unreliability of wireless connections. Furthermore, the *spike* anomalies in the real-world dataset are caused by direct sunlight hitting the temperature sensors. The large amount of *constant* anomalies shows in around 50% TP ratio for the baseline, but also results in a high FP ratio. From the other methods, the prediction model based on LLSE has the best performance of the learned models, with a 21.8% TP rate, using the adaptive threshold. RLS performs better with a static threshold for the real-world dataset, with 9.3% TP, while ELM performs better with an adaptive threshold. With $\alpha = 0.992188$ the TP ratio of ELM is 12.8%, but, with α approaching one the TP ratio goes down to 7.2%. However, the higher TP ratio for $\alpha = 0.992188$ also has a somewhat higher FP ratio compared to the static threshold method for RLS.

Overall, the OS-ELM forgetting factor α has a pronounced effect on the detection rate on the synthetic datasets,

for instance seen with *spike* and *noise* anomalies, this effect is not present in the real-world data. Furthermore, this effect is most noticeable when using adaptive thresholding. Due to this, we suggest to set α to values close to 1.

V. CONCLUSION

We have evaluated the performance of a fixed-point, OS-ELM trained, SLFN combined with static and adaptive thresholding for anomaly detection. For the used datasets RLS slightly outperforms ELM for the detection of *spikes* and *noise* anomalies, but ELM performs comparably on *constant* anomalies and outperforms RLS for *drift* anomalies. Overall, all ELM results are similar and, moreover, ELM can form a non-linear model with good generalization performance, without user tuning [22].

In all cases, ELM combined with a static threshold detects fewer anomalies, with higher false positives, than using other prediction methods. The static threshold method assumes that the average prediction error is zero. Investigating the predictions results for our datasets reveals, however, that ELM seems to have a slight bias. This leads us to conclude that other methods to analyze the prediction error may be more suitable for anomaly detection, which also shows in the higher performance of the adaptive thresholding method.

This work opens several areas of research to be investigated further. For one, data preprocessing could be applied, such that relevant features for anomaly detection are extracted. Another aspect to investigate, related to preprocessing, is the use of measurement history features, because environmental sensor data are usually time-series. Finally, this work has only investigated the results of node-local processing. Future work should investigate the incorporation of information available from neighboring devices, making use of the distributed nature of most embedded systems, such as WSN.

To conclude, we have demonstrated that fixed-point OS-ELM is a suitable online SLFN learning algorithm for embedded systems, when the correction mechanism described in Section III-B is applied to resolve the fixed-point stability issues. ELM allows for more complex, non-linear, online modeling of sensor data on embedded systems. Future work can further improve the anomaly detection results.

ACKNOWLEDGMENT

INCAS³ is co-funded by the Province of Drenthe, the Municipality of Assen, the European Fund for Regional Development and the Ministry of Economic Affairs, Peaks in the Delta.

REFERENCES

- [1] G. Fortino, M. Pathan, and G. Di Fatta, "Bodycloud: Integration of cloud computing and body sensor networks," in *Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on*, December 2012, pp. 851–856.
- [2] Y. Liu, Y. He, M. Li, J. Wang, K. Liu, L. Mo, W. Dong, Z. Yang, M. Xi, J. Zhao *et al.*, "Does wireless sensor network scale? a measurement study on greenorbs," in *INFOCOM, 2011 Proceedings IEEE*. IEEE, 2011, pp. 873–881.
- [3] R. Jurdak, X. Wang, O. Obst, and P. Valencia, "Wireless sensor network anomalies: Diagnosis and detection strategies," *Intelligence-Based Systems Engineering*, pp. 309–325, 2011.
- [4] S. Ding, H. Zhao, Y. Zhang, X. Xu, and R. Nie, "Extreme learning machine: algorithm, theory and applications," *Artificial Intelligence Review*, pp. 1–13, 2013.
- [5] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan, "A fast and accurate online sequential learning algorithm for feedforward networks," *Neural Networks, IEEE Transactions on*, vol. 17, no. 6, pp. 1411–1423, 2006.
- [6] K. Ni, N. Ramanathan, M. Chehade, L. Balzano, S. Nair, S. Zahedi, E. Kohler, G. Pottie, M. Hansen, and M. Srivastava, "Sensor network data fault types," *ACM Transactions on Sensor Networks*, vol. 5, no. 3, p. 25, 2009.
- [7] A. B. Sharma, L. Golubchik, and R. Govindan, "Sensor faults: Detection methods and prevalence in real-world datasets," *ACM Transactions on Sensor Networks*, vol. 6, no. 3, 2010.
- [8] Y. Yao, A. Sharma, L. Golubchik, and R. Govindan, "Online anomaly detection for sensor systems: A simple and efficient approach," *Performance Evaluation*, 2010.
- [9] C. Phua, V. Lee, K. Smith, and R. Gayler, "A comprehensive survey of data mining-based fraud detection research," *Arxiv preprint arXiv:1009.6119*, 2010.
- [10] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Macia-Fernandez, and E. Vazquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *computers & security*, vol. 28, no. 1-2, pp. 18–28, 2009.
- [11] K. Tiwari, M. Arora, and D. Singh, "An assessment of independent component analysis for detection of military targets from hyperspectral images," *International Journal of Applied Earth Observation and Geoinformation*, vol. 13, no. 5, pp. 730–740, 2011.
- [12] S. Budalakoti, A. Srivastava, and M. Otey, "Anomaly detection and diagnosis algorithms for discrete symbol sequences with applications to airline safety," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 39, no. 1, pp. 101–113, 2009.
- [13] Y. Zhang, N. Meratnia, and P. Havinga, "Outlier detection techniques for wireless sensor networks: A survey," *Communications Surveys & Tutorials, IEEE*, vol. 12, no. 2, pp. 159–170, 2010.
- [14] M. Xie, S. Han, B. Tian, and S. Parvin, "Anomaly detection in wireless sensor networks: A survey," *Journal of Network and Computer Applications*, 2011.
- [15] P. Wang and T. Wang, "Adaptive routing for sensor networks using reinforcement learning," in *Computer and Information Technology, 2006. CIT'06. The Sixth IEEE International Conference on*. IEEE, 2006, pp. 219–219.

- [16] M. Saleem, G. A. Di Caro, and M. Farooq, "Swarm intelligence based routing protocol for wireless sensor networks: Survey and future directions," *Information Sciences*, vol. 181, no. 20, pp. 4597–4624, 2011.
- [17] D.-I. Curiac and C. Volosencu, "Ensemble based sensing anomaly detection in wireless sensor networks," *Expert Systems with Applications*, vol. 39, no. 10, pp. 9087–9096, 2012.
- [18] M. Chang, A. Terzis, and P. Bonnet, "Mote-based online anomaly detection using echo state networks," in *Distributed Computing in Sensor Systems*. Springer, 2009, pp. 72–86.
- [19] R. Hecht-Nielsen, "Theory of the backpropagation neural network," in *Neural Networks, 1989. IJCNN., International Joint Conference on*. IEEE, 1989, pp. 593–605.
- [20] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks," in *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, vol. 2. IEEE, 2004, pp. 985–990.
- [21] G.-B. Huang, L. Chen, and C.-K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *Neural Networks, IEEE Transactions on*, vol. 17, no. 4, pp. 879–892, 2006.
- [22] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1, pp. 489–501, 2006.
- [23] G. E. Bottomley, "A novel approach for stabilizing recursive least squares filters," *IEEE Transactions on Signal Processing*, vol. 39, 1991.
- [24] H. H. W. J. Bosman, A. Liotta, G. Iacca, and H. J. Woertche, "Anomaly detection in sensor systems using lightweight machine learning," in *Systems, Man and Cybernetics*. IEEE, 2013, to appear.
- [25] S. Roberts, "Control chart tests based on geometric moving averages," *Technometrics*, vol. 1, no. 3, pp. 239–250, 1959.
- [26] J. Eriksson, A. Dunkels, N. Finne, F. sterlind, and T. Voigt, "MSPsim – an Extensible Simulator for MSP430-equipped Sensor Boards," in *Proceedings of the European Conference on Wireless Sensor Networks (EWSN), Poster/Demo session*, jan 2007. [Online]. Available: <http://dunkels.com/adam/eriksson07mbspim.pdf>
- [27] J. Polastre, R. Szewczyk, and D. Culler, "Telos: enabling ultra-low power wireless research," in *Proceedings of the 4th international symposium on Information processing in sensor networks*, ser. IPSN '05. Piscataway, NJ, USA: IEEE Press, 2005. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1147685.1147744>
- [28] Texas Instruments, "Msp430f15x, msp430f16x, msp430f161x mixed signal microcontroller (rev. g)," March 2011. [Online]. Available: <http://www.ti.com/product/msp430f1611>
- [29] P. Aimonen, "libfixmatrix - c library for fixed point matrix, quaternion and vector calculations," 2012. [Online]. Available: <https://github.com/PetteriAimonen/libfixmatrix>
- [30] B. Brewer, "libfixmath - cross platform fixed point maths library," 2012. [Online]. Available: <http://code.google.com/p/libfixmath/>