Towards Machine Learning-based Anomaly Detection on Time-Series Data

Daniel Vajda, Adrian Pekar, and Karoly Farkas

***Abstract*—The complexity of network infrastructures is exponentially growing. Real-time monitoring of these infrastructures is essential to secure their reliable operation. The concept of telemetry has been introduced in recent years to foster this process by streaming time-series data that contain feature-rich information concerning the state of network components. In this paper, we focus on a particular application of telemetry — anomaly detection on time-series data. We rigorously examined state-of-the-art anomaly detection methods. Upon close inspection of the methods, we observed that none of them suits our requirements as they typically face several limitations when applied on time-series data. This paper presents Alter-Re^2, an improved version of ReRe, a state-of-the-art Long Short-Term Memory-based machine learning algorithm. Throughout a systematic examination, we demonstrate that by introducing the concepts of ageing and sliding window, the major limitations of ReRe can be overcome. We assessed the efficacy of Alter-Re^2 using ten different datasets and achieved promising results. Alter-Re^2 performs three times better on average when compared to ReRe.**

***Index Terms*—anomaly detection, LSTM, neural network, time-series data, Alter-Re^2.**

I. INTRODUCTION

NOWADAYS, infrastructure monitoring, including networks, systems, and services, is more critical than ever before. It is essential for several reasons, such as alerting partial or total system malfunction, outage prevention based on predictive identification of such situations, performance tracking, and, last but not least, security detection of system penetration.

However, with the exponential increase in the number of interconnected devices and traffic volume, it has become far from obvious how to achieve timely, reliable, and sound infrastructure monitoring. It requires understanding the details of system processes and recognize how they influence each other or the whole infrastructure. The concept of network telemetry has been introduced to streamline this goal. It allows automated, fast, and simultaneous collection of a wide variety of time-series data from a large number of devices. However, processing massive data volumes is challenging, especially in terms of timeliness and scalability.

Machine learning techniques can process, understand, and classify problematic infrastructure behaviours, even in massive data volumes. Despite recent advances in machine learning, their application to anomaly detection remains poorly understood and investigated in the network telemetry domain. This paper attempts to shed new light on anomaly detection on time-series telemetry data.

Anomaly detection is a critical component of network and services management as it can provide valuable insights into the operation of the network and its components. Broadly speaking, measurement data are created by a generating process. If this generating process behaves unusually due to the system's abnormal behaviour or the entity that impacts the generating process, it produces anomalies. The manifestation of anomalous behaviour can be identified by examining the generated time-series data.

Our survey of anomaly detection on time-series data yielded ReRe [1], a Long Short-Term Memory (LSTM) [2] based machine learning algorithm, as one of the most promising state-of-the-art approaches. ReRe is claimed to achieve high accuracy in detecting abnormal behaviour while minimizing false positives and re-trainings. However, our evaluation revealed several limitations when ReRe was applied on time-series data. Thus, we introduce Alter-Re^2, an enhanced version of ReRe which extends the original algorithm with two additional features to improve its efficacy. The first feature ensures that the collected data ages out; thus, its weight decreases as time passes, allowing faster adaption to short-term history and more precise anomaly detection. The second feature serves the purpose of a sliding window that reduces the anomaly detector's resource demands.

The evaluation of our approach with ten different time-series datasets has shown promising results. Alter-Re^2 achieved approximately three times higher but never worse anomaly detection accuracy as opposed to ReRe. Not only could we eliminate issues obstructing real-time use, but we also enhanced sensitivity to detect anomalies with smaller amplitude and length. That said, we argue the strong applicability of Alter-Re^2 in real-world scenarios.

The rest of this paper is organized as follows. In Section II, we discuss related works in the field of anomaly detection on time series, especially network telemetry-related data. Section III describes two state-of-the-art time-series anomaly detection algorithms — RePAD [3] and ReRe [1]. In Section IV, we present Alter-Re^2, our approach to address the identified limitations. In Section V, we present experiments to examine Alter-Re^2 performance when compared to ReRe. Finally, Section VI draws conclusions while also discussing further research implications and work directions.

II. RELATED WORK

Despite the recent proliferation of telemetry in networking, there is only a handful of research in the topic of anomaly

[1] The authors are with the Department of Networked Systems and Services, Budapest University of Technology and Economics, Budapest, Hungary. Karoly Farkas is also with NETvisor Ltd., Budapest, Hungary.

E-mail: vajdadaniellaszlo@edu.bme.hu, {apekar, farkask}@hit.bme.hu

DOI: 10.36244/ICJ.2021.1.5

detection. Putina *et al.* [4] at Cisco developed a streaming telemetry-based anomaly detection engine for BGP anomalies. However, it uses a legacy clustering algorithm called DenStream [5] that has limited performance.

Other works with limited applicability to anomalies and computer networks include [6], [7]. Ye *et al.* [6] use a statistical approach to detect zero-day attacks and malicious intent. The paper claims that machine learning techniques miss the bigger picture of network behaviour. Kaiafas *et al.* [7] use multiple unsupervised machine learning algorithms in an ensemble to identify fraudulent private exchange phone calls, yet their approach only works on off-line data.

More generic real-time anomaly detection approaches are AnomalyDetectionTs (ADT) and AnomalyDetectionVec (ADV) developed by Twitter [8]. ADT works on time-series data, while ADV is designed for vectors without timestamp information. These algorithms employ statistical-based approaches, therefore require massive data points. This results in poor applicability to streaming time-series data.

In spite of the limited number of directly applicable research in the area of network traffic anomaly detection, the trend of using machine learning solutions for time-series analysis is emerging. Syal *et al.* [9] propose an SVM (Support Vector Machine) [10] based supervised learning method to detect abnormally slow network transfers in real time, focusing on TCP flows. Pilinszki-Nagy *et al.* [11] analyze and investigate a special type of artificial neural network called HTM (Hierarchical Temporal Memory) [12] to model and predict sequential data, including time series. They use off-line, unsupervised learning to train the HTM model. Unfortunately, they discuss only the prediction part of anomaly detection and do not deal with the decision logic to indicate anomalous data.

A few recent works have proposed and investigated the applicability of deep learning, especially LSTM [2] and AutoEncoder [13] based methods for anomaly detection on time-series data. Zhou *et al.* [14] propose an off-line, variational LSTM learning model based on reconstructed feature representation to detect anomaly on industrial big data. Lazaris *et al.* [15] aim to predict fine-grained network traffic using an LSTM model in the domain of SDN (Software-Defined Networking). Soheil *et al.* [16] investigate aspects of network traffic forecasting using real-world data streams and two machine learning models, namely LSTM and SARIMA (Sequential Auto-Regressive Integrated Moving Average) [17], in a supervised manner. Unfortunately, they focus only on traffic prediction and do not deal with the anomaly detection logic. Gjorgiev *et al.* [18] propose several off-line deep learning architectures based on variational AutoEncoders [19] for detecting cyber-attacks on water distribution system. They calculate the Mahalanobis distance [20] instead of the traditional mean square error in the objective function to get better performance. RE-ADTS [21] is an unsupervised anomaly detection approach using a deep AutoEncoder model that can be applied either to batch or real-time anomaly detection. It seems to perform evenly well on time-series datasets from various domains. STAD [22] is a dynamic on-line data mining technique; an automated framework to detect cellular network anomalies. It uses a combination of machine learning methods, such as OC-SVM (One-class SVM) [23], SVR (Support Vector Regression) [24], and LSTM. Similarly, Said Elsayed *et al.* [25] propose a hyper approach based on LSTM AutoEncoder and OC-SVM to detect anomalies based attacks in SDN environments. However, it is not clear how this method can be applied in real time. LSTM-FUZZY [26] is a system to detect and mitigate different attacks in SDN environments. In this system, LSTM is used for network traffic forecasting since fuzzy logic is applied for anomaly detection.

The Greenhouse [27] algorithm fits our design goals best, combining state-of-the-art machine learning and data management methods for anomaly prediction over immense volumes of time-series data. The algorithm must be trained on normal data but does not require labeled anomalies, which technique is referred to as 'zero-positive' or semi-supervised learning. Greenhouse uses a look-back, predict-forward approach to detect anomalies. It employs an LSTM model to first predict new values based on old ones, then compares the prediction to the actual data point. RePAD [3] is an improvement of Greenhouse that eliminates the need for normal training data. ReRe [1] is an upgrade of RePAD that aims to mitigate false positive detections. As ReRe appeared to be the most suitable approach for our work, we base our approach for streaming telemetry anomaly detection on it. A detailed description of RePAD and ReRe is provided in Section III.

In summary, there is only a handful of existing works aimed at anomaly detection in streaming telemetry data. Generic real-time anomaly detectors, however, have gone through a major improvement in the last decade. Despite recent advances, they still face several challenges. For example, anomaly detection in an unsupervised manner is limited, while other approaches need domain knowledge to set critical parameters or cannot adapt to changing behaviours.

III. Background and Motivation

A. RePAD

RePAD [3] developed by Lee *et al.* is a cutting-edge LSTM-based algorithm designed for time-series anomaly detection. The authors claim that RePAD can detect anomalies proactively in real time, without domain knowledge.

RePAD uses short-term historical data points to predict the upcoming value; then, it compares this prediction with the real value to determine if an anomaly is likely to happen in the near future. RePAD can adjust detection thresholds dynamically, making it well-suited to tolerate minor pattern changes as well. Its fast convergence (i.e., it can detect anomalies soon after start) and unsupervised training (i.e., it does not require a labelled dataset) set it apart from previous approaches.

A key part of RePAD is the LSTM model used for data prediction. LSTMs are a type of recurrent artificial neural networks (RNNs). LSTMs form layers of neurons. One neuron can have many input and output connections. We assign weights to these connections that impact the propagation of information through the network. Training consists of multiple forward and backward passes (one round is called an 'epoch') during which these weights and other parameters are tuned to reflect certain patterns. Essentially, if an LSTM model has

a complicated structure or the training data is large, training time increases significantly, limiting real-time use. That is why the LSTM model used in RePAD has only one hidden layer with ten hidden units. Additionally, a fast learning speed is guaranteed by a learning rate of 0.15. The number of epochs is a key factor in determining the precision and speed of a neural network model. RePAD employs Early Stopping [28] to choose the number of epochs dynamically, which aims to prevent overfitting and underfitting. A detailed discussion of LSTMs is out of the scope of this work. For a better comprehension of the topic, we refer the reader to [2], [29], [30].

RePAD is based on a so-called 'look back, predict forward' approach. It takes the previous b data points (b is the look-back parameter) and uses them to predict the next f data points (f is the predict-forward parameter). RePAD uses $f = 1$. The operation of RePAD is presented in Algorithm 1 (M, M' denote the LSTM models). The algorithm uses four equations:

$$AARE_t = \frac{1}{t-b+1} \cdot \sum_{y=b}^{t} \frac{|v_y - \widehat{v_y}|}{v_y} \tag{1}$$

where

$AARE_x$ = Average Absolute Relative Error at timestep x;
t = current timestep, starts from $t = 0$;
b = look-back parameter;
v_x = data point at timestep x;
$\widehat{v_x}$ = predicted data point for timestep x.

AARE is a well-known measure for determining the accuracy of prediction. A low AARE value indicates that the forecast value is close to the observed value [3].

$$\mu_{AARE,t} = \frac{1}{t-b+1} \cdot \sum_{y=b}^{t} AARE_y \tag{2}$$

where

$\mu_{AARE,x}$ = the average of $AARE_y$ values at timestep x.

$$\sigma_{AARE,t} = \sqrt{\frac{\sum_{y=b}^{t}(AARE_y - \mu_{AARE,t})^2}{t-b+1}} \tag{3}$$

where

$\sigma_{AARE,x}$ = the standard deviation of $AARE_y$ values at timestep x.

$$thd_t = \mu_{AARE,t} + 3 \cdot \sigma_{AARE,t} \tag{4}$$

where

thd_x = threshold value at timestep x.

As seen above in Equation (4), the thd_x values are determined using the Three-Sigma Rule [31], which is commonly used for anomaly detection threshold calculation.

From Algorithm 1, it is evident that RePAD requires only a short preparation for anomaly detection. However, a major flaw of this approach is the relatively high number of false positives it yields. Lee *et al.* strive to overcome this shortcoming by ReRe, a refinement of the RePAD algorithm.

Algorithm 1 RePAD

```
t = 0
while (t++) do
    Collect data point v_t
    if t < b - 1 then
        ▷ Step 1: collect b − 1 points passively
    else if t == b - 1 then
        ▷ Step 2: first training and prediction
        Train the LSTM model M using the first b data points: v_0, ..., v_t = v_{b−1}
        Predict the next data point v_{t+1}^ = v_b^ using M
    else if b - 1 < t < 2b - 1 then
        ▷ Step 3: preparing for detection
        Calculate AARE_t using Equation (1)
        Train the LSTM model M using the previous b data points: v_{t−b+1}, ..., v_t
        Predict the next data point v_{t+1}^ using M
    else if t ≥ 2b - 1 then
        ▷ Step 4: anomaly detection
        Calculate AARE_t using Equation (1)
        Calculate thd_t using Equations (2), (3) and (4)
        if AARE_t ≤ thd_t then
            ▷ v_t is similar to previous points, NO ANOMALY
            Predict the next data point v_{t+1}^ using M
        else
            ▷ Pattern change or anomaly
            Train the LSTM model M' using the previous b data points
            Predict the current point v_t^ again using M'
            Recalculate AARE_t using Equation (1)
            if AARE_t > thd_t then
                Signal an ANOMALY
                Discard M' and use M to predict the next data point v_{t+1}^
            else
                Signal a PATTERN CHANGE
                Replace M with M' to predict new points accurately
                Use M to predict the next data point v_{t+1}^
```

B. ReRe

ReRe employs two LSTM models that provide two levels of detection sensitivity. They are deployed in two detectors:

Detector 1: operates the same way as RePAD described in Section III-A. It first acquires t and v_t values and then produces one of the three output signals, namely 'normal,' 'pattern change,' and 'anomaly.' Fundamentally, it stores and calculates $AARE_t$ and thd_t using RePAD methods and Equations (1) to (4). When it detects a pattern change, it retrains its own LSTM model, M_1 that is used for data value prediction.

Detector 2: uses the same algorithm structure as RePAD while also acquiring t and v_t in the same way. However, the input values for $AARE_t$ and thd_t are calculated variously. Specifically, Detector 2 uses its own M_2 LSTM model to predict the values. M_2 is structurally identical to M_1, but it

produces various predictions as it uses a different equation for calculating the thd_t threshold. Indeed, Detector 2 uses only $AARE_x$ for this calculation, where v_x is considered 'normal' (no 'pattern change' or 'anomaly') by itself. Broadly speaking, this difference results in giving ReRe the capability to suppress anomalies detected by Detector 1 (RePAD). Finally, an anomaly or pattern change is only detected and signalled if both detectors return the same detection for a given timestep.

C. Limitations

The current implementation of ReRe stores all the computed values and derived information since the beginning of its operation. Consequently, there is no upper bound on the memory requirements of ReRe. Moreover, the performance of ReRe significantly depends on the setting of the look-back parameter b, which has been pointed out and also discussed by the authors of RePAD / ReRe in their recent study [32]. Additionally, ReRe can only detect anomalies when both detectors have an $AARE_t$ value higher than thd_t, and both detectors indicate no pattern change. That means, typically, $AARE_y$ values are lower than thd_y.

In both RePAD and ReRe, the $AARE_t$ and thd_t values are computed using Equation (1) and Equation (4). Both values include a sum whose starting value is fixed (i.e., timestep $y = b$) and ends at the current timestep ($y = t$). All terms in these sums have an equal weight of 1, which means that all terms have an equal priority in determining current $AARE_t$ and thd_t values. As a result, the speed of anomaly detection is adversely impacted.

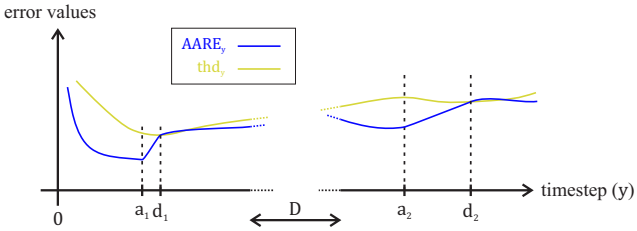


Fig. 1: The slope of AARE depends on the current timestep.

This issue is illustrated in Figure 1 on the left side of the graph. At timestep a_1, there is an anomaly in the dataset. $AARE_y$ values start to rise rapidly as new absolute relative error terms are high, and there are only a few error terms to be averaged. At timestep d_1, the $AARE_y$ curve crosses the thd_y curve and an anomaly is detected. However, when a considerable time has passed (D is at least a few thousand timesteps), a high count of terms in $AARE_t$ calculation is to be averaged. Consequently, when an anomaly occurs at timestep a_2, the $AARE_y$ curve starts to rise slowly, as averages increase slower with more terms (the impact of one new high term is proportionally less). As the starting timestep for the average is always $y = b$, the steepness of the slope decreases as y increases. That is why the anomaly at a_2 is only detected at d_2.

Consider the algorithm runs in a real-world production environment, uninterrupted. Due to the flaw mentioned above, it becomes very likely that some anomalies are unnoticed past a certain timestep. Indeed, in such a scenario, the $AARE_y$ curve may never reach the thd_y curve since it normalises before normal data points arrive. These limitations of ReRe have motivated us to introduce our enhancements.

IV. ALTER-RE^2

Streaming network telemetry anomaly detection is not obvious. While there has been some attempts to detect anomalies on time-series data, as described above, they fall short when applied in a real-world production environment. To fill this important gap in the field, we enhanced ReRe, a real-time anomaly detection algorithm that is, to the best of our knowledge, a cutting-edge technique. Alter-Re^2, our attempt towards machine learning-based anomaly detection on time-series data, comprises two enhancements — ageing and sliding window. In what follows, we describe these enhancements.

A. Ageing

We designed ageing so that it places greater emphasis on a few previous data points instead of averaging them with the same weight. Our ultimate goal was to decrease the weight of a given data point as time elapses. We achieve this by introducing an extra ageing coefficient C_y in Equation (1), which yields the following formula:

$$AARE_{t,ageing} = \frac{1}{t-b+1} \cdot \sum_{y=b}^{t} C_y \cdot \frac{|v_y - \widehat{v_y}|}{v_y} \tag{5}$$

The C_y coefficient is calculated using the following equation:

$$C_y = \left(\frac{y-W}{t-W} \right)^{AP} \tag{6}$$

where

y = timestep running variable in the sum;
W = starting timestep of the window (see details in Section IV-B);
AP = age power;
C_y = ageing coefficient.

B. Sliding Window

ReRe stores all data from the point in time they were generated. We mitigate the consequence of this issue by storing only the previous value of the average error terms and the number of data points it was calculated from. This recursive method is formally expressed as follows:

$$AARE_t = \frac{1}{t-b+1} \cdot \left(AARE_{t-1} \cdot (t-b) + \frac{|v_t - \widehat{v_t}|}{v_t} \right) \tag{7}$$

To calculate the threshold value thd_t, we need to determine the standard deviation of the $AARE_y$ values. However, $\sigma_{AARE,t}$ cannot be expressed only using values from the previous timestep $t-1$ and the number of timesteps due to the

changing $\mu_{AARE,y}$ values in every timestep. Therefore, every $AARE_y$ value must be stored from the beginning.

We avoid storing every data point by implementing a sliding window with only one parameter — WINDOW_SIZE or simply WS. The starting timestep W is calculated via the following formula:

$$\begin{cases} W = t - WS + 1 & \text{if } t - WS + 1 > b \\ W = b & \text{if } t - WS + 1 \leq b \end{cases} \tag{8}$$

where

W = beginning timestep of the window.

As a result, the data points before the beginning of the window are discarded. The equations known from ReRe (see Section III) are modified according to the following formulas:

$$AARE_t = \frac{1}{t - W + 1} \cdot \sum_{y=W}^{t} C_y \cdot \frac{|v_y - \widehat{v_y}|}{v_y} \tag{9}$$

$$\mu_{AARE,t} = \frac{1}{t - W + 1} \cdot \sum_{y=W}^{t} AARE_y \tag{10}$$

$$\sigma_{AARE,t} = \sqrt{\frac{\sum_{y=W}^{t} (AARE_y - \mu_{AARE,t})^2}{t - W + 1}} \tag{11}$$

$$thd_t = \mu_{AARE,t} + 3 \cdot \sigma_{AARE,t} \tag{12}$$

C. Effects of the Enhancements

The effects of ageing and sliding window can perhaps be best comprehended via Figure 2. The impact of ageing becomes evident, as the age power variable (AP) in Equation (6) determines the pace of ageing, i.e., the extent to which previous data points should impact the operation of ageing. If $AP = 1$, there is linear ageing. Negative numbers are not recommended, as they result in inverse ageing.

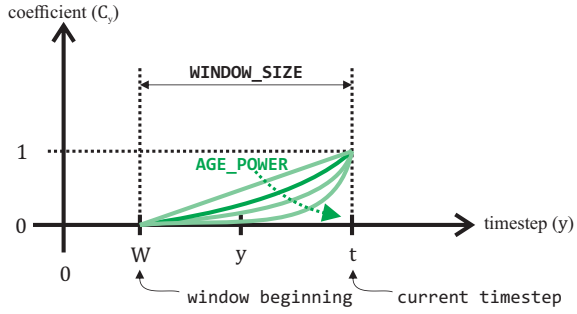
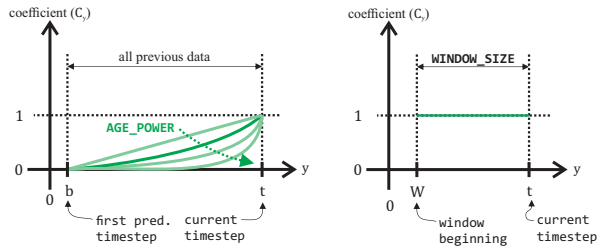


Fig. 2: The operational principles of ageing and sliding window.

Obviously, AP becomes an additional hyperparameter of the algorithm. Equation (6) always produces a number between 0 and 1 for C_y if $y \in [W, t]$. This way, the last few data points will remain approximately the same, while the ones closer to the start will be scaled down. Ageing gives the algorithm the capability to address slow or no reaction to certain anomalies (see Section III-C) since new high error terms influence $AARE_y$ values more than older smaller ones.

The implemented sliding window has a significant benefit, as shown in Figure 2. The values of thd_y are calculated using $AARE_y$ values only from within the window. Therefore, the detection threshold adjusts faster and more precisely, and high $AARE_y$ values from a few thousand past timesteps do not distort the performance until the very end of the operation.

Figure 2 also demonstrates the dependence between ageing and sliding window. Equation (6) produces values of 0 at the beginning of the window (timestep W) and produces values of 1 at the current timestep t. If ageing is deactivated, all values of C_y are set to 1. On the other hand, if the sliding window is disabled, the window size parameter WS is set to one more than the current timestep t. This way, Equation (8) always chooses the timestep b as the beginning of the window because the first predicted value is produced then. In this case, ageing is implemented on the whole previous dataset, i.e., from b to the current timestep t. The disabled sliding window and disabled ageing effects are visualized in Figure 3.



(a) Disabled sliding window (b) Disabled ageing

Fig. 3: Effects of disabled sliding window and ageing.

V. RESULTS

The introduced enhancements in the previous section serve the purpose of real-time anomaly detection on time-series data. In what follows, we assess the performance of Alter-Re^2 when compared to ReRe.

A. Preliminaries

To date, there is no publicly available implementation (i.e. source code) of the RePAD and ReRe algorithms. Therefore, we have implemented them in Python. However, it is worth mentioning that our way of implementing LSTM, including the functions used for training/testing, might differ from the method implemented by Lee *et al.* This speaks to the continuous research challenge of reproducibility.

Initially, we configured ReRe with the parameters used in [1]. Specifically, we defined one hidden LSTM layer and set the number of its neurons to 10. However, such a configuration has been shown to be insufficient, as the neural network model made unreasonable predictions likely due to the inability to learn data patterns with enough complexity at the beginning. After rigorous experimentation with various configuration settings and manual hyperparameter optimization, we observed that 30 neurons in the one hidden layer with 30 epochs

yielded the best results. On the one hand, increasing these numbers further significantly increased the training time. On the other hand, decreasing these values degraded the prediction performance.

We also conducted experiments to determine the look-back parameter (b). Lee *et al.* used a relatively low parameter, i.e., $b = 3$. Interestingly, in our assessment, such a value of b could not produce satisfying results. Upon experimenting with the effect of all parameters on the overall performance, we concluded that setting b to 30 resulted in the best performance. A higher or lower number has occasionally manifested in a constant offset between the original and predicted data points. The predict-forward parameter f was set to a constant value of 1, as discussed in Section III-A (ReRe predicts only the next datapoint $\widehat{v_{t+1}}$).

Our finding regarding the WINDOW_SIZE (*WS*) parameter is that a too-small value results in an unstable operation since the number of data points are low to learn long-term dependencies. The upper bound on *WS* is the storage capacity allocated for the algorithm. CPU and time constraints might also have to be considered as the volume of data points impacts the computational complexity and timeliness. Eventually, we deduced experimentally that the setting of *WS* to 1000 was optimal with the datasets we performed our tests on.

Last, we also observed that the most promising results could be obtained by setting the AGE_POWER (*AP*) parameter to 2. This ends up in a quadratic ageing equation. If *AP* is significantly lower, old data points have a strong influence on the current $AARE_t$ value, which is undesirable. On the contrary, if *AP* is too high, only the last data points affect the performance, resulting in unstable operation. In our experiments, we eventually set *AP* to 2. Table I summarizes the parameter settings we have used in our experiments.

TABLE I: Summary of the used parameter settings.

Parameter	ReRe	Alter-Re^2
neurons	30	30
epochs	30	30
b	30	30
WS	-	1000
AP	-	2.0

B. ReRe vs. Alter-Re^2

Our experiments have used the Numenta Anomaly Benchmark (NAB) [33] datasets. NAB is destined for evaluating streaming and real-time applications of anomaly detection algorithms. The datasets comprise real-world and artificial data containing human-labelled anomalous behaviour periods. The majority of the data is real-world from various sources such as AWS server metrics, Twitter volume, advertisement clicking metrics, and traffic data. Data are ordered, timestamped, and single-valued metrics. The timestamps of the anomaly labels are provided as a separate file.

We examined the efficacy of Alter-Re^2 compared to ReRe using various datasets (see Table II for details on the used NAB datasets containing anomalies related to CPU utilization, request latency, and request count, among others). In this work, we detail the observations achieved with the 'ec2_cpu_utilization_ac20cd' dataset, which contains CPU utilization percentages collected from AWS servers using the CloudWatch monitoring tool [34].

Figures 4 and 5 depict the detection results achieved by ReRe and Alter-Re^2, respectively. The figures have the same layout. They consist of three graphs that show the details of algorithm operation. The top graphs show the original data points (v_y) with green, and the LSTM-predicted data points ($\widehat{v_y}$) using red. The original and predicted values have been scaled down to the $[0, 1]$ interval (a requirement of LSTM). In the middle graph, we plot the absolute average relative error ($AARE_y$) using blue and the thresholds (thd_y) with yellow. The bottom figures display detections made by ReRe. Furthermore, anomaly detection is drawn with purple, while the pattern changes have a turquoise colour. The configuration of WINDOW_SIZE (WS) and AGE_POWER (AP) parameters used in our experiment are shown in the figure captions. In the captions, we also indicate the timesteps of the anomaly labels.

From Figures 4 and 5, we find that the first anomaly is detected by both algorithms (see the bottom graphs). When the $AARE$ curve in the middle graph rises above the thd curve, the algorithms determine that an anomaly has occurred. At around timestep 600, the original data rises again. Both ReRe and Alter-Re^2 signal a pattern change as a result and retrain the LSTM models accordingly.

However, the shortcoming of ReRe comes to the surface soon. As shown in the middle graph of Figure 4, after detecting the first anomaly, thd values rise significantly as they are calculated using the average and standard deviation of $AARE$ values. Additionally, as the second anomaly comes at timestep 3576, many data points have already been collected, and the issue (the slope of the $AARE$ curve is much lower) arises. In consequence, ReRe fails to detect the anomaly.

Alter-Re^2, on the other hand, reliably detects the second anomaly, as shown in Figure 5. The thd curve is being 'reset' between timesteps 1500 and 2500 caused by the sliding window. This leads to increased detection accuracy. The slope of the $AARE$ curve increases significantly due to the implemented ageing. Note that because of the limited range of the plots that can be expressed, there is a trade-off in these figures between the visibility of the individual points and the expression of pattern changes/anomalies.

C. Discussion

The above-discussed experiment had also been performed using nine other NAB datasets. The used datasets contain anomalies related to CPU utilization, request latency, and request count, among others.

The properties of these datasets are highlighted in Table II. We have used the same parameter settings, collected in Table I, in all of the experiments. Interestingly, these LSTM based methods provide, on each occasion, slightly deviating predictions even in the case of running the same test with the same settings on the same dataset. This deviation can be explained by the inherent randomness in the LSTM

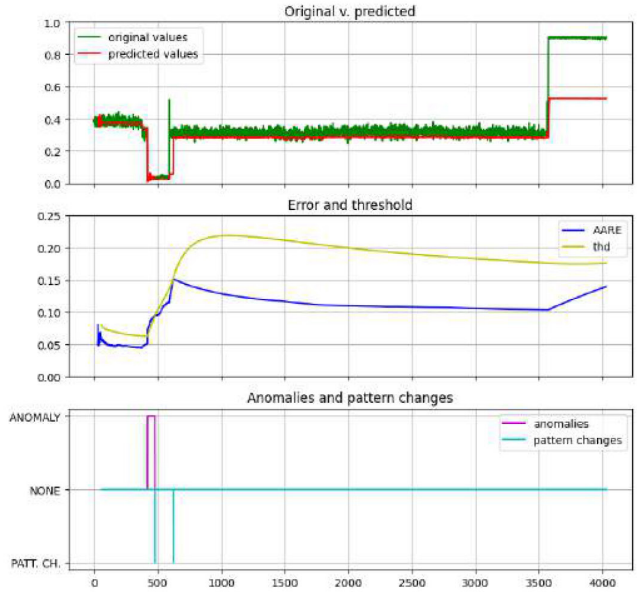


Fig. 4: ReRe output for ec2_cpu_utilization_ac20cd.csv (no ageing, no window). Labeled anomalies at timesteps 421, 3576.

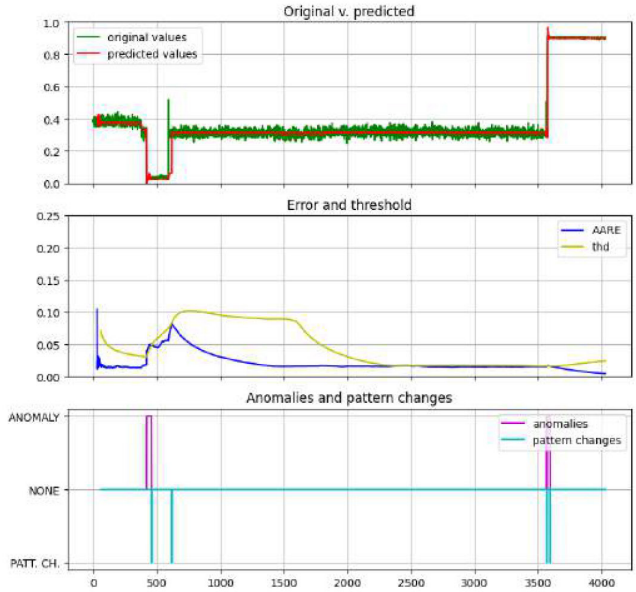


Fig. 5: Alter-Re^2 output for ec2_cpu_utilization_ac20cd.csv (AP: 2, WS: 1000). Labeled anomalies at timesteps 421, 3576.

model's implementation, resulting in different initial weights and parameters. Unfortunately, we were unable to compare our implementation with the original one because there was no publicly available source code of the ReRe algorithm as the date of carrying out this work. However, the apparent difference of prediction visible in Figures 4 and 5 is not due to this phenomenon. Instead, it results from the difference in the timesteps where the algorithms triggered an LSTM retrain. To recap, this happens when ReRe or Alter-Re^2 detects a pattern change. From this point on, they use their new LSTM model that, in this case, are trained on slightly different data resulting in slightly different predictions. Nonetheless, reducing the offset between predicted and original values constitutes a high priority task on our list of future work.

The results of the ten experiments mentioned above are presented in Table III. For the traditional performance metrics, like Precision, Recall and F-score, to be calculated, the elements of the confusion matrix have to be populated. These are the True Positives (TP), False Positives (FP), True Negatives (TN) and False Negatives (FN). Other special metrics, e.g., LSTM retrain ratio, can also provide valuable pivots in performance evaluation. We selected the metrics to be used in this work and implemented their calculation techniques according to the study of Lee *et al.* [32].

Mapping the ground truth anomalies to the detected ones relies on anomaly windows around the timesteps of the ground truth labels. The only parameter K determines the size of this anomaly window. If the timestep for a ground truth anomaly is denoted by y_{GT}, the anomaly window starts at $y_{GT} - K$ and ends at $y_{GT} + K$. We set $K = 7$, as per the recommendations in [35] for minute-based time series.

Anomaly signals have also been processed. More specifically, all detections longer than one timestep are remapped to the timestep they started at. After creating anomaly windows and remapping signals, the algorithm calculates the confusion matrix elements in the following way. All signals outside the anomaly window are considered as false positive. Only the first signal inside each window is regarded as a true positive. All others inside the same window are discarded. If no signals are present in the whole window, the number of false negatives is incremented by one. True negatives are the remaining number of timesteps to ensure the length of the dataset is equal to the sum of the elements in the confusion matrix.

The values of the selected performance metrics of our experiments using the ten datasets listed in Table II are summarized in Table III. In total, using the evaluation method detailed above, Alter-Re^2 was able to identify seven anomalies, while ReRe detects only one correctly. Simultaneously, the number of false positives (not depicted in the table) also increased by introducing Alter-Re^2. Upon closer examination of the experiment results, we found that this phenomenon comes from the oscillations in the anomaly signals (i.e., an alternation between signal and no signal) just outside the anomaly window and rarely from really false detections. Comparing the number of true negatives (not depicted in the table) between the two algorithms has little significance, as values differ only by fractions of percentages. On the other hand, false negatives (not depicted in the table) are reduced when contrasting Alter-Re^2 with ReRe.

In Table III, we have also depicted the traditional metrics generally used for performance analysis. Precision is the number of correctly identified anomalies divided by the number of all anomaly signals. It is a meaningful metric for anomaly detection, and Alter-Re^2 outperforms ReRe almost all the time, as shown in the table (the 'nan' value means that division by zero would be required to calculate the metric). The next

TABLE II: Datasets used for evaluation.

No.	Dataset	First record time	Last record time	No. of data points	No. of anomalies	Sampling rate
01	ec2_cpu_utilization_5f5533.csv	2014-02-14 14:27:00	2014-02-28 14:22:00	4032	2	5 min
02	ec2_cpu_utilization_825cc2.csv	2014-04-10 00:04:00	2014-04-24 00:09:00	4032	2	5 min
03	ec2_cpu_utilization_ac20cd.csv	2014-04-02 14:29:00	2014-04-16 14:49:00	4032	2	5 min
04	ec2_network_in_257a54.csv	2014-04-10 00:04:00	2014-04-24 00:09:00	4032	1	5 min
05	ec2_request_latency_system_failure.csv	2014-03-07 03:41:00	2014-03-21 03:41:00	4032	3	5 min
06	elb_request_count_8c0756.csv	2014-04-10 00:04:00	2014-04-24 00:39:00	4032	2	5 min
07	grok_asg_anomaly.csv	2014-01-16 00:00:00	2014-02-01 01:00:00	4621	3	5 min
08	iio_us-east-1_i-a2eb1cd9_NetworkIn.csv	2013-10-09 16:25:00	2013-10-13 23:55:00	1243	2	5 min
09	rds_cpu_utilization_cc0c53.csv	2014-02-14 14:30:00	2014-02-28 14:30:00	4032	2	5 min
10	rds_cpu_utilization_e47b3b.csv	2014-04-10 00:02:00	2014-04-23 23:57:00	4032	2	5 min

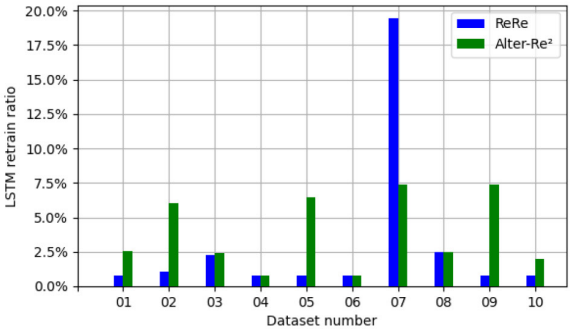


Fig. 6: LSTM retrain ratio.

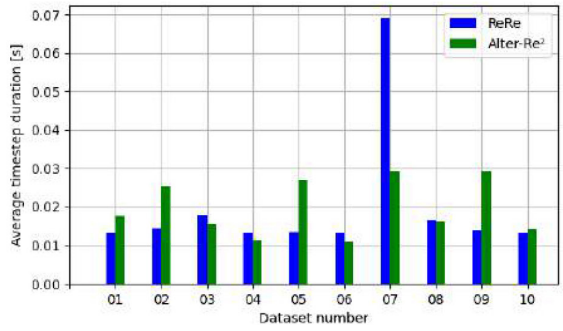


Fig. 7: Average timestep duration.

depicted metric is Recall, which denotes the ratio of the number of true positives to the number of all ground truth labels. This is also an essential indicator in anomaly detection, and Alter-Re^2 overperforms ReRe by also this regard in most cases. Finally, we depicted the F-score, which is the harmonic mean of Precision and Recall. When available, Alter-Re^2 is superior to ReRe considering this metric, as well.

We believe that a more appropriate setting of the anomaly window (instead of $K = 7$) can eliminate the above-mentioned phenomenon of indicating detection signals outside this window as false positives. Nonetheless, our experiment results confirm the significance and superiority of our Alter-Re^2 algorithm over ReRe since the performance metrics used in the comparison were calculated in the same way. In order to empirically compare the number of anomalies detected by ReRe and Alter-Re^2, we manually adjusted the number of true positives (we increased the number of true positives by the number of signals just outside of the given anomaly window, that were clearly resulted from the given ground truth anomaly). We found that even so, ReRe detected only three anomalies compared to the ten ones identified by Alter-Re^2, resulting in a more than three-fold increase.

The other special metrics we have also used in our evaluations were: a) LSTM retrain ratio (the number of timesteps with an LSTM retrain divided by the total number of timesteps); b) Average timestep duration (total time divided by the number of timesteps); and c) Preparation period (number of timesteps after which ReRe or Alter-Re^2 can detect anomalies). As depicted in Figure 6, the LSTM retrain ratio increased significantly in the case of Alter-Re^2 in most of the experiments. This is due to the larger number of signals (anomalies and pattern changes) raised by the algorithm. Considering the Average timestep duration, the two algorithms show similar performance in most of the experiments as depicted in Figure 7. Finally, the Preparation period was the same for both algorithms on all datasets, as it depends solely on the b parameter. In our experiments, it was $2b - 1 = 2 \cdot 30 - 1 = 59$. This can be interpreted as both algorithms can detect anomalies soon, 59 timesteps after being turned on.

Nevertheless, there are some limitations of LSTM based approaches. The predictive power of LSTM networks is relatively low at the beginning. Essentially, LSTM was designed to learn long term dependencies, *i.e.*, it remembers the information for long periods. A well-performing LSTM on very long sequences of non-stationary data commensurate with an increase in required network capacity and training. In response, if a pattern change is detected in our approach, the LSTM is retrained using the previous b data points. However, in extreme scenarios where anomalies are relatively frequent, the LSTM network might constantly have low predicting power.

Transfer learning appears suitable to address this limitation. Broadly speaking, in transfer learning, a model developed for a task is reused as the starting point for a model in the next task. By applying this logic in LSTM time series, the weights obtained from the first task could be used in the next task [36]. Consequently, in cases where anomalies are frequent, the low predictive performance of LSTM could presumably be overcome. However, further research is needed to study the applicability of transfer learning (and other

TABLE III: Evaluation results (traditional metrics; R: ReRe, A: Alter-Re^2).

Dataset no.	No. of detected anomalies		Precision		Recall		F-score	
	R	A	R	A	R	A	R	A
1	0	**1**	nan	**0.3333**	0.0	**0.5**	nan	0.4
2	0	**1**	0.0	**0.3333**	0.0	**0.5**	nan	0.4
3	1	**2**	**1.0**	0.6667	0.5	**1.0**	0.6667	**0.8**
4	0	0	nan	nan	0.0	0.0	nan	nan
5	0	**1**	nan	**0.3333**	0.0	**0.3333**	nan	**0.3333**
6	0	0	nan	nan	0.0	0.0	nan	nan
7	0	**1**	0.0	**0.2**	0.0	**0.3333**	nan	**0.25**
8	0	0	nan	nan	0.0	0.0	nan	nan
9	0	**1**	nan	**0.3333**	0.0	**0.5**	nan	**0.4**
10	0	0	nan	0.0	0.0	0.0	nan	nan

advanced approaches, such as online learning) in time-series anomaly detection.

Furthermore, another question yet to be examined is the efficiency of *AARE*. When an outlier value is detected in our prototype implementation, its error is included in the calculation. This can result in a significant increase of *AARE*, implying that anomalies might remain undetected for the period when *AARE* is high. Undoubtedly, further study is needed to assess the practicability of *AARE* in anomaly detection accuracy. A naive solution of this challenge considered in future work could be excluding the extreme values causing anomalies from the *AARE* calculation and focus only on the values considered normal instead.

VI. Conclusions

Real-time anomaly detection in time-series data is an emerging area with approaches mostly based on neural networks, while there is a noticeable increase in the use of LSTMs. Despite recent advances in anomaly detection, classifying abnormal behaviour in time-series data is still challenging.

In this paper, we introduced Alter-Re^2, an enhancement of the cutting-edge ReRe algorithm. Specifically, we discuss our sliding window and ageing techniques. The former is aimed at limiting memory and CPU overheads. The latter serves the purpose of ageing the data points that are used for calculating error terms. Ageing addresses the issue of slow anomaly detection times. Furthermore, it is also destined to observe abnormal behaviour that is unpredicted by prior works.

We rigorously evaluated Alter-Re^2 and observed promising results. Our approach achieved significantly better performance when compared to ReRe. Specifically, it can detect three times more anomalies. Furthermore, Alter-Re^2 can detect such anomalies too that ReRe falls short.

As future work, we plan to evaluate the applicability and usefulness of several other concepts, such as offset compensation, adaptive threshold sigma-coefficient, automatic setting of some hyperparameters, introducing real-time normalization, adapting the algorithm to support multivariate data, and incorporate them into our algorithm. We also plan to study the applicability of transfer learning in time series anomaly detection.

As time-series data streams are now an integral part of almost every field of technology, real-time anomaly detection on these data is a vital tool that deserves more attention. We believe that our contribution is valuable in facilitating the development of relevant techniques, yet we argue our approach has immediate real-world applicability.

Acknowledgment

The research reported in this paper and carried out at the BME has been supported by the NRDI Fund based on the charter of bolster issued by the NRDI Office under the auspices of the Ministry for Innovation and Technology.

References

[1] M.-C. Lee, J.-C. Lin, and E. G. Gran, "ReRe: A Lightweight Real-Time Ready-to-Go Anomaly Detection Approach for Time Series," in *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*, 2020, pp. 322–327.

[2] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[3] M.-C. Lee, J.-C. Lin, and E. G. Gran, "RePAD: Real-Time Proactive Anomaly Detection for Time Series," in *Proceedings of the Advanced Information Networking and Applications*, L. Barolli, F. Amato, F. Moscato, T. Enokido, and M. Takizawa, Eds., Cham: Springer International Publishing, 2020, pp. 1291–1302.

[4] A. Putina et al., "Telemetry-Based Stream-Learning of BGP Anomalies," in *Proceedings of the 2018 Workshop on Big Data Analytics and Machine Learning for Data Communication Networks*, ser. Big-DAMA '18, Budapest, Hungary: Association for Computing Machinery, 2018, pp. 15–20.

[5] F. Cao, M. Estert, W. Qian, and A. Zhou, "Density-Based Clustering over an Evolving Data Stream with Noise," in *Proceedings of the 2006 SIAM International Conference on Data Mining*, pp. 328–339.

[6] N. Ye, D. Montgomery, K. Mills, and M. Carson, "Multivariate metrics of normal and anomalous network behaviors," in *Proceedings of the 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, IEEE, 2019, pp. 55–58.

[7] G. Kaiafas et al., "An Experimental Analysis of Fraud Detection Methods in Enterprise Telecommunication Data using Unsupervised Outlier Ensembles," in *Proceedings of the 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, IEEE, 2019, pp. 37–42.

[8] Twitter. (). "Twitter/anomalydetection [online code repository]," [Online]. Available: https://github.com/twitter/AnomalyDetection. Accessed: 2021-01- 13.

[9] A. Syal, A. Lazar, J. Kim, A. Sim, and K. Wu, "Automatic Detection of Network Traffic Anomalies and Changes," in *Proceedings of the ACM Workshop on Systems and Network Telemetry and Analytics*, ser. SNTA '19, Phoenix, AZ, USA: Association for Computing Machinery, 2019, pp. 3–10.

[10] C. Cortes and V. N. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[11] C. Pilinszki-Nagy and B. Gyires-Tóth, "Performance Analysis of Sparse Matrix Representation in Hierarchical Temporal Memory for Sequence Modeling," *Infocommunications Journal*, vol. 12, no. 2, pp. 41–49, 2020.

[12] J. Hawkins, S. Ahmad, S. Purdy, and A. Lavin, "Biological and Machine Intelligence (BAMI)," Initial online release 0.4, 2016.

[13] M. A. Kramer, "Nonlinear principal component analysis using autoassociative neural networks," *AIChE Journal*, vol. 37, no. 2, pp. 233–243, 1991.

[14] 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.

[15] A. Lazaris and V. K. Prasanna, "An LSTM framework for modeling network traffic," in *Proceedings of the 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, IEEE, 2019, pp. 19–24.

[16] J. Soheil et al., "On the Practicality of Learning Models for Network Telemetry," *Proceedings of Network Traffic Measurement and Analysis Conference*.

[17] G. Box and G. Jenkins, *Time Series Analysis: Forecasting and Controls rev*. Oakland California: Holden-Day, 1976.

[18] L. Gjorgiev and S. Gievska, "Time Series Anomaly Detection with Variational Autoencoder Using Mahalanobis Distance," in *Proceedings of the ICT Innovations 2020. Machine Learning and Applications*, V. Dimitrova and I. Dimitrovski, Eds., Cham: Springer International Publishing, 2020, pp. 42–55.

[19] P. Diederik and M. Welling, *Auto-Encoding Variational Bayes*, 2013. arXiv:1312.6114.

[20] G. McLachlan, "Mahalanobis distance," *Resonance*, vol. 4, no. 6, pp. 20–26, 1999.

[21] T. Amarbayasgalan, V. H. Pham, N. Theera-Umpon, and K. H. Ryu, "Unsupervised Anomaly Detection Approach for Time-Series in Multi-Domains Using Deep Reconstruction Error," *Symmetry*, vol. 12, no. 8, 2020.

[22] A. Dridi, C. Boucetta, S. E. Hammami, H. Afifi, and H. Moungla, "STAD: Spatio-Temporal Anomaly Detection Mechanism for Mobile Network Management," *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 894–906, 2021.

[23] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt, "Support Vector Method for Novelty Detection," *Advances in Neural Information Processing Systems*, pp. 582–588, 2000.

[24] H. Drucker, C. C. Burges, L. Kaufman, A. J. Smola, and V. N. Vapnik, "Support Vector Regression Machines," in *Proceedings of the Advances in Neural Information Processing Systems*, ser. NIPS 1996, vol. 4, MIT Press, 1997, pp. 155–161.

[25] M. Said Elsayed, N.-A. Le-Khac, S. Dev, and A. D. Jurcut, "Network Anomaly Detection Using LSTM Based Autoencoder," in *Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, ser. Q2SWinet '20, Alicante, Spain: Association for Computing Machinery, 2020, pp. 37–45.

[26] M. P. Novaes, L. F. Carvalho, J. Lloret, and M. L. Proença, "Long Short-Term Memory and Fuzzy Logic for Anomaly Detection and Mitigation in Software-Defined Network Environment," *IEEE Access*, vol. 8, pp. 83 765–83 781, 2020.

[27] T. J. Lee, J. Gottschlich, N. Tatbul, E. Metcalf, and S. Zdonik, *Greenhouse: A Zero-Positive Machine Learning System for Time-Series Anomaly Detection*, 2018. arXiv:1801.03168 [cs.AI].

[28] Deeplearning4j. (). "What is early stopping?" [Online]. Available: https://deeplearning4j.konduit.ai/tuning-and-training/early-stopping. Accessed: 2021-01-13.

[29] K. Gurney, An introduction to neural networks. CRC press, 1997.

[30] A. Sherstinsky, "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network," *Physica D: Nonlinear Phenomena*, vol. 404, p. 132 306, 2020.

[31] J. Hochenbaum, O. S. Vallis, and A. Kejariwal, *Automatic Anomaly Detection in the Cloud Via Statistical Learning*, 2017. arXiv:1704.07706 [cs.LG].

[32] M.-C. Lee, J.-C. Lin, and E. G. Gran, *How Far Should We Look Back to Achieve Effective Real-Time Time-Series Anomaly Detection?* 2021. arXiv:2102.06560 [cs.LG].

[33] A. Lavin and S. Ahmad, "Evaluating Real-Time Anomaly Detection Algorithms – The Numenta Anomaly Benchmark," in *Proceedings of the IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, 2015, pp. 38–44.

[34] I. Numenta. (). "NAB: Numenta Anomaly Benchmark [Online code repository]," [Online]. Available: https://github.com/numenta/NAB. Accessed: 2021-01-13.

[35] H. Ren et al., "Time-series anomaly detection service at microsoft," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '19, Anchorage, AK, USA: Association for Computing Machinery, 2019, pp. 3009–3017.

[36] F. Karim, S. Majumdar, H. Darabi, and S. Chen, "LSTM Fully Convolutional Networks for Time Series Classification," *IEEE Access*, vol. 6, pp. 1662–1669, 2018.

Daniel Vajda received his B.Sc. degree in electrical engineering at the Budapest University of Technology and Economics, Hungary, in 2021. Currently, he is an M.Sc. student with the same university. He is a research fellow at NETvisor Ltd. His research interests include network and service management, network telemetry, machine learning, and big data analytics.

Adrian Pekar received the Ph.D. degree in computer science from the Technical University of Košice, Slovakia, in 2014. Currently, he is an Assistant Professor with the Department of Networked Systems and Services, Budapest University of Technology and Economics, Hungary. Prior to this, he held research, teaching, and engineering positions in New Zealand and Slovakia. His research interests include network traffic classification and management, traffic measurement data reduction and visualisation, and stream processing.

Karoly Farkas did his habilitation in 2017 at the Budapest University of Technology and Economics (BME), received his Ph.D. degree in Computer Science in 2007 from ETH Zurich, and his M.Sc. degree in Computer Science in 1998 from BME. Currently he is working in parallel as an associate professor at BME, and as the head of R&D at NETvisor Ltd. His research interests cover, among others, the field of infocommunication networks, IoT, Industry 4.0 and AI. He has published more than 100 scientific papers, given a plenty of regular and invited talks and acted as organizer or TPC member of numerous scientific conferences. Currently he acts as board member of the Hungarian Scientific Association for Infocommunications (HTE). He is the coordinator of the local Cisco Networking Academy and was the founding initiator of the Cisco IPv6 Training Laboratory and the NetSkills Challenge student competition at BME. Between 2012-2015 Dr. Farkas has been awarded the Bolyai Janos Research Fellowship of the Hungarian Academy of Sciences