Cyber-attack detection method based on RNN

Dušan Nedeljković, Živana Jakovljević, *Member, IEEE*

Abstract—**Current and forthcoming market requirements bring huge challenges to today manufacturing. Answer to the changing demands and high product variety is found in the integration of the Internet of Things (IoT) and Cyber-Physical Systems (CPS) into industrial plants. CPS as smart devices capable of data processing and information exchange enable fast adaptation of manufacturing resources to production of diversified products. Nevertheless, fully implemented internet communication at factory shop floor opens up a whole new area for potential cyber-attacks. The consequences of attacks can have a negative influence on the system or even endanger human lives. Therefore, defence techniques must be developed to ensure a high level of protection. Early detection of cyber-attacks is crucial to minimize or completely avoid the negative effects of the attack and keep the system safe and reliable. In this work, we propose an attack detection method based on deep learning approach. We explore the application of several deep learning architectures based on Simple Recurrent Neural Networks (Simple RNN) and Long Short-Term Memory (LSTM) based RNN for generation of the detection mechanisms tailored to the concrete process. Our method was experimentally verified using real world data and it proved to be effective, as it detected all considered attacks without false positives.**

Index Terms—**Cyber security; Cyber Physical Systems; Internet of Things; Deep learning; Recurrent Neural Network.**

I. Introduction

Traditional industrial systems are experiencing changes driven by market requirements, especially in terms of manufacturing systems adaptability to changing demands that should be achieved along with the preservation of their efficiency and productivity. Such changes are seeking smart components (sensors, actuators…) in the form of physical devices with integrated computation and communication capabilities - Cyber-Physical Systems (CPS) [1, 2]. Within an industrial plant, smart devices interact with a central control system and with each other, share information over the Internet (usually wireless), creating and exchanging large amounts of data through Industrial Internet of Things (IIoT) [3, 4]. Data generated through fully implemented communication are stored in clouds or on the edge and used to make business related conclusions and timely control decisions. The integration of IoT and CPS into plant represents the core of the concept called Industry 4.0 [5]. This approach was created to meet performances, availability, flexibility requirements, as well as a new level of product individualization. However, openness and adopting the internet as the main way of communication in traditionally isolated industrial control systems introduce new risks and make these systems exposed to various types of cyber-attacks. Consequences of the attacks can degrade system performance, cause serious damages and production losses, or even affect human lives [6]. Therefore, the development of cyber-attack detection techniques represents one of the main industrial security issues today. To reduce or completely avoid the negative effects of attacks, they must be timely detected despite the fact they are trying to stay stealthy.

In general, cyber-attacks can be roughly divided into two categories: Denial of Service (DoS) attacks and deception attacks [7]. DoS attacks interfere with the data flows, making them temporarily or permanently unavailable [8]. DoS attacks are not necessarily system-dependent, i.e. do not require prior knowledge about the system they attack. Deception attacks, on the other hand, affect data integrity by injecting malicious data with an aim to completely change system behaviour [9]. Unlike DoS, deception attacks need knowledge about system resources, in order to attack the most sensitive system parts, while remaining stealthy. Mathematically, DoS attacks can be described with $\bar{x}_i \in \emptyset$, and deception attacks with $\bar{x}_i = x_i + x_i^d$, where $\bar{x}_i$, x_i and x_i^d represent received, measured and data injected by attackers, respectively [10].

The appearance of new attacks also triggers the development of new defence techniques. Today, there are a number of techniques that can be split into two categories: data-centric and design-centric [11]. Data-centric techniques are based on the collected data, while design-centric are oriented to system analytical models and its control algorithms. For example, the detection of false injected data [12] and compromised sensory data [13] were performed using a design-centric method based on cumulative sum (CUSUM). Also, a number of data-centric detection techniques were developed, such as 1D convolutional neural networks [14], and autoregression modelling and control limits [15] that were deployed for anomaly detection in Industrial Control Systems. ε-insensitive support vector regression (ε-SVR) was utilized for this security issue in the system with distributed control [16].

In this work, we propose a data-centric method for cyber-attacks detection in continuous time controlled systems. The method is based on deep learning model of normal system behaviour, created using Recurrent Neural Networks (RNN). The performance of the method is evaluated on a signal from the real-world system with several cyber-attacks. In particular, we have considered data obtained from a smart actuator as a part of the system with distributed control. The idea of

Dušan Nedeljković is with the Faculty of Mechanical Engineering, University of Belgrade, Kraljice Marije 16, 11020 Belgrade, Serbia (e-mail: dnedeljkovic@mas.bg.ac.rs).

Živana Jakovljević is with the Faculty of Mechanical Engineering, University of Belgrade, Kraljice Marije 16, 11020 Belgrade, Serbia (e-mail: zjakovljevic@mas.bg.ac.rs).

utilizing RNN for this purpose is based on their capability to use sequences and memorize the information obtained in the previous calculations. The memory from prior computation allows RNN to better understand data dependencies and to create a good prediction. Additionally, we compare the results obtained using different RNN architectures and select the best model according to the defined criteria.

The remainder of the paper is structured as follows. Section 2 briefly outlines used RNN architectures, whereas Section 3 refers to the developed method for signal attacks detection. In Section 4 we represent the results of the proposed method evaluation using data from the real-world application with inserted attacks. Conclusions and future work guidelines are provided in Section 5.

II. Recurrent Neural Network

Recurrent Neural Networks differ from standard feed-forward neural networks by connection where the current output vector **y**(*t*) depends not only on the present input **x**(*t*), but also on the recurrent input representing previous hidden state **h**(*t*-1). Thus, using history, network learns and understands the sequential nature of the data. Fed with the input and the previous hidden state vector, RNN cyclically computes output for every element of a sequence. Two types of RNNs are used in this paper: 1) Simple RNN, and 2) Long Short-Term Memory (LSTM).

A. *Simple RNN*

Simple RNN also called Elman network [17], represents a fully-connected network with a feedback. The loop keeps the hidden state vector at a previous time step **h**(*t*-1) and feeds it with the new input vector **x**(*t*), as shown in Fig. 1. Therefore, Simple RNN has the most general topology and most similar to the regular neural networks architectures.

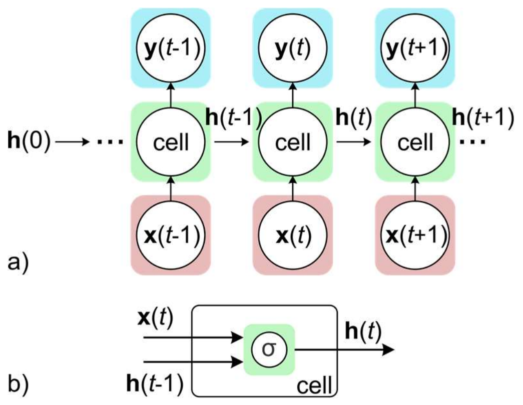

Fig. 1. Simple RNN: a) network architecture; b) cell(t) architecture

The initial value (usually set to 0) of the hidden state vector is denoted by **h**(0). Hidden state vector **h**(*t*) at a time step *t* is calculated as follows:

$$\mathbf{h}(t) = \sigma(\mathbf{W}_{\mathrm{xh}}\mathbf{x}(t) + \mathbf{W}_{\mathrm{hh}}\mathbf{h}(t-1) + \mathbf{b}_{\mathrm{h}}), \qquad (1)$$

where $\mathbf{b}_\mathrm{h}$ is a bias vector, σ represents the activation function, $\mathbf{W}_\mathrm{xh}$ and $\mathbf{W}_\mathrm{hh}$ denote the input and hidden weight matrices, respectively. RNN output **y**(*t*) is defined by the following equation:

$$\mathbf{y}(t) = \sigma(\mathbf{W}_{\mathrm{ho}}\mathbf{h}(t) + \mathbf{b}_{\mathrm{o}}), \qquad (2)$$

where σ is output activation function, $\mathbf{W}_\mathrm{ho}$ represents output weight matrix, and $\mathbf{b}_\mathrm{o}$ is a bias vector.

B. *LSTM*

LSTM is developed to overcome the vanishing gradient problem that in a number of cases makes regular RNN hard to train [18]. Recurrently connected modules called memory blocks present the basis of LSTM architecture. Each memory block consists of one or more memory cells connected in a certain way with a set of multiplicative units (gates), as shown in Fig. 2.

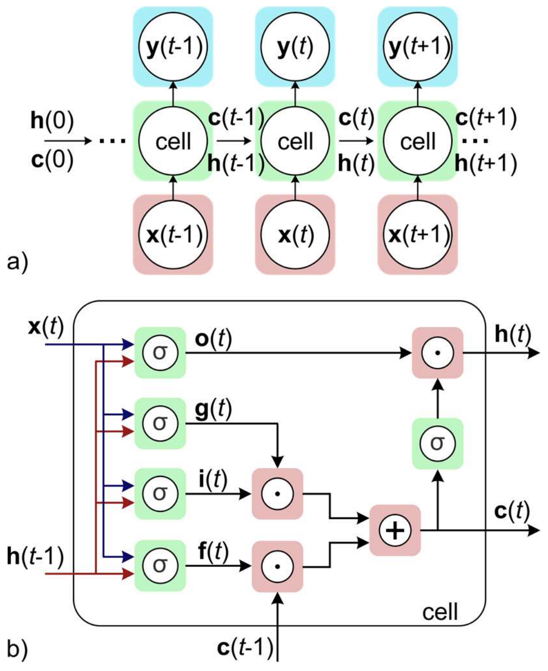

Fig. 2. LSTM: a) network architecture; b) cell (t) architecture

h(0) and **c**(0) are usually set to 0 and represent the initial hidden state and cell state vectors, respectively. Input (*i*), Output (*o*) and Forget (*f*) gates regulate when new information enters, select useful information as output and forget (remove) unnecessary information from the current cell state, respectively. All gates have a common task to prevent memory from perturbation by irrelevant inputs and outputs and thus ensure long term memory storage. LSTM block also contains candidate hidden state (*g*), which is based on current input and the previous hidden state. Gates and hidden state vectors are computed in the following way:

$$\begin{aligned}\mathbf{i}(t) &= \sigma(\mathbf{W}_{\mathrm{xi}}\mathbf{x}(t) + \mathbf{W}_{\mathrm{hi}}\mathbf{h}(t-1) + \mathbf{b}_{\mathrm{i}}),\\ \mathbf{f}(t) &= \sigma(\mathbf{W}_{\mathrm{xf}}\mathbf{x}(t) + \mathbf{W}_{\mathrm{hf}}\mathbf{h}(t-1) + \mathbf{b}_{\mathrm{f}}),\\ \mathbf{o}(t) &= \sigma(\mathbf{W}_{\mathrm{xo}}\mathbf{x}(t) + \mathbf{W}_{\mathrm{ho}}\mathbf{h}(t-1) + \mathbf{b}_{\mathrm{o}}),\end{aligned} \qquad (3)$$

$$\mathbf{g}(t) = \sigma(\mathbf{W_{xg}}\mathbf{x}(t) + \mathbf{W_{hg}}\mathbf{h}(t-1) + \mathbf{b_g}).$$

Weight matrices $\mathbf{W_x}$ and $\mathbf{W_h}$ are divided into 4 parts, where each part belongs to corresponding gate: *i*, *f*, *o* and *g*. Other variables labelling is in accordance with Section II-A. The cell state is determined by its previous value $\mathbf{c}(t\text{-}1)$, candidate hidden state, input and forget gates.

$$\mathbf{c}(t) = \mathbf{f}(t) \odot \mathbf{c}(t-1) + \mathbf{i}(t) \odot \mathbf{g}(t), \tag{4}$$

where $\odot$ represents element-wise multiplication of the vectors. Hidden state vector represents the output of LSTM cell:

$$\mathbf{h}(t) = \mathbf{o}(t) \odot \sigma\big(\mathbf{c}(t)\big). \tag{5}$$

LSTM output $\mathbf{y}(t)$ is defined in the following way:

$$\mathbf{y}(t) = \sigma\big(\mathbf{W_{hy}}\mathbf{h}(t) + \mathbf{b_y}\big), \tag{6}$$

where $\mathbf{W_{hy}}$ represents output weight matrix, and $\mathbf{b_y}$ is a bias vector.

III. Sensor Signal Attack Detection Method

The method that we propose in this paper consists of two phases: offline RNN training and online attack detection. The training process is based on sensory data recorded under normal conditions (without anomalies/attacks). The model generated during training describes proper sensor operating and represents the basis for signal prediction in attack detection part. The second step considers the difference between signal values estimated by model and measured sensor signal values (Fig. 3).

In our method, the current value of the sensor signal $x(t)$ is predicted based on the sequence of previous z values $x(t\text{-}z)$,..., $x(t\text{-}1)$. Thus, through the RNN training process ordered pairs are created:

$$\begin{aligned}(\mathbf{x}(t), y(t)) \in \{&([x(1), \dots, x(z)], x(z+1)),\\ &\big([x(2), \dots, x(z+1)], x(z+2)\big), \dots,\\ &([x(n-z), \dots, x(n-1)], x(n))\},\end{aligned} \tag{7}$$

where $\mathbf{x}(t)$, $t\in[z+1, n]$ represents vector of input variables, and $y(t)$ denotes corresponding response. During training, a few parameters should be tuned to get the best possible RNN model. The sequence length z specifies the number of samples which determine the next predicted value. Batch size (*bs*) defines the number of samples to work with before updating the weights. The network architecture is determined by the number of layers (*nl*) and units per layer (*ul*). A unique RNN model is obtained by varying one or more parameter values. The first criterion for selecting the best model is the minimum value of p, defined as follows:

$$p = \frac{1}{n-z}\sum_{t=z+1}^{n}|x(t) - \hat{x}(t)|. \tag{8}$$

Parameter p represents the mean absolute difference between real $(x(t))$ and predicted $(\hat{x}(t) = y(t))$ values of sensor readings over the test dataset. To make the model as simple as possible and to reduce training and testing time, the number of model parameters should not be too large. Furthermore, to keep the latency that is introduced during online application of detection mechanisms at acceptable level, the number of RNN model parameters is especially significant and should be as low as possible. Thus, a small number of layers in the network and a small number of units per layer represent the second criterion for model selection. However, it should be emphasized that insufficient number of layers and units may lead to an inaccurate model and erroneous attack detection thereof.

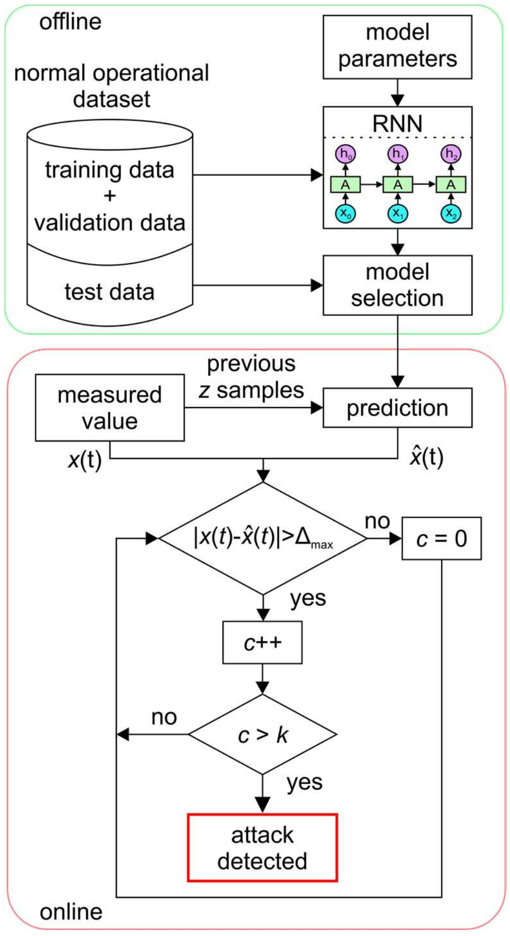

Fig. 3. Algorithm for sensor signal attacks detection

The selected RNN model, according to the defined criteria, represents the output from the offline phase, i.e. the input to the online phase. Online attack detection is based on the absolute difference between measured and predicted values. If error exceeds the detection threshold Δ_{max} consecutively for k samples, the attack is present. The threshold value Δ_{max} is defined as a sum of mean value μ and standard deviation σ of discrepancy between measured and estimated values over the training data:

$$\Delta_{max} = \mu + \sigma, \tag{9}$$

where μ and σ are:

$$\mu = \frac{1}{n-z}\sum_{t=z+1}^{n}(x(t) - \hat{x}(t)),$$
$$\sigma = \sqrt{\frac{1}{n-z}\sum_{t=z+1}^{n}(x(t) - \mu)^2}. \quad (10)$$

IV. Experimental Results

The proposed method for attacks detection is experimentally evaluated on electro-pneumatic positioning system [16] with control distributed on smart sensor and smart actuator. Smart actuator consists of linear rodless pneumatic cylinder supplied by air through electro-pneumatic air pressure regulator (output pressure in the range 2-6 bar) on one, and mechanically controlled air pressure regulator (constant output pressure of 4 bar) on the other side. As a part of the smart actuator, local controller – wireless node controls output pressure on the electro-pneumatic regulator.

On the other hand, the smart sensor represents linear encoder (placed along the cylinder) that is equipped with its own local controller – wireless node (LC2). LC2 obtains pulses from the encoder and determines the position of the piston. Corresponding to the desired position, LC2 generates a control signal (in the range 0-1) and transmits it to LC1 using wireless communication. Furthermore, LC1 converts the received signal value to the analog voltage in the range 0-10 V and sends it to electro-pneumatic regulator. Electro-pneumatic regulator gives at its output the pressure proportional to the input voltage. Air pressure difference between the two sides of the piston causes piston movement.

Sensor signals were recorded during normal system functioning, without attacks. In particular, the voltage between LC1 and electro-pneumatic air pressure regulator was acquired using characteristic piston trajectory that contained positions of 50, 400, 250, 400, and 100 mm. The defined trajectory was cyclically repeated 100 times. The data acquisition was performed with a sampling rate of 100 Hz, which led to a total of 400,000 records.

To find the optimal model of sensory data we used different architectures based on two selected RNN types. We have opted to use two RNN layers since a model with more than 2 RNN layers contains a large number of model parameters, whereas, a single layer cannot meet the required model performance. The architecture that has shown the best results (Fig. 4) for all models is: RNN layer (units) - Dropout (rate) - RNN layer (units) - Dropout (rate) - Dense (1).

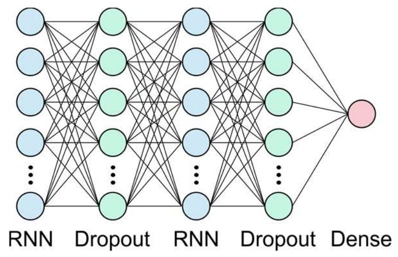

Fig. 4. The chosen network architecture

Dropout layer helps prevent overfitting by temporarily removing minimal units from the network with a rate from 0 to 1 (0 to 100% of all units). A Dense layer is a fully connected layer, where each input node is connected to each output node. Five RNN architecture related parameters were varied during the process of finding the optimal model. The considered values of RNN architecture parameters are given in Table I. In addition, 4 different batch sizes that affected the values of particular RNN parameters were explored.

This variation of parameters resulted in a total of 432 different models (216 models for both RNN types: Simple RNN and LSTM). During models training, the whole dataset is divided into training, validation and test part, with a ratio of 80/10/10 %, respectively. The validation and training losses were almost unchanged after 5 epochs, so this was selected as the optimal number of epochs during models training. Through the process of finding the most appropriate model, we used Adam optimizer with learning rate of 0.001 and rectified linear unit (ReLU) activation function. The cost function that we utilized was mean squared error (MSE).

TABLE I
VARIED RNN PARAMETERS

Parameter	Value
sequence length (*z*)	2, 5, 10
dropout rate (*dp*)	0.05, 0.1
number of units in layer1 (*ul1*)	8, 16, 32
number of units in layer2 (*ul2*)	8, 16, 32
batch size (*bs*)	8, 16, 32, 64

The models were trained in Python using a Visual Code Studio with Keras and TensorFlow at the background. To test our proposed signal attack detection method and to choose an appropriate model, a number of attacks have been created. In this paper, we present three attacks of different types and duration. Attack 1 (A_1) utilizes sinus function to generate x value, attack 2 (A_2) increases x value linearly, whereas in attack 3 (A_3) value of x is immediately set to 0. The following equations respectively define these three attacks:

$$A_1: x(t) = 0.5 + \sin(0.005 \cdot t), t = 1, 2, \dots, 1300$$
$$A_2: x(t) = x(t) + 0.00007 \cdot t + 0.0005 \cdot \text{rand}(),\ t = 1, 2, \dots, 400 \quad (11)$$
$$A_3: x(t) = 0,\ t = 1, 2, \dots, 500.$$

According to the first criterion (minimum value of p), the best five models of both RNN types were chosen. The performances of the selected models are given in Table II where the best-performing models for both of the considered RNN types are highlighted; Fig. 5 presents histogram of absolute errors between measured and predicted values for these models.

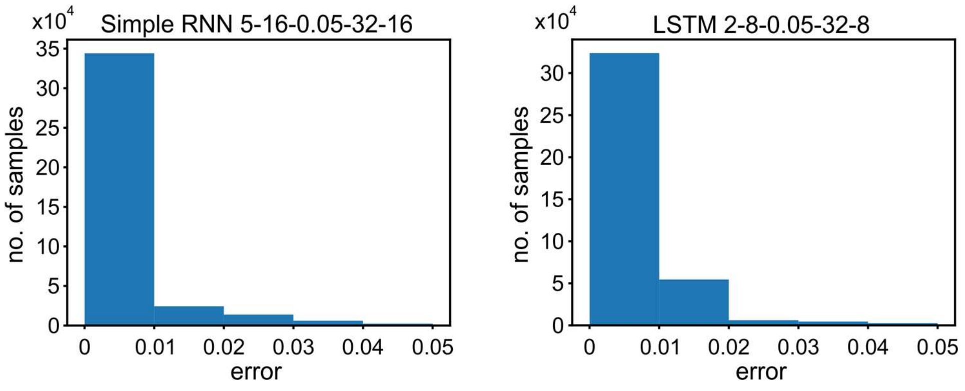

Fig. 5. Histograms of absolute errors between measured and predicted values for selected models; for the clarity of presentation, histograms do not contain 8084 (2.02%) samples whose errors are in the range [0.05, 0.98795] for Simple RNN and 7482 (1.88%) samples whose errors are in the range [0.05, 0.97650] for LSTM model.

TABLE II
PERFORMANCES OF THE SELECTED RNN MODELS

RNN type-*z*-*bs*-*dp*-*ul1*-*ul2*	*p*	no. of param.	attacks detected	false positives	false negatives
SimpleRNN-5-16-0.05-32-16	**0.00650**	**1889**	**2/3**	**1**	**1**
SimpleRNN-5-64-0.05-16-16	0.00697	833	3/3	>10	0
SimpleRNN-10-32-0.05-32-16	0.00720	1889	3/3	>10	0
SimpleRNN-10-16-0.1-32-16	0.00813	1889	3/3	>10	0
SimpleRNN-5-64-0.05-32-16	0.00857	1889	3/3	>10	0
LSTM-5-8-0.1-32-16	0.00391	7505	1/3	0	2
LSTM-10-64-0.05-32-16	0.00457	7505	2/3	0	1
LSTM-5-8-0.1-32-32	0.00486	12705	3/3	6	0
LSTM-5-64-0.1-16-16	0.00487	3281	2/3	0	1
LSTM-2-8-0.05-32-8	**0.00500**	**5673**	**3/3**	**0**	**0**

From Table II it can be observed that LSTM-2-8-0.05-32-8 model can detect all three attacks without false-positive results. However, no model with Simple RNN architecture meets such requirement. Some of the models are not suitable for attack detection, although they provide an excellent prediction. The reason is that these models do not make difference between the attacks and signal under normal operating conditions. Simple RNN-5-16-0.05-32-16 is the model that is the closest to detecting all attacks without false positives when Simple RNN architectures are considered. Measured signal, predicted values, and detected attacks for two selected models are represented in Fig. 6. The input data and their prediction are shown in blue and green lines, respectively. Moments when the attack was detected, are marked with red *. The detection method based on both models successfully detected attacks 1 and 3.

However, the method based on the Simple RNN model was not able to detect attack 2. Besides, this method also has one false-positive result that is not shown in Fig. 6.

It can be observed from Fig. 6 that the LSTM-based method detected attack 3 more effectively than Simple RNN method, i.e. it required the smaller number of samples from the beginning of the attack to its detection resulting in lower attack detection latency. In the case of attack 1 detection, both methods proved to be equally effective. Therefore, model LSTM-2-8-0.05-32-8 is considered as the best.

V. CONCLUSION

We have proposed a method for signal attack detection based on a prediction of signal value using deep learning algorithms. More precisely, we used two popular recurrent neural network architectures: Simple RNN and LSTM.

Deep learning models have been trained on a dataset obtained from an electro-pneumatic positioning system under normal conditions (without attacks). We have generated over 400 models by varying several parameters and according to

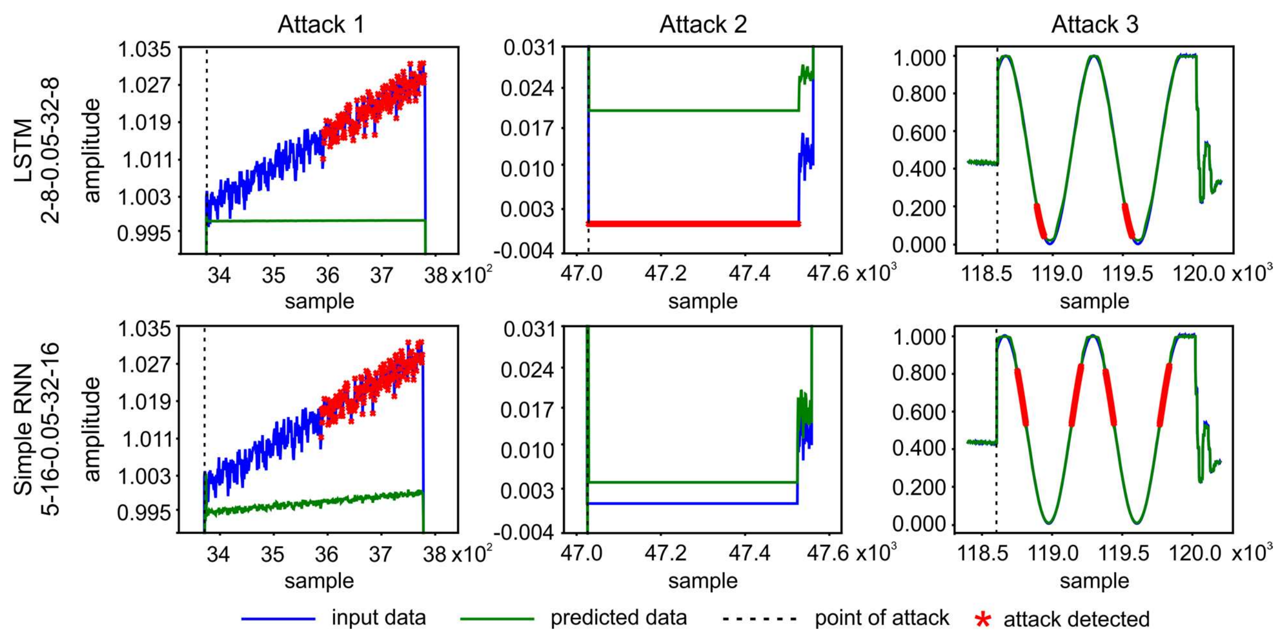

Fig. 6. Detected attacks

the defined criteria, we have selected the best models for each RNN architecture. For the method evaluation, we have created three different attacks. The method proved effective in cases of LSTM architecture as it detected all attacks without false positives.

Further research will focus on the implementation of the method in real-world on low level controllers of CPS, primarily on the electro-pneumatic positioning system. Additionally, the method will be tested on publicly available datasets with a number of different attacks.

Acknowledgment

This research was supported by the Science Fund of the Republic of Serbia, grant No. 6523109, AI - MISSION4.0, 2020-2022.

References

[1] L. Monostori, B. Kádár, T. Bauernhansl, S. Kondoh, S. Kumara, G. Reinhart, O. Sauer, G. Schuh, W. Sihn, K. Ueda, "Cyber-physical systems in manufacturing," *CIRP Annals*, vol. 65, no. 2, pp. 621-641, 2016.

[2] Z. Jakovljevic, V. Majstorovic, S. Stojadinovic, S. Zivkovic, N. Gligorijevic, M. Pajic, "Cyber-physical manufacturing systems (CPMS)," Proc. 5th Int. Conf. Adv. Manuf. Eng. Technol. (NEWTECH 2017), Belgrade, Serbia, pp. 199-214, June 2017.

[3] L. Atzori, A. Iera, G. Morabito, "The Internet of Things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787-2805, Oct. 2010.

[4] Z. Jakovljevic, V. Lesi, S. Mitrovic, M. Pajic, "Distributing Sequential Control for Manufacturing Automation Systems," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 4, pp. 1586-1594, 2020.

[5] H. Kagermann, W. Wahlster, J. Helbig, *Recommendations for implementing the strategic initiative INDUSTRIE 4.0*, 2013. [Online]. Available: http://www.acatech.de

[6] Z. Jakovljevic, V. Lesi, M. Pajic, "Attacks on Distributed Sequential Control in Manufacturing Automation," *IEEE Transactions on Industrial Informatics*, 2020, doi: 10.1109/TII.2020.2987629

[7] A. Teixeira, D. Pérez, H. Sandberg, K. H. Johansson, "Attack Models and Scenarios for Networked Control Systems," Proceedings of the 1st International Conference on High Confidence Networked Systems, Beijing, China, pp. 55-64, Apr. 2012.

[8] S. Amin, A. A. Cárdenas, S. S. Sastry, "Safe and secure networked control systems under Denial-of-Service attacks," Proceedings of the 12th International Conference on Hybrid Systems: Computation and Control, San Francisco, USA, pp. 31-45, Apr. 2009.

[9] Q. Zhang, K. Liu, Y. Xia, A. Ma, "Optimal Stealthy Deception Attack Against Cyber-Physical Systems," *IEEE Transactions on Cybernetics,* May 2019.

[10] D. Ding, Q.-L. Han, Y. Xiang, X. Ge, X.-M. Zhang, "A survey on security control and attack detection for industrial cyber-physical systems," *Neurocomputing*, vol. 275, pp. 1674-1683, 2018.

[11] H. S. Sánchez, D. Rotondo, T. Escobet, V. Puig, J. Quevedo, "Bibliographical review on cyber attacks from a control oriented perspective," *Annual Reviews in Control*, vol. 48, pp. 103-128, Sep. 2019.

[12] Y. Huang, J. Tang, Y. Cheng, H. Li, K. A. Campbell, Z. Han, "Real-Time Detection of False Data Injection in Smart Grid Networks: An Adaptive CUSUM Method and Analysis," *IEEE Systems Journal*, vol. 10, no. 2, pp. 532-543, June 2016.

[13] C. Murguia, J. Ruths, "CUSUM and chi-squared attack detection of compromised sensors," 2016 IEEE Conference on Control Applications (CCA), Buenos Aires, Argentina, pp. 474-480, Sept. 2016.

[14] M. Kravchik, A. Shabtai, "Detecting Cyber Attacks in Industrial Control Systems Using Convolutional Neural Networks," Proceedings of CPS-SPC 18 Conference, Toronto, Canada, pp. 72-83, Oct. 2018.

[15] D. Hadžiosmanović, R. Sommer, E. Zambon, P. H. Hartel, "Through the eye of the PLC: semantic security monitoring for industrial processes," Proceedings of 30th Annual Computer Security Applications Conference, New Orleans, USA, pp. 126–135, Dec. 2014.

[16] D. M. Nedeljkovic, Z. B. Jakovljevic, Z. Dj. Miljkovic, M. Pajic, "Detection of cyber-attacks in electro-pneumatic positioning system with distributed control," 27th Telecommunications Forum (TELFOR), Belgrade, Serbia, art. no. 8971062, Nov. 2019.

[17] J. Elman, "Finding Structure in Time," *Cognitive Science*, vol. 14, no. 2, pp. 179-211, Mar. 1990.

[18] S. Hochreiter, J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.