Next Generation Automated Reservoir Computing for Cyber Defense

Konstantinos Demertzis[1(✉)] and Lazaros Iliadis[2]

[1] School of Science and Technology, Informatics Studies, Hellenic Open University, Patra, Greece
demertzis.konstantinos@ac.eap.gr

[2] School of Engineering, Department of Civil Engineering, Faculty of Mathematics Programming and General Courses, Democritus University of Thrace, Kimmeria, Xanthi, Greece
liliadis@civil.duth.gr

Abstract. It is a fact, that important information related to systems' behavior and dynamics, can be revealed as time passes. Observing changes over time, can often lead to the detection of patterns and trends that might not be immediately apparent from a single system's snapshot. Additionally, the concept of time can be essential in understanding cause-and-effect relationships. Observing how changes in one variable over time affect changes in another, can gain insights into the causal relationships between different system components. Time series analysis of data that change over time, can be a powerful tool for understanding complex systems in the field of cyber security. Reservoir Computing (RC) is a Machine Learning technique, using a fixed and randomly generated high-dimensional dynamic system, called a Reservoir, to transform and classify input data. The reservoir acts as a nonlinear and temporal filter of the input data, which is then readout by a linear output layer. Continuous-Time Reservoir Computing (CTRC) is a type of recurrent neural network, aiming to model the continuous-time dynamics of the network's neurons. It is particularly useful for applications where time is critical and it can provide insights into the underlying system's dynamics. This paper proposes a next-generation CTRC for cyber defense, where the reservoir neurons are modeled as continuous-time dynamical systems. This means that their behavior is described by a system of differential equations that change over time. In order to model the drift phenomenon, identify the abnormal changes in the data, and adaptively stabilize the learning system. The CTRC parameters are optimized using the Reinforcement Learning (RL) method. The proposed system, as proved experimentally, has several advantages over discrete systems, including the ability to handle signals with high sampling rates and to effectively capture real cyber security signals' continuous nature.

Keywords: Reservoir Computing · Continuous-Time Reservoir Computing · Cyber Defense · Time Series Analysis

Published by Springer Nature Switzerland AG 2023
I. Maglogiannis et al. (Eds.): AIAI 2023, IFIP AICT 676, pp. 16–27, 2023.
https://doi.org/10.1007/978-3-031-34107-6_2

1 Introduction

RC is a computational framework that utilizes the dynamics of a high-dimensional, randomly connected network to process and learn from input signals [1]. It is an efficient and powerful technique for solving complex machine-learning tasks, particularly those involving time-series data. At the heart of a reservoir computing system is a reservoir, which is a randomly connected network of nodes. The nodes in the reservoir are typically arranged recurrently, meaning they have connections to each other that allow information to be processed over time. The input signal is fed into the reservoir, interacting with the network's dynamics to produce a high-dimensional state vector. The state vector represents the current state of the reservoir, and it captures the temporal patterns and features of the input signal. The state vector is then fed into a readout layer, a simple, linear layer that maps the reservoir states to the desired output [2]. The readout layer is typically trained using linear regression or another simple learning algorithm. It learns to produce the correct output based on the input signal and the current state of the reservoir [3].

Reservoir computing is a machine learning algorithm employing linear optimization. It is suitable for processing information generated by dynamical systems, using observed time-series data. It is important that it requires very small training data sets, and thus it has small requirements of computing resources. One of the key advantages of reservoir computing is that the reservoir itself is not trained. Instead, the reservoir's random connections and internal dynamics are fixed, and the readout layer is the only trained part of the system. This greatly simplifies the training process and makes it more efficient. Another advantage of RC is that it can have many nodes, enabling it to capture the input signal's complex temporal patterns and features. This makes it particularly well-suited to sophisticated tasks where the input signal is a complex, time-varying waveform [4].

Next-generation automated RC [5] is an emerging technology that has the potential to enhance cyber defense greatly. It involves applying the basic principles of RC to analyze network traffic [6] and identify anomalies indicative of cyberattacks or other security threats. The basic concept of the next-generation automated RC is that a high-dimensional, randomly connected network can be used to model complex data patterns and relationships on cyber security prospects. This can be particularly useful when traditional rule-based detection methods are ineffective, such as zero-day attacks or other advanced persistent threats.

One extremely novel application of next-generation automated RC for cyber defence is the CTRC [7]. It is an extension of the RC paradigm that operates in continuous rather than discrete time. In a traditional discrete-time reservoir computing system, input signals are fed into the reservoir at discrete intervals. In contrast, the input signal is processed in real-time without discretization in a CTRC system. Specifically, in CTRC, the reservoir dynamics are described by continuous-time differential equations. The input signal drives the reservoir dynamics, producing an output signal. The output signal is then passed through a readout layer to produce the desired output. The continuous-time nature of the system enables the reservoir to process input signals in real time without discretization. This is particularly useful in applications where the input signal is a continuous data stream, such as large-scale network traffic analyses, which can be used to make predictions and detect anomalies in continuous data streams [8].

One of the challenges of CTRC is that it can be more difficult to train and optimize than discrete-time RC. This is because the reservoir dynamics are described by continuous-time differential equations, which can be more complex to model and optimize than discrete-time ones. Specifically, the biggest challenge with continuous-time differential equations is that they require numerical integration, which can be computationally expensive and lead to instability or accuracy issues if not implemented correctly. Furthermore, optimizing continuous-time differential equations can be more difficult than that discrete-time equations, as the optimization problem becomes a Partial Differential Equation (PDE) instead of a simple algebraic equation. In order to overcome these challenges, the proposed CTRC's system parameters are optimized using RL method.

2 Methodology

The proposed CTRC system is modelled using continuous-time differential equations, and the system parameters are optimized using the RL method and, specifically, the Q-Learning approach. The following paragraphs are presenting the specific steps required to combine these approaches:

2.1 Defining the CTRC System

The architecture of a CTRC system, includes the input, reservoir, and output layers. The respective connection and input weights are random in RC. The reservoir weights are scaled in such a way as to ensure the *Echo State Property* (ESP), which is defined as the state in which the reservoir is an "echo" of the entire input history. Of course, this is partly determined by its architecture [9]. The discrete layers of the CTRC are only those of input $u(n)$ and output $y(n)$ as they are defined by the problem. The hidden layers are clustered in an RC region, and their number is indistinguishable. The neurons in the RC, $x(n)$, are connected by some percentage, which determines how sparsity the RC will be. A depiction of the RC architecture [10] is presented in the following Fig. 1.

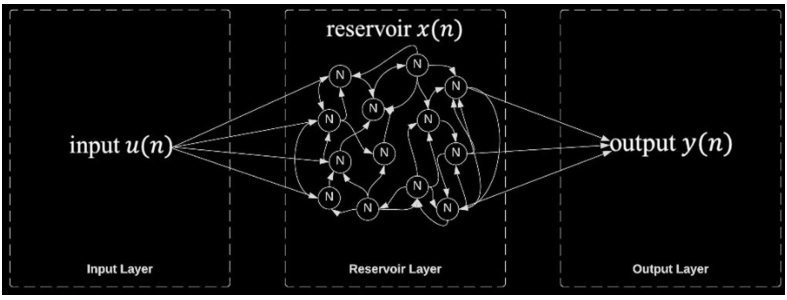

Fig. 1. RC architecture

The synaptic associations that link the levels together and the RC are characterized by a value that identifies the weights [11]. In CTRC, each input neuron is connected via

$W^{in}{}_{ij}$ weights (i-input neuron, j-neuron in the RC) to each neuron from the RC. Although normalized, these weights are determined randomly before training, and their values are the final ones as they do not change during training. Also, each RC neuron is connected to each other neuron, via weights W_{jk} (j-neuron in RC, k-neuron RC, and $j \neq k$). The respective weights, although normalized, are randomly determined before training and their values do not change. We use $x^{(l)}(t) \in R^{N_R}$ to declare the status of level l at time t. By omitting the bias conditions, the first level state transition function is defined by the following Eq. 1, [12, 13]:

$$x^{(1)}(t) = \left(1 - a^{(1)}\right)x^{(1)}(t-1) + a^{(1)} \tanh\left(W_{in}u(t) + \hat{W}^{(1)}x^{(1)}(t-1)\right) \quad (1)$$

For each level higher than $l > 1$ Eq. 1, has the following form (2) [7, 14]:

$$x^{(l)}(t) = \left(1 - a^{(l)}\right)x^{(l)}(t-1) + a^{(l)} \tanh\left(W^{l}x^{l-1}(t) + \hat{W}^{(l)}x^{(l)}(t-1)\right) \quad (2)$$

where $W_{in} \in R^{N_R \times N_U}$ is the input weight matrix, $\widehat{W}^{(l)} \in R^{N_R \times N_R}$ is the recurrent weight matrix for layer l, $W^{(l)} \in R^{N_R \times N_R}$ is the matrix containing the connection weights between layer l-1 and l, $a^{(l)}$ is the leaky parameter of layer l and *tanh* is the Tangent Hyperbolic function. Finally, each RC neuron is connected via $W^{out}{}_{jm}$ weights (j-neuron in the RC, m-neuron input) to the neurons in the output layer. The weights, located in the readout layer, are the only ones trained to get their final values [1, 15].

2.2 Defining the Differential Equations

The differential Eqs. 3 and 4, govern the behaviour of the state vectors $x1(t)$ and $x2(t)$ in the continuous-time reservoir computing system, enable the system to capture both temporal and spatial patterns of the network's traffic data. They are given by [16, 17]:

$$dx1/dt = -x1(t) + f1(W1x1(t) + \text{Win1}u(t)) \quad (3)$$

$$dx2/dt = -x2(t) + f2(W2x2(t) + \text{Win2}u(t) + Vx1(t)) \quad (4)$$

In the above Eqs. 3, 4, $x1(t)$ represents the state vector that captures the temporal patterns of the network traffic data, while x2(t) represents the state vector that captures the spatial patterns of the data. In Eq. 3, the term $-x1(t)$ represents the leaky integrator, which causes the state vector to decay over time. The term $f1(\text{W1} \times 1(\text{t}) + \text{Win1u(t)})$ represents the nonlinear activation function, which maps the input signal and the current state of the reservoir to a new state vector [18]. In Eq. 4, the term $-x2(t)$ represents the leaky integrator, and the term $\text{f2(W2} \times 2(\text{t}) + \text{Win2u(t)} + \text{V} \times 1(\text{t}))$ represents the nonlinear activation function. Moreover, V is a weight matrix that captures the interaction between the temporal and spatial patterns of the network traffic data. The above differential equations describe how the state vectors $x1(t)$ and $x2(t)$ change over time in response to the input signal u(t) and the current state of the reservoir. The weight matrices W1, $W2$, $Win1$, $Win2$, and V are optimized using Q-Learning algorithm to minimize the difference between the predicted output of the system and the true labels in a training dataset in order to model the drift phenomenon, identify the abnormal changes in the data, and adaptively stabilize the learning system.

2.3 Defining the RL Algorithm

The RL algorithm optimizes the CTRC system parameters. This involves using the Q-learning algorithm. Q-learning is a RL algorithm, and the following are the exact methodology steps to use Q-learning to optimize the CTRC system parameters [19, 20]:

1. Defining the state space: The state space for the Q-learning algorithm is based on the output of the CTRC system. The state space captures relevant information about the current network traffic analysis of the cyber defense scenario. The state space, denoted by S, represents the possible states of the cyber defense system at any given time. The state space can be defined in terms of the output of the CTRC system, which might include features such as network traffic patterns, system logs, or other relevant data. The state space can be discretized into a set of discrete states, $S = \{s_1, s_2,\ldots, s_n\}$, where n is the number of discrete states.
2. Defining the action space: The action space for the Q-learning algorithm is based on the possible actions that the agent can take to mitigate cyber-attacks. For example, the action space might include blocking traffic from a specific IP address, shutting down a compromised system, or initiating a backup of critical data. The action space, denoted by A, represents the set of actions that the agent can take to mitigate cyber-attacks. The action space can be defined based on the available defensive actions, such as blocking traffic from a specific IP address, shutting down a compromised system, or initiating a backup of critical data. The action space can be discretized into a set of discrete actions, $A = \{a_1, a_2,\ldots, a_m\}$, where m is the number of discrete actions.
3. Defining the reward function: The reward function for the Q-learning algorithm is based on the agent's performance in the cyber defence task. The reward function encourages the agent to mitigate cyber-attacks effectively and discourage ineffective or harmful actions. The reward function, denoted by R, is a function that maps a state-action pair to a scalar reward value. The reward function should encourage the agent to mitigate cyber-attacks effectively and discourage ineffective or harmful actions. The reward function 5, can be defined as follows [21]:

$$R(s, a) = r(s, a) \tag{5}$$

   where $r(s, a)$ is the immediate reward associated with taking action a in state s.
4. Initializing the Q-table: The Q-table is a lookup table that maps states and actions to expected rewards. The Q-table is initially set to arbitrary values. The Q-table, denoted by Q, is a lookup table that maps states and actions to expected rewards. The Q-table is initially set to arbitrary values. The Q-value for a state-action pair (s, a) is denoted by $Q(s, a)$.
5. Running the Q-learning algorithm: The use of the Q-learning algorithm to update the Q-table by iteratively selecting actions based on the current state and the values in the Q-table. The Q-table is updated using the Bellman equation, which estimates the expected future reward from the current state and action. The algorithm works as follows [19, 22]:
   a. At each time step t, the agent observes the current state st and selects an action at based on an exploration-exploitation tradeoff.
   b. The agent receives an immediate reward $rt = r(st, at)$ and transitions to a new state $st + 1$.

c. The Q-value for the current state-action pair (st, at) is updated using the Bellman Eq. 6:

$$Q(st, at) \leftarrow Q(st, at) + \alpha\left[rt + \gamma \text{maxa}a' Q\left(st + 1, a'\right) - Q(st, at)\right] \quad (6)$$

where α is the learning rate, γ is the discount factor, and $maxa' Q(st + 1, a')$ is the maximum expected reward over all actions in the next state $st + 1$.

d. The Q-table is updated at each time step, and the process continues until convergence is achieved.

6. Training the CTRC system: The Q-table adjusts the CTRC system parameters, such as the weight matrices and bias terms. The CTRC system is trained to optimize the expected future reward as the Q-table estimates.
7. Evaluating the performance: The performance evaluation process of the trained CTRC system is implemented based on a test dataset. This involves measuring accuracy, precision, recall, and F1-score metrics.
8. Refining the system: The CTRC algorithm parameters are refined based on the evaluation results. This involves automatically adjusting the set of parameters optimized in the Q-learning process.

3 Experiments

To provide a perfect simulation environment, the ***Factry.io*** data collecting platform and the ***InfluxDB*** were used [23]. Factry.io is a data collection and visualization platform that enables users to easily collect, monitor, and analyze data from various sources, such as machines, sensors, and devices [24]. It provides a user-friendly interface to connect to different data sources, set up data collection parameters, and visualize data in real-time. InfluxDB, on the other hand, is a time-series database designed to handle high volumes of data that are time-stamped. It is optimized for storing and retrieving time-stamped data and enables fast queries and data analysis. InfluxDB supports a variety of data types and formats, including numerical, string, and Boolean data, and provides a SQL-like query language to access and manipulate data. Factry.io can integrate with InfluxDB to store and query time-series data collected from various sources. Users can configure Factry.io to send data to InfluxDB, which can then be used for further analysis or visualization. InfluxDB's efficient data storage and retrieval capabilities make it an ideal choice for managing large volumes of time-series data collected by Factry.io.

The goal was to gather data about the industrial environment using the open-source OPC-UA collector protocol [25]. It is a time series of sensor data that has been compiled. In order to store sensor data in InfluxDB, programmable PLC controllers, SCADA systems, and construction equipment are used to collect the data. Time series or timestamps are best served by the storage database. Measurements or events that are tracked throughout time and gathered as time series data are used to create them. Such occurrences may include transactions, application performance monitoring, and server analytics [26]. Sensors or different kinds of analytics are examples of potential sources. In this instance, data for one year was gathered from three sensors' hourly quantifiable values within the context of a machine condition that runs continuously. The attack modifies the sensor settings to alter how some mechanisms operate, but the meters and displays of the entire system are not aware of this.

There is a tank specifically for raw water storage. A water level sensor is part of this, along with a valve that opens when the sensor detects a level of less than or equal to 0.5 m and closes when the level is higher than 0.8 m. Also, it has a pump whose operation is based on a procedure that separates the pressure levels through a semipermeable membrane. This is seen as a safety device because the pump shuts off immediately if the water level in the tank falls below 0.25 m. The attacker wants to increase the water without being noticed by a typical detection system that looks for irregularities. This is accomplished by altering the sensor and actuator information and creating the proper packets, which are altered so that the devices' functionality can be changed by the fieldbus communication.

The model was trained to recognize anomalies during the operation of the SCADA automations that manage the water tank in the situation under examination using the above-described technique. The success or failure of the anomaly recognition approach is largely dependent on the class separation threshold. This research suggests a trustworthy heuristic approach of selection, based only on assessment criteria, to identify an ideal threshold. The suggested technique specifically implies that a distance function, which calculates the distance d between the objects and the appropriate target category, is constructed during the training phase. For the binary class separation (normal or abnormal), a threshold is utilized.

The proposed algorithm calculating the density around each data point in order to identify the dynamic threshold. This is achieved by counting the number of points in a user-defined neighborhood (*Eps*-Neighbourhood) with the definition of thresholds. The purpose is to locate points in the center of the areas (core), on their borders (border), and points that involve noise (noise). The extra data points are added to the center of the regions if they are densely accessible, i.e., there is a chain of core points where each one belongs to the neighborhood (*Eps*-Neighborhood) of the next point and therefore to distinguish the extreme values for each time frame. Specifically, the neighborhood area of a point p is defined as the set of points for which the Euclidean distance between the points p, q is smaller than the parameter *Eps* [27, 28]:

$$\mathrm{N_{Eps}(p)} = \{\mathrm{q} \in \mathrm{D} |\ \mathrm{dist(p, q)} \leq \mathrm{Eps}\} \tag{7}$$

provided that $p = (p1, p2)$ and $q = (q1, q2)$, the Euclidean distance is defined as:

$$\sqrt{(q_{1-}p_1)^2 + (q_{2-}p_2)^2} \tag{8}$$

So, a point p is considered to be reachable from a point q based on a density determined by the parameters *Eps*, *MinPts* if:

$$\mathrm{p} \in \mathrm{N_{Eps}(q)} \text{ and } \mathrm{N_{Eps}}\ (q) \geq \mathrm{MinPts} \tag{9}$$

Two plots to visualize the dynamic threshold calculation depicted in the following Fig. 2.

Samples (outliers) are considered abnormal when the anomaly score departs from the expected behavior by the application of the dynamic threshold.

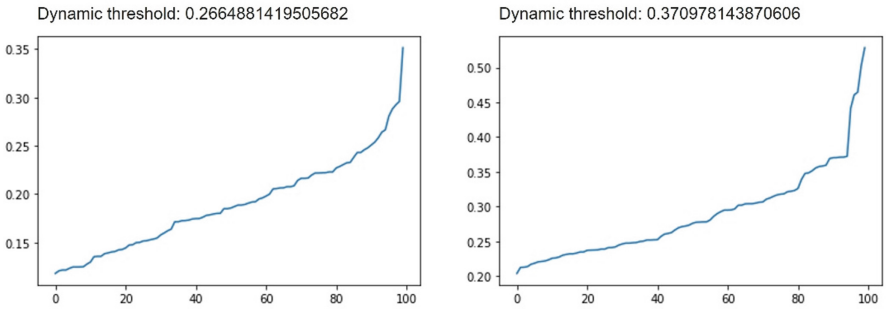

Fig. 2. Dynamic threshold calculation

Evaluating the performance of anomaly detection algorithms is important to understand how well they are able to detect anomalies in a given dataset. Here are the used performance metrics for anomaly detection [29–31]:

1. Accuracy: The percentage of correctly classified data points as normal or anomalous.
2. RMSE: It measures the average distance between the predicted and actual values in a dataset taking into account the scale of the values.
3. Precision: The proportion of true positive predictions out of all positive predictions.
4. Recall (Sensitivity): The proportion of true positive predictions out of all actual positives
5. F-Score: The harmonic mean of precision and recall.
6. AUC: A metric that measures the performance of the model across different thresholds for labeling data as normal or anomalous. It is calculated as the area under the ROC curve, which plots the true positive rate (sensitivity) against the false positive rate (1-specificity) at different threshold values. A higher AUC indicates better performance.

Table 1 shows the classification accuracy and performance metrics of five different classifiers: CTRC, One Class SVM, Long Short-Term Memory (LSTM), Isolation Forest and k-NN.

Table 1. Classification Accuracy and Performance Metrics

Classifier	Accuracy	RMSE	Precision	Recall	F-Score	AUC
CTRC	97.89%	0.0821	0.980	0.980	0.978	0.9887
One Class SVM	93.66%	0.0912	0.937	0.936	0.937	0.9752
LSTM	93.17%	0.0932	0.932	0.933	0.933	0.9703
Isolation Forest	91.38%	0.1007	0.914	0.914	0.913	0.9588
k-NN	87.99%	0.1185	0.880	0.880	0.880	0.9502

In the above Table 1, it is shown that the CTRC has the highest accuracy (97.89%) and AUC (0.9887), as well as the highest precision, recall, and F-score. One Class

SVM, LSTM, Isolation Forest and k-NN also have relatively high accuracy, but their performance metrics are lower than those of the CTRC. In terms of RMSE, the CTRC has the lowest value (0.0821), indicating that its predictions are on average closer to the actual values than the other classifiers. The Precision, Recall, and F-score for CTRC are also relatively high, indicating that it is able to correctly identify positive cases while minimizing false positives.

The CTRC model as a type of Recurrent Neural Network has shown to be effective for time series forecasting and classification tasks. Specifically, CTRC can be trained easily, using a single forward pass through the reservoir, making it computationally efficient and faster to train than other recurrent neural networks such as LSTM networks. This efficiency is beneficial for real-time applications where the model needs to make predictions quickly. In addition is robust to noise in the input data and to perform well even when the input data is corrupted or contains missing values. This robustness is useful in real-world cybersecurity applications where the data may not be perfectly clean. Moreover, CTRC is relatively easy to implement and does not require complex optimization algorithms or hyperparameter tuning. This make it more accessible to researchers and practitioners who may not have extensive experience with machine learning. Finally, it adapts to changing input data over time due to the differential equation, allowing it to continuously learn and improve its predictions based on Q-learning algorithm. This adaptability is extremely useful in the cyber defense applications where the input data changes frequently or where there are multiple sources of variability.

Overall, the CTRC model's efficiency, robustness, ease of implementation, and adaptability make it a promising approach for time series classification and forecasting tasks, and it may outperform other classifiers in certain scenarios. In conclusion, based on the obtained values of the performance indices and considering the objective difficulties raised in this research, the proposed model has been proven very efficient, able to cope with complex situations and to recognize anomalies.

4 Discussion and Conclusions

This research presents a next-generation anomaly detection model for cyber defence where the reservoir neurons are modelled as continuous-time dynamical systems. The CTRC system uses the dynamics of a high-dimensional, randomly connected network to process and learn from complicated input signals. In particular, CTRC expands the constantly operating reservoir computing paradigm. It provides discretization-free real-time processing of input signals, and the Q-Learning method is used to optimize the system parameters. The fundamental goal of CTRC is to convert the time-series data input into a high-dimensional representation that can be applied to tasks involving classification or prediction. The input is transformed through the reservoir, a fixed network of nonlinear dynamical systems with random initialization. Differential equations with a continuous time are used to describe reservoir dynamics. The output layer is trained to distinguish between normal (i.e., non-anomalous) and abnormal data to apply CTRC for anomaly detection.

The proposed approach is extremely helpful when conventional rule-based detection techniques fall short, such as zero-day assaults or other sophisticated, persistent threats.

It also has several advantages over other machine learning methods, such as a quick and easy training procedure and the ability to interpret time-series data effectively.

While Continuous-Time Reservoir Computing (CTRC) has shown promising results, there are two specific limitations to consider in future research using this model:

1. Model Complexity: CTRC can have a high model complexity due to many reservoir nodes and recurrent connections. This can make it difficult to interpret and analyze the model and make it prone to overfitting.
2. Limited Memory: The reservoir in CTRC has a limited memory capacity, which means that it may struggle to process long-term dependencies in data. This makes it less suitable for applications with important long-term patterns or trends.

References

1. Bala, A., Ismail, I., Ibrahim, R., Sait, S.M.: Applications of metaheuristics in reservoir computing techniques: a review. IEEE Access **6**, 58012–58029 (2018). https://doi.org/10.1109/ACCESS.2018.2873770
2. Demertzis, K., Iliadis, L., Pimenidis, E.: Geo-AI to aid disaster response by memory-augmented deep reservoir computing. Integr. Comput.-Aided Eng. **28**(4), 383–398 (2021). https://doi.org/10.3233/ICA-210657
3. Freiberger, M., Katumba, A., Bienstman, P., Dambre, J.: Training passive photonic reservoirs with integrated optical readout. IEEE Trans. Neural Netw. Learn. Syst. **30**(7), 1943–1953 (2019). https://doi.org/10.1109/TNNLS.2018.2874571
4. Li, S., Pachnicke, S.: Photonic reservoir computing in optical transmission systems. In: 2020 IEEE Photonics Society Summer Topicals Meeting Series (SUM), pp. 1–2 (2020). https://doi.org/10.1109/SUM48678.2020.9161045
5. Gauthier, D.J., Bollt, E., Griffith, A., Barbosa, W.A.S.: Next generation reservoir computing. Nat. Commun. **12**(1), Art. no. 1 (2021). https://doi.org/10.1038/s41467-021-25801-2
6. Demertzis, K., Kikiras, P., Tziritas, N., Sanchez, S.L., Iliadis, L.: The next generation cognitive security operations center: network flow forensics using cybersecurity intelligence. Big Data Cogn. Comput. **2**(4), Art. no. 4 (2018). https://doi.org/10.3390/bdcc2040035
7. Hart, A.: Generalised Synchronisation for Continuous Time Reservoir Computers. Rochester, NY (2021). https://doi.org/10.2139/ssrn.3987856
8. Smith, L.M., Kim, J.Z., Lu, Z., Bassett, D.S.: Learning continuous chaotic attractors with a reservoir computer. Chaos Interdiscip. J. Nonlinear Sci. **32**(1), 011101 (2022). https://doi.org/10.1063/5.0075572
9. Abu, U.A., Folly, K.A., Jayawardene, I., Venayagamoorthy, G.K.: Echo state network (ESN) based generator speed prediction of wide area signals in a multimachine power system. In: 2020 International SAUPEC/RobMech/PRASA Conference, pp. 1–5 (2020). https://doi.org/10.1109/SAUPEC/RobMech/PRASA48453.2020.9041236
10. Duport, F., Smerieri, A., Akrout, A., Haelterman, M., Massar, S.: Fully analogue photonic reservoir computer. Sci. Rep. **6**(1), Art. no. 1 (2016). https://doi.org/10.1038/srep22381
11. Manjunath, G.: An echo state network imparts a curve fitting. IEEE Trans. Neural Netw. Learn. Syst. **33**(6), 2596–2604 (2022). https://doi.org/10.1109/TNNLS.2021.3099091
12. Wang, Z., Yao, X., Huang, Z., Liu, L.: Deep echo state network with multiple adaptive reservoirs for time series prediction. IEEE Trans. Cogn. Dev. Syst. **13**(3), 693–704 (2021). https://doi.org/10.1109/TCDS.2021.3062177

13. Whiteaker, B., Gerstoft, P.: Memory in echo state networks and the controllability matrix rank. In: ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 3948–3952 (2022). https://doi.org/10.1109/ICASSP43922.2022.9746766
14. Shao, Y., Yao, X., Wang, G., Cao, S.: A new improved echo state network with multiple output layers for time series prediction. In: 2021 6th International Conference on Robotics and Automation Engineering (ICRAE), pp. 7–11 (2021). https://doi.org/10.1109/ICRAE53653.2021.9657812
15. Li, X., Bi, F., Yang, X., Bi, X.: An echo state network with improved topology for time series prediction. IEEE Sens. J. **22**(6), 5869–5878 (2022). https://doi.org/10.1109/JSEN.2022.3148742
16. Kidger, P.: On Neural Differential Equations. arXiv (2022). https://doi.org/10.48550/arXiv.2202.02435
17. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J. Comput. Phys. **378**, 686–707 (2019). https://doi.org/10.1016/j.jcp.2018.10.045
18. Demertzis, K., Iliadis, L.: Adaptive elitist differential evolution extreme learning machines on big data: intelligent recognition of invasive species. In: Angelov, P., Manolopoulos, Y., Iliadis, L., Roy, A., Vellasco, M. (eds.) INNS 2016. AISC, vol. 529, pp. 333–345. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-47898-2_34
19. Bai, Z., Pang, H., Liu, M., Wang, M.: An improved Q-Learning algorithm and its application to the optimized path planning for unmanned ground robot with obstacle avoidance. In: 2022 6th CAA International Conference on Vehicular Control and Intelligence (CVCI), pp. 1–6 (2022). https://doi.org/10.1109/CVCI56766.2022.9964859
20. Huang, D., Zhu, H., Lin, X., Wang, L.: Application of massive parallel computation based Q-learning in system control. In: 2022 5th International Conference on Pattern Recognition and Artificial Intelligence (PRAI), pp. 1–5 (2022). https://doi.org/10.1109/PRAI55851.2022.9904213
21. Yin, Z., Cao, W., Song, T., Yang, X., Zhang, T.: Reinforcement learning path planning based on step batch Q-learning algorithm. In: 2022 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA), pp. 630–633 (2022). https://doi.org/10.1109/ICAICA54878.2022.9844553
22. Chouiekh, C., Yahyaouy, A., Aarab, A., Sabri, A.: Road traffic: deep q-learning agent control traffic lights in the intersection. In: 2022 International Conference on Intelligent Systems and Computer Vision (ISCV), pp. 1–5 (2022). https://doi.org/10.1109/ISCV54655.2022.9806135
23. InfluxDB Times Series Data Platform. InfluxData (2022). https://www.influxdata.com/home/. Accessed 28 Feb 2023
24. Industrial IoT (IIoT) solutions for smart industries – Factry. Factry - Open Manufacturing Intelligence. https://www.factry.io/. Accessed 28 Feb 2023
25. Nguyen, Q.-D., Dhouib, S., Chanet, J.-P., Bellot, P.: Towards a web-of-things approach for OPC UA field device discovery in the industrial IoT. In: 2022 IEEE 18th International Conference on Factory Communication Systems (WFCS), pp. 1–4 (2022). https://doi.org/10.1109/WFCS53837.2022.9779181
26. Demertzis, K., Iliadis, L.S., Anezakis, V.-D.: An innovative soft computing system for smart energy grids cybersecurity. Adv. Build. Energy Res. **12**(1), 3–24 (2018). https://doi.org/10.1080/17512549.2017.1325401
27. Wang, H., Wang, Y., Wan, S.: A density-based clustering algorithm for uncertain data. In: 2012 International Conference on Computer Science and Electronics Engineering, vol. 3, pp. 102–105 (2012). https://doi.org/10.1109/ICCSEE.2012.91

28. Khan, M.M.R., Siddique, M.A.B., Arif, R.B., Oishe, M.R.: ADBSCAN: adaptive density-based spatial clustering of applications with noise for identifying clusters with varying densities. In: 2018 4th International Conference on Electrical Engineering and Information & Communication Technology (iCEEiCT), pp. 107–111 (2018). https://doi.org/10.1109/CEEICT.2018.8628138
29. Botchkarev, A.: Performance metrics (error measures) in machine learning regression, forecasting and prognostics: properties and typology. Interdiscip. J. Inf. Knowl. Manag. **14**, 045–076 (2019). https://doi.org/10.28945/4184
30. Koyejo, O.O., Natarajan, N., Ravikumar, P.K., Dhillon, I.S.: Consistent binary classification with generalized performance metrics. In: Advances in Neural Information Processing Systems, vol. 27 (2014). https://papers.nips.cc/paper/2014/hash/30c8e1ca872524fbf7ea5c519ca397ee-Abstract.html. Accessed 24 Oct 2021
31. Liu, Y., Zhou, Y., Wen, S., Tang, C.: A strategy on selecting performance metrics for classifier evaluation. Int. J. Mob. Comput. Multimed. Commun. IJMCMC **6**(4), 20–35 (2014). https://doi.org/10.4018/IJMCMC.2014100102