# An Autonomous Self-learning and Self-adversarial Training Neural Architecture for Intelligent and Resilient Cyber Security Systems

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

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

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

**Abstract.** Cybersecurity systems have become increasingly important as businesses and individuals rely more on technology. However, the increasing complexity of these systems and the evolving nature of cyber threats require innovative solutions to protect against cyber attacks. One promising approach is the idea of autonomous self-learning and auto-training neural architectures. Autonomous self-learning refers to the ability of the system to adapt to new threats and learn from past experiences without human intervention. Auto-training, on the other hand, refers to the ability of the system to improve its performance over time by automatically adjusting its parameters and algorithms. This research proposes an autonomous Self-Learning and Self-Adversarial Training (SLSAT) neural architecture for intelligent and resilient cyber security systems. It is an extension of the next-generation Continuous-Time Reservoir Computing (CTRC) that was proposed by the authors recently. The CTRC is a time-series anomaly detection system controlled by time-varying differential equations. It uses Reinforcement Learning (RL) to dynamically fine-tune the reservoir computing parameters in order to identify the aberrant changes in the data. The proposed method in this research improves the CTRC's architecture by including a Conditional Tabular Generative Adversarial Network (CTGAN). Specifically, including CTGAN allows the SLSAT architecture to generate synthetic data based on the identified abnormalities to improve the model's performance and adapt to new and evolving threats without manual intervention. This, as proved experimentally, helps the model identify aberrant changes in the data and fend off poison and zero-day attacks.

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


L. Iliadis et al. (Eds.): EANN 2023, CCIS 1826, pp. 461–478, 2023.
https://doi.org/10.1007/978-3-031-34204-2_38

## 1 Introduction

Cyber security is a constantly evolving field, and the threat landscape is constantly changing. Attackers are becoming more sophisticated and are constantly developing new techniques and strategies to bypass traditional security measures [1]. This means the cybersecurity industry must continually innovate and develop more advanced and proactive security solutions to stay ahead of attackers. One approach that is gaining popularity in the cybersecurity industry is using artificial intelligence and machine learning [2]. Machine learning algorithms can be trained to detect anomalies in network traffic and identify behavior patterns indicative of an attack. These algorithms can be trained using large network traffic and attack data datasets and continuously updated to adapt to new attack techniques. The cybersecurity industry needs to take a proactive security approach rather than relying on reactive measures. By using advanced technologies such as artificial intelligence and machine learning [3], sharing threat intelligence, and collaborating with other organizations, the industry can stay ahead of the evolving threat landscape and better protect against cyber-attacks [4, 5].

The proposed approach is an extension of the next-generation Continuous-Time Reservoir Computing (CTRC) that was proposed by the authors recently [6, 7]. Reservoir computing is a machine learning algorithm that uses the dynamics of a high-dimensional, randomly connected network to process and learn from input signals. It is an efficient and powerful technique for solving complex machine-learning tasks, particularly those involving time-series data. At the heart of the system is a reservoir, which is a randomly connected network of nodes. The input signal is fed into the reservoir, and the state vector is fed into a readout layer, which is trained using linear regression or another simple learning algorithm. Next-generation automated RC is an emerging technology that has the potential to enhance cyber defense greatly, using the basic principles of RC to analyze network traffic and identify anomalies indicative of cyberattacks or other security threats [8, 9].

The CTRC 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 a CTRC system, the input signal drives the reservoir dynamics, producing an output signal. The continuous-time nature of the system enables the reservoir to process input signals in real-time without discretization, which is particularly useful in applications where the input signal is a continuous data stream. However, it can be more difficult to train and optimize than discrete-time RC due to the complexity of the reservoir dynamics [8, 10]. The proposed CTRC's system parameters are optimized using the RL method to overcome these challenges.

This paper extends the above architecture and proposes an autonomous self-healing neural architecture for cyber security that leverages advanced machine learning techniques to detect, respond to, and mitigate cyber threats automatically. Specifically, the proposed method enhances the architecture by adding a CTGAN [11] to the CTRC. It creates a model that can withstand poison and zero-day attacks by enhancing the network's capacity for self-learning and self-training based on the identified abnormalities. The STSAT refers to the system's ability to detect and respond to security incidents without human intervention. This is important in cyber security, where threats can emerge and spread quickly, making it difficult for human operators to respond quickly.

## 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 CTRC system's architecture comprises input, reservoir, and output layers. In RC, the connection and input weights are assigned at random. The Echo State Property (ESP) [12, 13], the condition in which the reservoir is an "echo" of the complete input history, is ensured by scaling the reservoir weights in a fashion that does just that. The input u(n) and output y(n) discrete layers of the CTRC follow the problem's definitions. The number of hidden layers is grouped in an RC zone [14, 15]. The amount to which the RC's neurons, x(n), are coupled defines how sparse the RC will be.

The CTRC system can record temporal and spatial patterns of the network's data thanks to differential equations $dx1$ and $dx2$, which control how state vectors $x1(t)$ and $x2(t)$ behave as follows [16, 17]:

$$dx1/dt = -x1(t) + f1(W1x1(t) + \text{Win}\,1u(t))$$
$$dx2/dt = -x2(t) + f2\big(W2x2(t) + \text{Win}\,2u(t) + Vx1(t)\big)$$

While $x2(t)$ represents the state vector that captures the spatial patterns of the data, $x1(t)$ represents the state vector that captures the temporal patterns of the network traffic data in equations $dx1$ and $dx2$, respectively. The state vectors $x1(t)$ and $x2(t)$ change over time in response to the input signal u(t) and the reservoir's current state, as shown by the differential equations above. A depiction of the CTRC architecture is presented in Fig. 1.

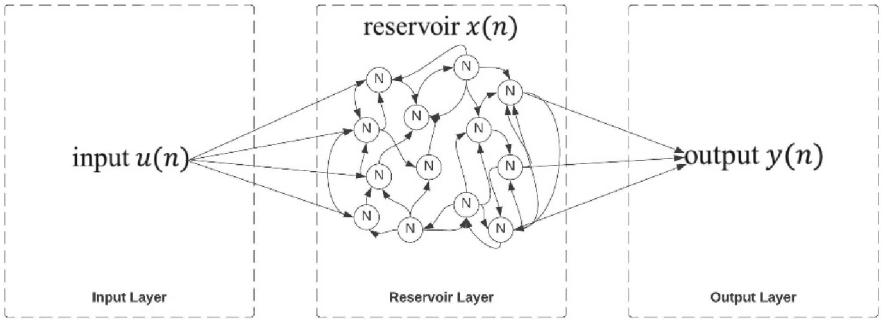


**Fig. 1.** CTRC architecture

In order to model the drift phenomenon, identify the abnormal changes in the data, and adaptively stabilize the learning system, the weight matrices W1, W2, Win1, Win2, and V are optimized using the Q-Learning algorithm [18, 19] to minimize the difference between the predicted output of the system and the true labels in a training dataset. The state space is defined in the first step of the Q-learning algorithm. The state space records pertinent data about the cyber defense scenario's current network traffic analysis. The Q-learning algorithm's action space is then based on the potential steps the agent could take to counteract cyberattacks. The agent's performance in the cyber defense task is

then used as the basis for the reward function for the Q-learning algorithm. The reward function motivates the agent to counter cyberattacks effectively and dissuades ineffective or harmful behavior. The Q-table is a lookup table associating expected rewards with states and actions. The Q-table is updated using the Q-learning algorithm to choose actions iteratively based on the current state and the values in the Q-table. The Bellman equation calculates the expected future reward from the present state and action and is used to update the Q-table [20]. The Q-table modifies the weight matrices and bias terms of the CTRC system. The Q-table estimates train the CTRC system to optimize the expected future reward. A test dataset implements the trained CTRC system's performance evaluation process. Based on the evaluation outcomes, the parameters of the CTRC algorithm are adjusted. This entails automatically modifying the group of parameters that the Q-learning process optimized.

In order to increase the CTRC's capacity for self-learning and self-training and create a model that can thwart poison and zero-day attacks, the proposed method in this research incorporates a CTGAN into the CTRC's architecture. CTGAN is a Generative Adversarial Network (GAN) for generating synthetic tabular data. GAN is a type of deep learning algorithm used to generate new data by learning the patterns and features in a given dataset. GANs are composed of two main components: a generator and a discriminator. The generator inputs random noise and produces a fake sample that resembles the original data. The discriminator, on the other hand, is trained to distinguish between real and fake samples. The generator is trained to fool the discriminator by producing samples that are indistinguishable from the real ones. The discriminator, in turn, is trained to identify whether a sample is real or fake correctly.

The training process involves alternating between training the generator to produce better fake samples and training the discriminator to better distinguish between real and fake samples. This process continues until the generator can produce samples that are difficult to distinguish from the real ones. Its objective is to learn the underlying probability distribution of the input data and then generate new samples that closely resemble the original data. The CTGAN consists of a generator and discriminator networks, similar to GANs. However, CTGAN is conditioned on the values of a subset of the input features. The generator network takes both a random noise vector and the conditioned input features as input. The objective function of CTGAN can be expressed as follows [11, 21]:

$$\begin{aligned} &min_G max_D V(D,G) \\ &= Ex{\sim}pdata(x)[log\ D(x|condition) \\ &+ Ez{\sim}pnoise(z), c{\sim}pcondition(c)\ [log(1 - D(G(z,c)|c))] \end{aligned}$$

where, $G$ is the generator network, $D$ is the discriminator network, $x$ is a real sample from the input data distribution *pdata*, $z$ is a noise vector sampled from a prior distribution *pnoise*, c is a conditioned subset of the input features sampled from the conditioning distribution *pcondition*, *condition* is a function that maps the input data $x$ to the conditioned subset of input features $c$, $D(x|condition)$ is the discriminator's probability of assigning a real sample $x$ a score of 1 (indicating that it is real) given the conditioning features condition, $G(z, c)$ is the generator's output, which is a synthetic sample generated by the generator network, conditioned on the input features $c$ and $1 - D(G(z, c)|c)$ is the discriminator's probability of assigning a synthetic sample $G(z, c)$ a score of 1

(indicating that it is real), given the conditioning features $c$. The objective function is optimized iteratively between the generator and discriminator networks until the generator produces synthetic samples that are indistinguishable from real samples according to the discriminator [11, 22]. A depiction of the CTGAN architecture is presented in Fig. 2.

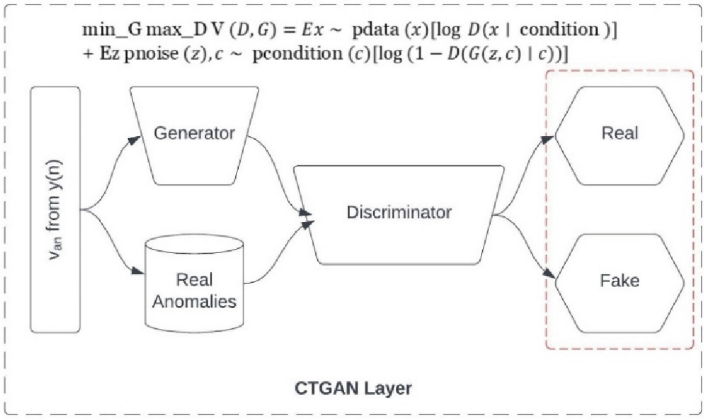


**Fig. 2.** CTGAN architecture

Because there is no sufficient defence mechanism capable of fully protecting against novel attacks, as these lie in the ingenuity of the attackers, this process extends the so far techniques in a prototype way, protecting not only against zero-day but also against poison attacks. Including CTGAN allows the SLSAT architecture to generate synthetic data based on the identified abnormalities. Specifically, as the adversary wants to find the least possible $x^{adv}$ that is closest to $x$ to generate a contradictory sample, the proposed mechanism pushes this $x^{adv}$ as far away from the legitimate sample as possible. In particular, during the training of this particular neural architecture, it learns to successfully recognize samples that belong to the distribution of the training data. Still, it recognizes large changes in its outputs for samples at a short distance but outside the specific distribution. By introducing to the training set patterns that are a linear combination of the original patterns, the neural network learns to recognize contradictory disturbances [23]. Specifically for $\lambda \in [0, 1]$ and $x_i, x_j$ two instances of the training set, introducing the case $x_{new}$ to the training set for which applies [22, 24, 25]:

$$x_{new} = \lambda x_i + (1 - \lambda)x_j \text{ και } f(x_{new}) = \lambda f(x_i) + (1 - \lambda)f(x_j)$$

Thus the training set acquires a more generalized distribution, with the result that the network generalizes better and does not have large changes in points of the input space outside the distribution of the original training set. The method improves the network's performance even on datasets where you expect the classification function to exhibit significant non-linearities.

The defense method is based on the observation that the aggressive cases do not belong to the distribution field to which the input data belongs. At the same time, they

are closer to the subfield to which the cases of their true class belong. Considering that the outputs of the last layers of the neural network are feature vectors that are entered as inputs to the network, the distribution to which the feature vectors that result as outputs of the neural network belong, when real data is present as input, is calculated. Specifically, suppose $Y_c$ is the set of training vectors belonging to class $c$. In that case, $y$ is the vector of input features, calculating $\hat{f}_c(y)$, which is the estimate of the density of the distribution of the real features of class $c$ at point $y$ as follows [26, 27]:

$$\hat{f}_c(y) = \frac{1}{|Y_c|} \sum_{y_i \in Y_c} exp\left(\frac{-\|y - y_i\|_2^2}{\sigma^2}\right)$$

where with $|Y_c|$ the number of elements of the set $Y_c$.

According to this method, an aggressive input with real class $c_1$, which is recognized as class $c_2$, will hold that $\hat{f}_{c_1}(x) > \hat{f}_{c_2}(x)$. Extending this consideration, Bayesian uncertainty can be extracted from a neural network, which has been trained using the dropout method, so that an input $x$ receives the outputs $y_1, y_2, \ldots, y_T$ for $T$ different sets of parameters of the network. The uncertainty $U(x)$ of the network at point $x$ is calculated from the equation [28, 29]:

$$U(x) = \frac{1}{T} \sum_{i=1}^{T} y_i^T y_i - \left(\frac{1}{T} \sum_{i=1}^{T} y_i\right)^T \left(\frac{1}{T} \sum_{i=1}^{T} y_i\right)$$

Given the assumption that aggressive inputs appear in regions of the network with high uncertainty, $U(x)$ is a useful metric for determining whether an input $x$ is aggressive. A depiction of the proposed architecture is presented in Fig. 3 (Appendix 1).

## 3 Dataset and Results

Factry.io and InfluxDB were used to provide a perfect simulation environment [30, 31]. 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. InfluxDB is a time-series database designed to handle high volumes of time-stamped data. It supports various data types and formats, including numerical, string, and Boolean data, and provides a SQL-like query language to access and manipulate data. The goal was to gather data about the industrial environment using the open-source OPC-UA collector protocol [32].

Data for one year was gathered from three sensors' hourly quantifiable values within a machine condition that runs continuously. There is a tank specifically for raw water storage, and a valve opens when the sensor detects a level of less than or equal to 0.5 m. This research suggests a trustworthy heuristic approach of selection, based only on assessment criteria, to identify an ideal threshold for binary class separation (normal or abnormal). The proposed algorithm calculates the density around each data point 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 extra data points are added to the center of the regions if they are densely accessible. 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 [33, 34]:

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

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

$$\sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2}$$

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

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

Two plots to visualize the dynamic threshold calculation depicted in the following Fig. 4.

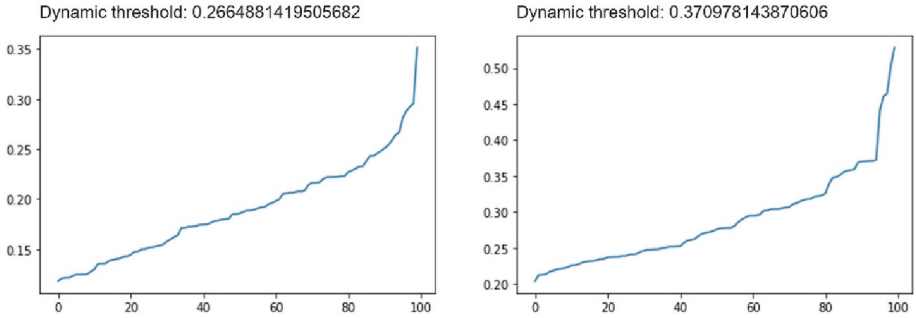


**Fig. 4.** Dynamic threshold calculation

Samples (outliers) are considered abnormal when the anomaly score departs from the expected behaviour by applying the dynamic threshold.

In order to easily make a comparison between the real and synthetic data, a visual evaluation method is used to generate a plot that allows comparing the distributions of the datasets visually. The plot consists of two panels, one for the real dataset and one for the synthetic dataset. Each panel shows the dataset's values distribution using a kernel density estimate (KDE) plot [11]. The Fig. 5 shows a dataset's absolute log mean and standard deviation of each numeric column.

The KDE plot shows the probability density function of the data, which represents the relative frequency of values in each interval of the range of the variable. The x-axis of the plot represents the values of the variable, and the y-axis represents the probability density of those values. The more similar the distributions of the real and synthetic datasets are, the more the two KDE plots will overlap. If the two datasets are very similar, the two KDE plots will overlap significantly. If the two datasets have very different distributions, the two KDE plots will not overlap much. By comparing the KDE plots for the real and synthetic datasets, one can understand how similar the two datasets are in their statistical

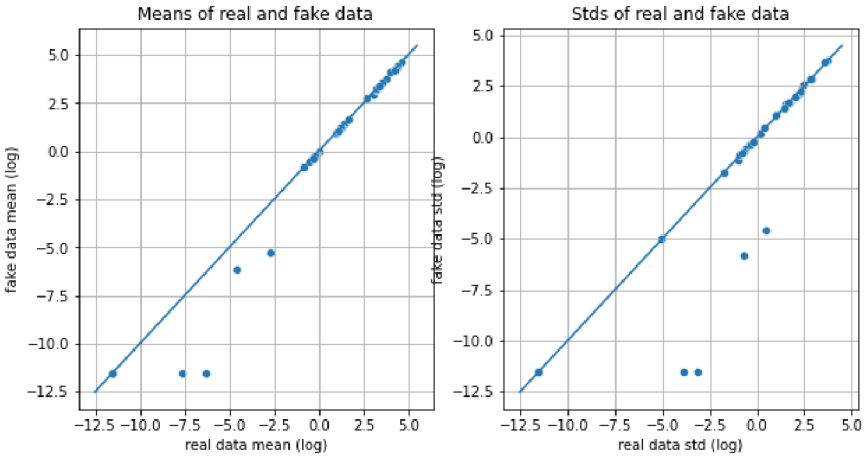


**Fig. 5.** Absolute Log Mean and STDs of real and fake data

properties. If the KDE plots are very similar, it suggests that the synthetic dataset has been generated successfully and has similar statistical properties to the real dataset. If the KDE plots are very different, it suggests that the synthetic dataset does not represent the real dataset well may not be suitable for the intended use.

Specifically, the blue line in the left plot represents the absolute log of the mean for each column. The mean is a measure of central tendency representing the average value of the data in that column. Taking the absolute value of the log of the mean ensures that we are looking at differences in magnitude rather than direction, making it easier to compare the means of columns with different scales. The blue line in the right plot represents the absolute log of the standard deviation for each column. The standard deviation is a measure of variability that represents the spread out of the data in each column. Taking the absolute value of the log of the standard deviation ensures that we are looking at differences in magnitude rather than direction and makes it easier to compare the standard deviations of columns with different scales. By looking at the plot, we can quickly identify columns with significantly different means or standard deviations. These columns may indicate outliers or other issues in the data that should be investigated further. In addition, columns with very low or zero values may be problematic for some types of analysis, as their logarithms can be undefined or very large negative numbers. Overall, the plot provides a quick overview of the numeric columns in the dataset and can help identify potential issues or areas for further investigation [35]. Figure 6 shows the cumulative sum of a column over time from 8 features. The plot's x-axis shows the time of each value in the column, and the y-axis shows the cumulative sum of those values up to that point.

The blue line (real data) represents the current cumulative sum values of the column, and the orange line (fake data) represents the column's expected feature cumulative sum values. The expected features cumulative sum values are obtained by shifting the blue line forward by a certain number of time periods. By comparing the blue and orange lines, it is easy to see how the cumulative sum of the column is expected to change

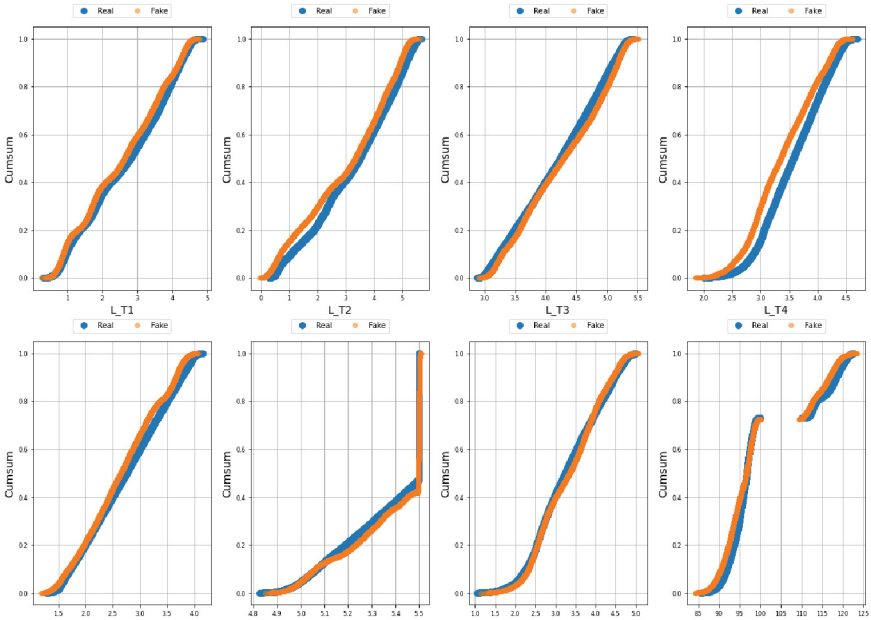


**Fig. 6.** Cumulative sums per feature (8 features)

over time. If the orange line is significantly higher than the blue line, it suggests that the cumulative sum of the column is expected to increase rapidly in the future. If the orange line is significantly lower than the blue line, it suggests that the cumulative sum of the column is expected to decrease rapidly in the future. The plot provides a way to visualize the trend of the data over time and how it is expected to change cumulatively in the future. It can be useful for predicting future trends or identifying patterns in the data that may be useful for making decisions [36]. Figure 7 shows the distribution per future time period.

The x-axis of each histogram shows the value of the column, and the y-axis shows the frequency of each value. The plot consists of multiple histograms, one for each feature time period. Each histogram shows the distribution of values in the column for that time period and provides a way to visualize how the distribution of values is expected to change over time. By comparing the histograms for different time periods, it is easy to see how the distribution of values in the column is expected to change over time. For example, if the histograms shift to the right over time, it suggests that the values in the column are expected to increase in the future. If the histograms shift to the left over time, it suggests that the values in the column are expected to decrease in the future. If the histograms remain relatively stable over time, it suggests that the values in the column are expected to remain relatively constant. The plot provides a way to visualize how the distribution of values in the column is expected to change over time, and can be useful for predicting future trends or identifying patterns in the data that may be useful for making decisions. It can also be used to identify potential outliers or other issues in the data that may affect its analysis [37].

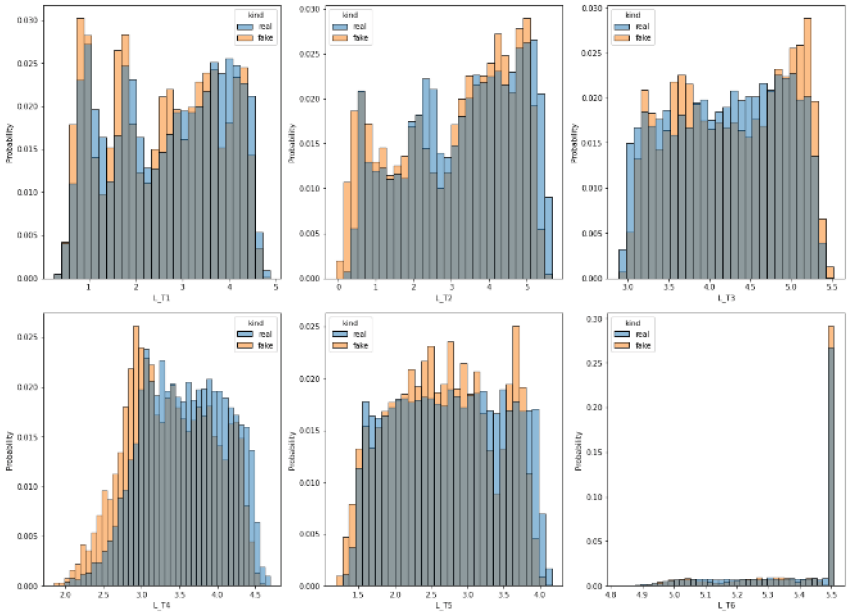


**Fig. 7.** Distribution per feature (6 features)

Figure 8 shows the correlation matrices of real and fake data and the differences between them.

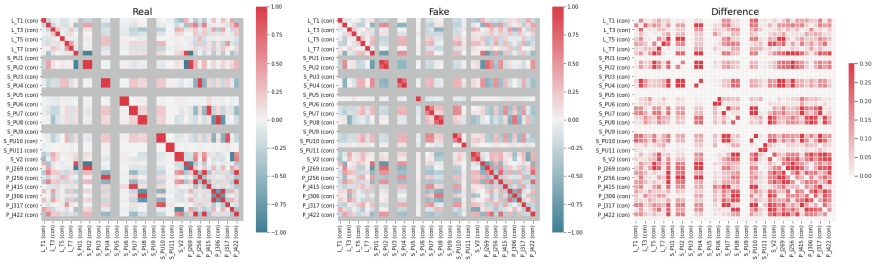


**Fig. 8.** Correlation matrices of real data, fake data and the differences

This plot concatenates the real and synthetic datasets and computes their difference. The x-axis and y-axis of each heatmap show the column names of the dataset, and each cell in the heatmap represents the correlation between a pair of columns. The color of each cell represents the strength and direction of the correlation, with blue indicating a negative correlation and red indicating a positive correlation. White cells indicate no correlation. The first heatmap shows the correlation matrix for the real data, allowing to see the correlation between pairs of columns in the real data. The second heatmap shows the correlation matrix for the synthetic data, allowing to see the correlation between pairs of columns in the synthetic data. Finally, the third heatmap shows the difference

in correlation matrices between the synthetic and real data, allowing to see where the synthetic data's correlation structure deviates from the real data's correlation structure. This can be useful for identifying areas where the synthetic data may not accurately represent the real data [11].

Finally, Fig. 9 shows a scatter plot of the first two principal components of a dataset using Principal Component Analysis (PCA).

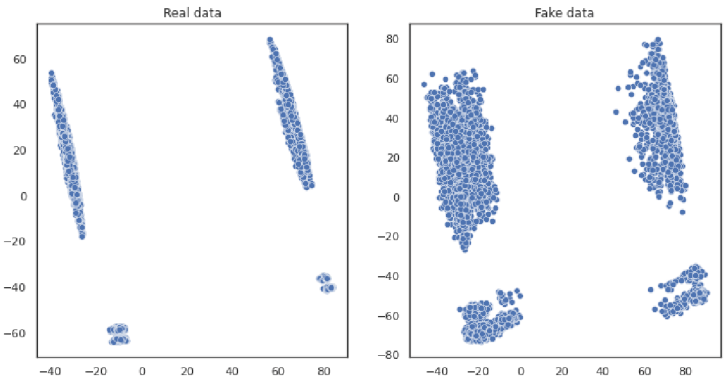


**Fig. 9.** First two components of PCA

The left plot represents the first principal component (PC1), and the right plot represents the second principal component (PC2). Each data point in the plot represents a row in the dataset, and its position on the plot represents its values in the first two principal component directions. The position of each data point on the plot is determined by the values of the data in the first two principal component directions, which are calculated by the PCA algorithm [38, 39]. The plot can be useful for identifying patterns and trends in the data, as well as for visualizing the similarity and differences between different data points. Data points that are close together on the plot are similar to each other in terms of their values in the first two principal component directions, while data points that are far apart are dissimilar.

The plot can also be used to identify potential outliers or other issues in the data that may affect its analysis. Outliers may appear as data points that are far away from the main cluster of data points on the plot and may be worth investigating further to determine if they represent genuine data points or errors in the data. In summary, the plot provides a useful visualization of the main directions of variation in the data and can be a useful tool for exploratory data analysis.

In order to perform analytical tests that will prove the value and capability of the proposed scheme, three datasets were created. The initial one includes only the real data where in this case, the last layer of the proposed architecture works, but it is not possible to evaluate its capabilities in detecting poisoning-induced anomalies. The evaluation performance metrics in all cases used to compare the anomaly detection algorithms are Accuracy, RMSE, Precision, Recall (Sensitivity), F-Score, and AUC [35, 36]. Tables 1, 2 and 3 shows the classification accuracy and performance metrics in real, fake and mix

datasets of six different classifiers: SLST, CTRC, One Class SVM, Long Short-Term Memory (LSTM), Isolation Forest and k-NN.

**Table 1.** Classification Accuracy and Performance Metrics in real data

| Classifier | Accuracy | RMSE | Precision | Recall | F-Score | AUC |
|---|---|---|---|---|---|---|
| SLSAT | 97.95% | 0.0819 | 0.980 | 0.985 | 0.984 | 0.9902 |
| 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 |

From the results of the above Table 1, it is evident that, there is little difference between the upgraded SLSAT scheme and the previous CTRC method, as the architecture remains the same. The slight increase in the classification accuracy of the proposed model may be related to the increase in the predictive ability of the algorithm based on the additional samples added to the dataset after finding some anomalies. In addition, the training processes' randomness which is used to calculate the density around each data point to identify the dynamic threshold. This is achieved by counting the number of points in a user-defined neighbourhood (Eps-Neighbourhood) with the definition of thresholds. This means that if the algorithm is run multiple times on the same dataset, it may produce slightly different accuracy scores due to this randomness.

**Table 2.** Classification Accuracy and Performance Metrics in fake data

| Classifier | Accuracy | RMSE | Precision | Recall | F-Score | AUC |
|---|---|---|---|---|---|---|
| SLSAT | 94.12% | 0.0903 | 0.941 | 0.940 | 0.941 | 0.9689 |
| CTRC | 83.71% | 0.1566 | 0.837 | 0.838 | 0.837 | 0.9124 |
| One Class SVM | 86.38% | 0.1207 | 0.865 | 0.865 | 0.870 | 0.9341 |
| LSTM | 82.97% | 0.1632 | 0.830 | 0.830 | 0.830 | 0.9108 |
| Isolation Forest | 80.26% | 0.1981 | 0.801 | 0.805 | 0.805 | 0.8894 |
| k-NN | 81.58% | 0.1873 | 0.816 | 0.816 | 0.816 | 0.8943 |

From the results of Table 2 above, it is evident that in this particular case, there is a significant difference between the proposed algorithm and the other methods, which mostly showed very low performance. This fact is obviously due to the inability of the other models to cope with the inability to find unknown patterns which, although very similar to the real ones, differ significantly. The fake data generated by CTGAN contain noise or uncertainty does not present in the real data. This can be due to several factors,

including the inherent stochasticity of the generative model, the use of random seeds, or other sources of variability in the training process. This makes it more difficult for the machine learning model to accurately distinguish between real and fake examples, especially if the noise is correlated with the target variable. This can lead to lower performance on the fake data than the real data. The proposed model uses self-learning and self-adversarial training to address this issue. The SLSAT model is self-training in order to distinguish between real and fake examples more accurately. This enhances the model's performance to recognize and handle the noise or uncertainty in the fake data and improves its overall performance on real and fake examples, as proved by results.

**Table 3.** Classification Accuracy and Performance Metrics in mix data

| Classifier | Accuracy | RMSE | Precision | Recall | F-Score | AUC |
|---|---|---|---|---|---|---|
| SLSAT | 99.05% | 0.0697 | 0.991 | 0.991 | 0.991 | 0.9938 |
| CTRC | 90.15% | 0.1123 | 0.900 | 0.905 | 0.905 | 0.9416 |
| One Class SVM | 89.81% | 0.1131 | 0.898 | 0.898 | 0.898 | 0.9409 |
| LSTM | 89.76% | 0.1136 | 0.898 | 0.897 | 0.897 | 0.9397 |
| Isolation Forest | 90.04% | 0.1127 | 0.905 | 0.905 | 0.905 | 0.9403 |
| k-NN | 84.12% | 0.1513 | 0.841 | 0.841 | 0.841 | 0.9073 |

The enormous superiority of the proposed system is confirmed in the mixed dataset. The table shows that the SLSAT classifier has the highest accuracy at 99.05% and the highest AUC at 0.9938. It also has high precision, recall, and F-score. The CTRC, One Class SVM, LSTM, and Isolation Forest classifiers have similar accuracies ranging from 89.76% to 90.15% and AUCs ranging from 0.9403 to 0.9416. The k-NN classifier has the lowest accuracy at 84.12% but still has a decent AUC of 0.9073. The reasons for this superiority could be attributed to the following:

1. Self-Learning Capability: SLSAT incorporates a CTGAN to enhance the model's self-learning capability. By generating synthetic data that mimics real data, SLSAT can improve its accuracy and generalization capabilities, which makes it better equipped to identify abnormal behavior in the data.
2. Self-Adversarial Training: The self-adversarial training approach in SLSAT allows the model to learn from attacks and adapt its defense strategy in real-time. This capability enables the model to detect and fend off zero-day attacks, a significant advantage in the current threat landscape.
3. Robustness to Poisoning Attacks: Poisoning attacks are a type of cyber-attack in which an attacker manipulates the training data to introduce biases or cause the model to make incorrect predictions. SLSAT's self-learning and self-adversarial training capabilities make it more robust to such attacks. By continuously learning from the data and adapting its defense strategy, SLSAT can detect and mitigate the effects of poisoning attacks.
4. Reinforcement Learning (RL): SLSAT employs RL to dynamically fine-tune the parameters of the Continuous-Time Reservoir Computing (CTRC) algorithm. This

allows SLSAT to optimize its performance continuously and adapt to changing conditions in the data.
5. Robustness: SLSAT is designed to be robust to noise and other perturbations in the data. By using CTRC, SLSAT can handle time-series data more effectively and efficiently, making it more robust than other algorithms.
6. Capacity for Self-Learning: Including CTGAN in the SLSAT architecture allows it to self-learn, meaning it can learn from data without supervision. This is particularly useful in cyber security applications, where anomalies can be hard to define or may change over time. SLSAT's ability to learn without supervision gives it an advantage over other algorithms that rely on labeled data.
7. Ability to Detect and Handle Complex Patterns: SLSAT's architecture, which includes CTRC and CTGAN, enables it to detect and handle complex patterns in the data. This is important in cyber security applications where anomalies may not be easily discernible or hidden within the noise of the data. SLSAT's ability to handle complex patterns gives it an advantage over other algorithms that may struggle to identify such anomalies.

These features enable the model to learn continuously, adapt to new threats, and perform well in various conditions, making it an effective tool for detecting and mitigating cyber threats.

## 4 Conclusion

The paper proposes an autonomous SLSAT neural architecture for intelligent and resilient cyber security systems. The proposed architecture extends the CTRC algorithm, incorporating a CTGAN to increase the network's capacity for self-learning and self-adversarial training. The proposed method allows for real-time adaptation to new and evolving cyber threats. The SLSAT model, as proved experimentally, outperforms other competitor algorithms in all performance metrics, including accuracy, RMSE, precision, recall, F-score, and AUC. The model's self-learning and self-adversarial approach enables it to detect and fend off zero-day and poison attacks, making it a valuable tool for next-generation cyber security applications [40].

Furthermore, the SLSAT architecture's capacity for self-learning and robustness to poisoning attacks make it a powerful tool for handling complex patterns in the data, which is crucial for detecting and mitigating advanced persistent cyber threats.

While the proposed SLSAT model has demonstrated superior performance in the current research, further development could be made in several areas:

1. Scalability: The current research evaluates the SLSAT model on a specific dataset. Future research could investigate the model's scalability and performance on larger datasets with a broader range of cyber threats.
2. Real-time Performance: The SLSAT model's ability to adapt to new and evolving threats in real time is a significant advantage. However, future research could further explore optimizing the model in streaming data performance.
3. Robustness: While the SLSAT model is designed to be robust to noise and other perturbations in the data, future research could further investigate ways to improve

its robustness. For instance, they are exploring new adversarial training techniques or enhancing the CTGAN's capacity to generate synthetic data that mimics rare events or evolving complex patterns in the data.

4. Explainability: While the SLSAT model's superior performance is clear, it is essential to understand how it arrives at its conclusions. Future research could investigate ways to make the model's decision-making process more transparent and interpretable, especially for regulatory compliance.
5. Deployment: The SLSAT model's real-world deployment raises several challenges, such as integrating it with existing cyber security infrastructure, managing the model's computational resources, and ensuring data privacy and security. Future research could address these deployment challenges to make the model more practical and useful in real-world applications.
6. The proposed SLSAT model has demonstrated excellent performance in cybersecurity applications. It is a significant contribution to the field of cyber defense, as it provides an intelligent and resilient solution for detecting and mitigating cyber threats in real time.

## Appendix 1

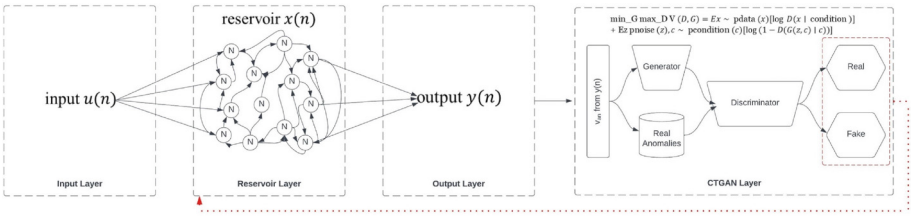


**Fig. 3.** The autonomous self-learning and self-adversarial training neural architecture

## References

1. Alhasan, S., Abdul-Salaam, G., Bayor, L., Oliver, K.: Intrusion detection system based on artificial immune system: a review. In: 2021 International Conference on Cyber Security and Internet of Things (ICSIoT), pp. 7–14, September 2021. https://doi.org/10.1109/ICSIoT55070.2021.00011
2. Elmrabit, N., Zhou, F., Li, F., Zhou, H.: Evaluation of machine learning algorithms for anomaly detection. In: 2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security), pp. 1–8, June 2020. https://doi.org/10.1109/CyberSecurity49315.2020.9138871
3. 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

4. Alromaihi, S., Elmedany, W., Balakrishna, C.: Cyber security challenges of deploying IoT in smart cities for healthcare applications. In: 2018 6th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW), pp. 140–145, December 2018. https://doi.org/10.1109/W-FiCloud.2018.00028.
5. Coulter, R., Han, Q.-L., Pan, L., Zhang, J., Xiang, Y.: Data-driven cyber security in perspective—intelligent traffic analysis. IEEE Trans. Cybern. **50**(7), 3081–3093 (2020). https://doi.org/10.1109/TCYB.2019.2940940
6. Hart, A.: Generalised synchronisation for continuous time reservoir computers. Rochester, NY, 17 December 2021. https://doi.org/10.2139/ssrn.3987856
7. 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
8. Cuchiero, C., Gonon, L., Grigoryeva, L., Ortega, J.-P., Teichmann, J.: Discrete-time signatures and randomness in reservoir computing. IEEE Trans. Neural Netw. Learn. Syst. **33**(11), 6321–6330 (2022). https://doi.org/10.1109/TNNLS.2021.3076777
9. 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
10. Al Jallad, K., Aljnidi, M., Desouki, M.S.: Anomaly detection optimization using big data and deep learning to reduce false-positive. J. Big Data **7**(1), 68 (2020). https://doi.org/10.1186/s40537-020-00346-1
11. Xu, L., Skoularidou, M., Cuesta-Infante, A., Veeramachaneni, K.: Modeling tabular data using conditional GAN. arXiv, 27 October 2019. https://doi.org/10.48550/arXiv.1907.00503
12. 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, January 2020. https://doi.org/10.1109/SAUPEC/RobMech/PRASA48453.2020.9041236
13. 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
14. 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
15. 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, February 2022. https://doi.org/10.1109/ICASSP43922.2022.9746766.
16. Kidger, P.: On neural differential equations. arXiv, 4 February 2022. https://doi.org/10.48550/arXiv.2202.02435
17. Raissi, M., Perdikaris, P., Karniadakis, G.E.: 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. Shi, Y., Rong, Z.: Analysis of Q-Learning like algorithms through evolutionary game dynamics. IEEE Trans. Circuits Syst. II Express Briefs **69**(5), 2463–2467 (2022). https://doi.org/10.1109/TCSII.2022.3161655
19. 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), June 2022, pp. 630–633. https://doi.org/10.1109/ICAICA54878.2022.9844553

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, December 2022. https://doi.org/10.1109/PRAI55851.2022.9904213
21. Habibi, O., Chemmakha, M., Lazaar, M.: Imbalanced tabular data modelization using CTGAN and machine learning to improve IoT Botnet attacks detection. Eng. Appl. Artif. Intell. **118**, 105669 (2023). https://doi.org/10.1016/j.engappai.2022.105669
22. Chauhan, R., Heydari, S.S.: Polymorphic adversarial DDoS attack on IDS using GAN. In: 2020 International Symposium on Networks, Computers and Communications (ISNCC), pp. 1–6, July 2020. https://doi.org/10.1109/ISNCC49221.2020.9297264
23. Demertzis, K., Tziritas, N., Kikiras, P., Sanchez, S.L., Iliadis, L.: The next generation cognitive security operations center: adaptive analytic lambda architecture for efficient defense against adversarial attacks. Big Data Cogn. Comput. **3**(1), Article no. 1, March 2019. https://doi.org/10.3390/bdcc3010006
24. Akhtar, N., Mian, A.: Threat of adversarial attacks on deep learning in computer vision: a survey. IEEE Access **6**, 14410–14430 (2018). https://doi.org/10.1109/ACCESS.2018.2807385
25. Demertzis, K., Iliadis, L., Kikiras, P.: A Lipschitz - shapley explainable defense methodology against adversarial attacks. In: Maglogiannis, I., Macintyre, J., Iliadis, L. (eds.) AIAI 2021. IAICT, vol. 628, pp. 211–227. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-79157-5_18
26. Dong, Y., et al.: Benchmarking adversarial robustness on image classification. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 318–328, June 2020. https://doi.org/10.1109/CVPR42600.2020.00040
27. Bousmalis, K., Silberman, N., Dohan, D., Erhan, D., Krishnan, D.: Unsupervised pixel-level domain adaptation with generative adversarial networks. arXiv, 23 August 2017. https://doi.org/10.48550/arXiv.1612.05424
28. Han, K., Li, Y., Xia, B.: A cascade model-aware generative adversarial example detection method. Tsinghua Sci. Technol. **26**(6), 800–812 (2021). https://doi.org/10.26599/TST.2020.9010038
29. Mahmood, K., Nguyen, P.H., Nguyen, L.M., Nguyen, T., Van Dijk, M.: Besting the Black-Box: barrier zones for adversarial example defense. IEEE Access **10**, 1451–1474 (2022). https://doi.org/10.1109/ACCESS.2021.3138966
30. InfluxDB Times Series Data Platform, InfluxData, 15 January 2022. https://www.influxdata.com/home/. Accessed 28 Feb 2023
31. Industrial IoT (IIoT) solutions for smart industries – Factry, Factry - Open Manufacturing Intelligence. https://www.factry.io/. Accessed 28 Feb 2023
32. 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, April 2022. https://doi.org/10.1109/WFCS53837.2022.9779181
33. 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, March 2012. https://doi.org/10.1109/ICCSEE.2012.91
34. Khan, M.M.R., Siddique, Md.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, September 2018. https://doi.org/10.1109/CEEICT.2018.8628138

35. 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
36. 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
37. 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
38. Li, X.: Fault data detection of traffic detector based on wavelet packet in the residual subspace associated with PCA. Appl. Sci. **9**(17), 3491 (2019). https://doi.org/10.3390/app9173491
39. Shamili, A.S., Bauckhage, C., Alpcan, T.: Malware detection on mobile devices using distributed machine learning. In: 2010 20th International Conference on Pattern Recognition, pp. 4348–4351, December 2010. https://doi.org/10.1109/ICPR.2010.1057
40. 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), Article no. 4, December 2018. https://doi.org/10.3390/bdcc2040035