Journal Article

A Deductive Approach for the Sensitivity Analysis of Software Defined Network Parameters

Sangodoyin, A. O., Akinsolu, M. O. and Awan, I.

Recommended citation:

Sangodoyin, A. O., Akinsolu, M. O. and Awan, I. (2020) 'A Deductive Approach for the Sensitivity Analysis of Software Defined Network Parameters', *Simulation Modelling Practice and Theory*, vol. 103, Sept 2020, pp. 102099 – 102114. doi: 10.1016/j.simpat.2020.102099

A Deductive Approach for the Sensitivity Analysis of Software Defined Network Parameters

Abimbola O. Sangodoyin[a,], Mobayode O. Akinsolu[b], Irfan Awan[a]

[a]*School of Electrical Engineering and Computer Science,University of Bradford, UK.*
[b]*Faculty of Arts, Science and Technology. Wrexham Glyndwr University, UK.*

Abstract

With the exponential growth in the number of internet-enabled devices, large scale security threats such as distributed denial of service (DDoS) attacks significantly affect software defined networks (SDNs). This necessitates efficient detection and mitigation solutions. Monitoring of SDN activities (typically identified using metrics such as throughput, jitter and response time) to ascertain deviations from profiles of normality (previously learned from benign traffic) is a key approach in detecting attacks on SDNs. In this paper, local sensitivity analysis (LSA) is implemented to identify the key network metrics that mainly influence the prediction of whether an SDN is under attack or secure. Using throughput, jitter and response time as the network impact metrics and a mathematical cost function based on min-max feature scaling to associate SDN scenarios with their respective SDN impact metrics, an artificial neural network (ANN)-based prediction model is built. The sensitivity of throughput, jitter and response time is then evaluated using the deviations of newly predicted target values of the ANN model from the actual target values when an additive white Gaussian noise (AWGN) is added to the respective impact metrics. The results of this study show that throughput, jitter and response time are all statistically sensitive to a DDoS flooding attack of the SDN. Also, Jitter was found to be the most sensitive network metric to a DDoS flooding attack of the SDN.

Keywords: DDoS; Network Security; Local Sensitivity Analysis; SDN

Email addresses: acadflint@yahoo.co.uk (Abimbola O. Sangodoyin), m.o.akinsolu@ieee.org (Mobayode O. Akinsolu)

1. Introduction

In the last decade, existing works in the literature and industrial collaboration on the subject of SDN implementation indicate high proliferation in its adoption. Realistic models and methodologies for understanding network traffic behaviour play an important role in facilitating efficient DDoS attack detection and mitigation[1].Using simulation data to describe dynamic network traffic characteristics and detect DDoS attacks is more reliable than traditional mathematical techniques[2]. Application of machine learning techniques to simulate network traffic characterisation has made it more realistic to mimic network traffic patterns and develop a robust models against security vulnerabilities for multiple reasons. A key advantage of this approach is the reduction of costly investment in data monitoring tools for day-to-day traffic analysis and performance overhead introduced.

Sensitivity analysis is an ad-hoc analysis that relies on historical data. Information gathered by network administrators and designers help in planning and responding to threat to input network parameters deemed sensitive. The full global analysis of all the historical network parameters gathered to monitor and detect DDoS attacks can be computationally expensive and may introduce a delay in end-to-end communication if implemented on an enterprise network. Therefore, there is a need to select a subset of parameters that are most probable to have strong effects in detecting an anomaly in network traffic.One-at-a-time local sensitivity analysis (LSA) technique analyses the impact of a single parameter on the cost function at a time, keeping the other parameters fixed. LSA is fast to compute. To the best of our knowledge, this work is a novel attempt to identify network parameters that are more sensitive in detecting DDoS attack in SDN using local sensitivity analysis.

The remainder of this paper is organised as follows: Section 2 discusses the related work, Section 3 provides an overview of sensitivity analysis, Section 4 discusses types of sensitivity analysis, Section 5 focuses on the application of ANN to sensitivity analysis, Section 6 provides a description of the dataset used, Section 7 reports on the experimental approach, results and discussion are presented in Section 8, and the concluding remarks are provided in Section 9.

2. Related Work

Application of sensitivity analysis to computer networks has become a popular research topic in recent years, especially in relation to security [3][4].

In [5], the authors examined using small sample computer models to make decisions and judgement in the face of uncertainty for model associated with risk assessment of disposal of radioactive waste. The work was further extended in [6] to determine the applicability of three widely used techniques to computer models having large uncertainties and varying degrees of complexity. Sensitivity analysis has also been applied to address computer networks availability in [7]. The authors implement parametric sensitivity analysis to compute effect of changes in the rate of constants of a Markov model on system dependability. Authors in [8] presents a method for computing network output sensitivity with respect to variations in the inputs for multilayer feedforward artificial neural network with different activation functions.

In terms of security, the authors in [9] performed sensitivity analysis on DARPA intrusion detection datasets and reported that 33 out of 41 features of the network traffic characteristics can be removed without causing great harm to the classification accuracy of DoS attacks and normal network traffic. Similarly, sensitivity analysis has been applied to attack pattern discovery in trusted routing scheme [10]. Using packet delivery ratio, normalised routing overhead, distrust threshold and trust update interval as performance metrics in different network conditions, the work carried out in [10] revealed that distrust threshold is more sensitive as compared to other metrics in optimising the detection rate of schemes employed. The authors in [4] explored the detection of bot in a compromised machine using dendritic cell algorithm (DCA). Their proposed algorithm and sensitivity analysis showed that incorporation of MAC value has significant effect on the detection of bot using DCA algorithm.

All the works mentioned above highlights the application of sensitivity analysis in identifying input parameters that significantly affect system response in designing attack detection algorithms. However, our approach differs from the ones mentioned in the following aspects. Firstly, the features of interest are extracted from emulated SDN environment with normal and DDoS attack traffic. Secondly, the implementation of local sensitivity analysis using artificial neural network to identify key network metrics that mainly influence the prediction of whether an SDN is under attack or secure.

3. Sensitivity Analysis

Anomaly detection techniques in networks rely on the assumption that variations in network parameters may have some effect on the state of network

performance when under attack. Several network features such as time-to-live (TTL), throughput, end-to-end delay, packet drops amongst others are considered input variables to assess the robustness of detection and to ensure appropriate mitigation technique is deployed. This approach is memory-intensive and can increase the computation time coupled with the purchase of high-performance computing device.

Sensitivity analysis involves the estimation of uncertainty in the output of a model as a result of different sources of uncertainty in the input[11]. Figure1 illustrates the basic representation of the relationship between input parameters and output response.

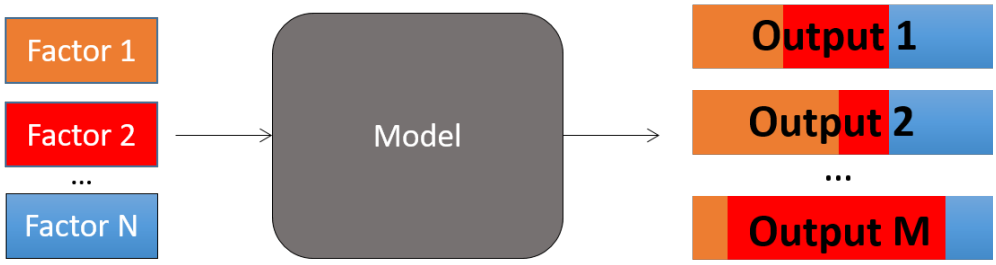

Figure 1: Relationship between input and output response in a sensitivity analysis

Sensitivity analysis offers an efficient approach to assess extent to which detection results are affected by changes in input network variables. In this context, sensitivity analysis is aimed at priority setting, to identify the key variables that are major influence in predicting whether the network is under attack or secure. As a result, sensitivity analysis provides an understanding of cause and effect reaction between changes in input variables and the corresponding output.

One of the key advantages of sensitivity analysis is that it identifies critical variables that may be given less consideration when designing a robust detection model.

4. Types of Sensitivity Analysis

When one or multiple inputs have relatively insignificant sensitivity as compared to others, the overall dimension of the neural network for training can be reduced by removing them and a smaller size neural network can be successfully retrained to develop a more efficient model.

- Local Sensitivity Analysis

Local sensitivity analysis (LSA) is the assessment of the local impact of input factors' variation on a model response by concentrating on the sensitivity in the vicinity of a set of factor values[12][13]. In LSA, the values of other input parameters are kept constant when studying how sensitive an input factor is.

LSA evaluates sensitivity for a single deterministic set of input parameters which is often based on the partial derivatives of the response with respect to the input parameters[14][15]. Given the model F defined as the following system:

$$\mathbf{y} = \mathbf{F}(\mathbf{x}, \gamma) \tag{1}$$

LSA indicates how independent variation x and parameters $\gamma = [\gamma_1, ..., \gamma_r]$ of F influence dependent variable $\mathbf{y}$.

The main concept of LSA is based on computation, after a training process of influence of pattern attributes x_i, i = 1,..., N or model's parameter γ on the output value y_j, j= 1,..., N, where N and J denote the number of features and outputs respectively [16][17]. This influence is characterised by real coefficients S_{ji}

$$S_{j,i}^{(p)} = \frac{\partial y_j(x_1^{(p)}, x_2^{(p)}, ..., x_N^{(p)})}{\partial x_i} \tag{2}$$

Equation 2 describes the sensitivity value of the jth neural network output signal on the ith attribute of the input vector $\mathbf{x}$, calculated based on the pth training pattern, p = 1,..., P.

LSA approach can be informative if there is little uncertainty in model input or if the inputs act linearly or additively [18]

- Global Sensitivity Analysis

Global Sensitivity Analysis (GSA)is the study of how uncertainty in the output of a model either numerical or otherwise can be apportioned to different sources of uncertainty in the model input [19].

GSA considers the impact of varying parameters simultaneously and uniformly over their full range of possible values [20][21]. GSA can show the relationship between multiple input parameters and cope well with a linear and non-linear response [22]. Unlike local sensitivity analysis, global sensitivity analysis requires more computational work and the approach is often probabilistic.

5. Artificial Neural Network Application to Sensitivity Analysis

The concept of artificial neural network (ANN) stems from an understanding of how neurons in the brain function to model a simple neural network using electrical circuits [23].

A key feature of ANN is its ability to learn. ANN employs three learning approaches namely: unsupervised learning, reinforcement learning and supervised learning. Any of this learning approaches can be combined or modified to produce a robust model that fits the generated data. ANN has found application in solving many different problems [24][25][26]. However,for known input to output process and rule-based systems, ANN becomes unreliable. Moreover, it might be detrimental if ANN algorithm attempts a better solution and begins to diverge from desired process. Some problems usually solved with ANN include classification, prediction, optimisation and pattern recognition. In intrusion detection and prevention system (IDPS), ANN can predict benign traffic from malicious traffic. Any form of prediction can be improved by learning from the data. These improvements and techniques depend on four major factors [27]: What data is to be improved? What prior information is available? What representation is used for the data? and What feedback is available to learn from?

5.1. Neural Network Training Algorithms

There are many different batch training algorithms that can be used to train a network. All have different characteristics and performance in terms of speed, precision, and memory requirement. Table 1 presents the lists of algorithm and the associated advantages and disadvantages.

Figure 2 illustrates the memory-speed comparison of neural network training algorithm. The gradient descent is the slowest training algorithm requiring less memory, while Levenberg-Marquardt is the fastest (but it require a lot of computational memory). For our experiment, we utilise the Levenberg-Marquardt training algorithm.The Levenberg-Marquardt training algorithm is regarded to be one of the most efficient training algorithms for ANNs [28]. It works by combining two algorithms (i.e., gradient descent method and the Gauss–Newton method) and as a result remedies their individual shortcomings [28][29]. Its major drawbacks are the increased computational cost due to the need to carry out Hessian matrix inversion calculation each time for weight updating and the storage of the Jacobian matrix whose size is decided

Table 1: Comparison of Neural Network training algorithm

Algorithm	Advantages	Disadvantages
Gradient descent	Employs first order algorithm to find minimum of a function	Require many iterations for functions which have long narrow valley structures. slow convergence. Prone to get stuck in local minima.
Newton's method	Require fewer steps than gradient descent to find minimum value of loss function.	Requires more information for evaluation, storage and inversion of Hessian Matrix.
Conjugate gradient	Faster convergence than gradient descent.	Line minimisation can be computationally expensive.
Quasi Newton	It is faster than gradient descent and conjugate gradient. Hessian matrix does not need to be computed and inverted.	It needs to store and update a matrix of size M x M.
Levenberg Marquardt	Works without computing exact Hessian matrix Performs well with loss functions which take the sum of squared errors. Error function is minimised, while the step size is kept small.	Requires a lot of memory when computing Jacobian matrix for big datasets.

by the number of patterns, number of outputs, and the number of weights [29].

For large-sized networks and training patterns, even though the Levenberg–Marquardt algorithm is very efficient, the computational cost may be too expensive or prohibitive to handle the Jacobian matrix storage and the Hessian matrix inversion calculation [29]. For our experiment, there are a few thousands of instances (i.e., 3600 in total), three input parameters and one output. This can be easily classified as small or medium sized problem. Hence, the Levenberg-Marquardt training algorithm is adopted in our experiment due to its speed, stable convergence and less memory consumption (in

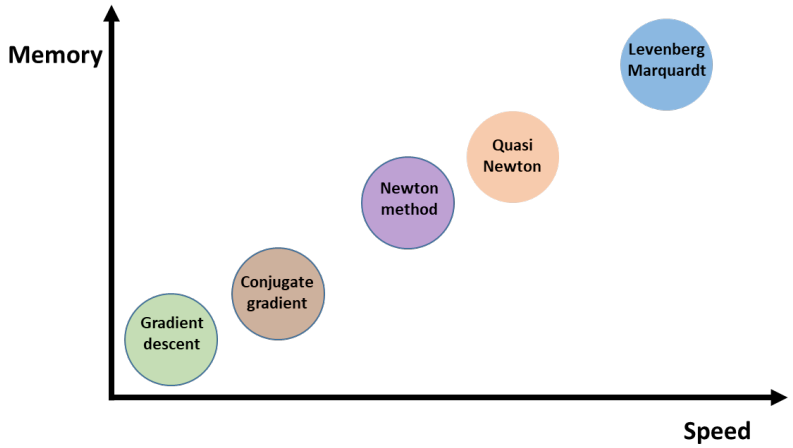

Figure 2: Memory speed comparison of neural network algorithm

this case due to a few parameters).

5.2. System Architecture and Setup

A custom network topology has been designed using Mininet emulator [30] to address the problems highlighted earlier. Tree topology is considered because it can be easily adopted for a wide area network and it offers easy expansion of node.

The custom topology shown in Figure 3 is implemented on 32G RAM Intel Xeon E3-1220 processor with Kali Linux as the base operating system. The floodlight controller [31] is deployed in the VirtualBox VM running Ubuntu 18.10 LTS while Mininet software is deployed on VirtualBox VM running Ubuntu 16.10 LTS.

The modelled network comprises 10 OpenFlow switches and 16 hosts which are connected using 100 Mbps link. The essential software tool includes iperf which is used to create a client-server relationship and low orbit ion cannon (LOIC) [32] to generate DDoS flooding attack. Using "iperf" and "ping" commands to generate legitimate traffic between the host and server, system properties such as response time, throughput and jitter values generated were recorded per second for a duration of 15 minutes.

In the attack scenario, an assumption was made that the attack is from an internal source. Hence, compromised hosts within the network were used to launch HTTP, TCP and UDP flood attacks on the server for 15 minutes respectively and results recorded from the server.

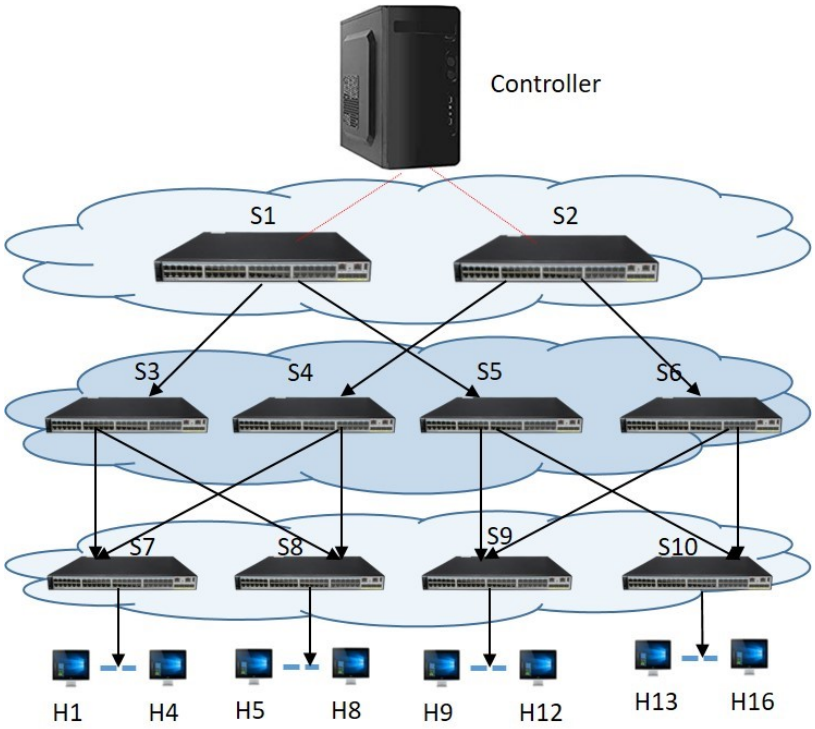

Figure 3: Modelled SDN tree architecture

6. Description of Dataset

Over time, emphasis has been on the development of algorithm to solve problems. With the growing generation of big data due to migration to 5G and beyond, internet of things (IoT) and cyber-physical processes, it becomes pertinent to develop a means for the accurate representation of data before developing an algorithm that fits the data [33][34].

The dataset for this experiment is generated via the modelled tree topology described in Figure 3. Four scenarios namely:1.Without attack (data collected when there was no attack), 2. With TCP flooding attack (data collected when TCP attack launched), 3. With UDP flooding attack (data collected when UDP attack launched) and 4. With HTTP flooding attack (data collected when HTTP attack launched) scenarios were considered. Each experiment was performed for 15 minutes and corresponding network traffic was recorded per second. Hence, there are 900 samples for each scenario to have a total of 3600 data samples. Throughput, Jitter and Response time

features were extracted using KNIME to create our dataset. Tables 2 - 5 provide the descriptive statistics for the generated data.

Table 2: Descriptive statistics of actual simulation data (over 900 data samples) for normal scenario

Metric	Min	Max	Average	Median	Standard Deviation
T_p	95.1000	95.9000	95.6332	95.6000	0.1402
R_t	0.0320	2.1200	0.2114	0.1980	0.1286
J_t	0.0040	0.4930	0.2271	0.1940	0.0943

Table 3: Descriptive statistics of T_p, R_t and J_t(over 900 data samples) for TCP attack scenario

Metric	Min	Max	Average	Median	Standard Deviation
T_p	0	95.9000	0.5441	0.0238	7.0999
R_t	0.2650	678	302.2676	299	110.6598
J_t	0.0040	0.4930	0.2271	0.1940	0.0943

Table 4: Descriptive statistics of T_p, R_t and J_t (over 900 data samples) for UDP attack scenario

Metric	Min	Max	Average	Median	Standard Deviation
T_p	95.1000	95.9000	95.6332	95.6000	0.1402
R_t	0.1980	82.1000	26.3097	24.8000	7.3245
J_t	9.1610	18.4280	10.5100	10.1725	1.0496

Table 5: Descriptive statistics of T_p, R_t and J_t(over 900 data samples) for HTTP attack scenario

Metric	Min	Max	Average	Median	Standard Deviation
T_p	0	95.9000	0.7429	0	8.3955
R_t	0.0200	1673	49.1262	23.7000	90.9398
J_t	0.0040	0.4930	0.2271	0.1940	0.0943

It can be seen from Tables 2 - 5 that the average throughput during attack drops significantly as compared to without attack scenario for TCP and HTTP flooding attack. Similarly, the jitter is affected adversely for the UDP attack and without attack scenario. In all cases of attack, there is an increase in response time. Thus, it can be deduced that each of these features are sensitive. However, the goal is to establish which is the most sensitive.

7. Experimental Approach

The local sensitivity analysis is carried out to determine the sensitivity of throughput, jitter, and response time to DDoS flooding attack. The goal is to determine if truly these metrics are sensitive to attack and which one is the most sensitive. Figure 4 shows the methodology flow chart of the 5-stage approach employed in performing local sensitivity analysis.

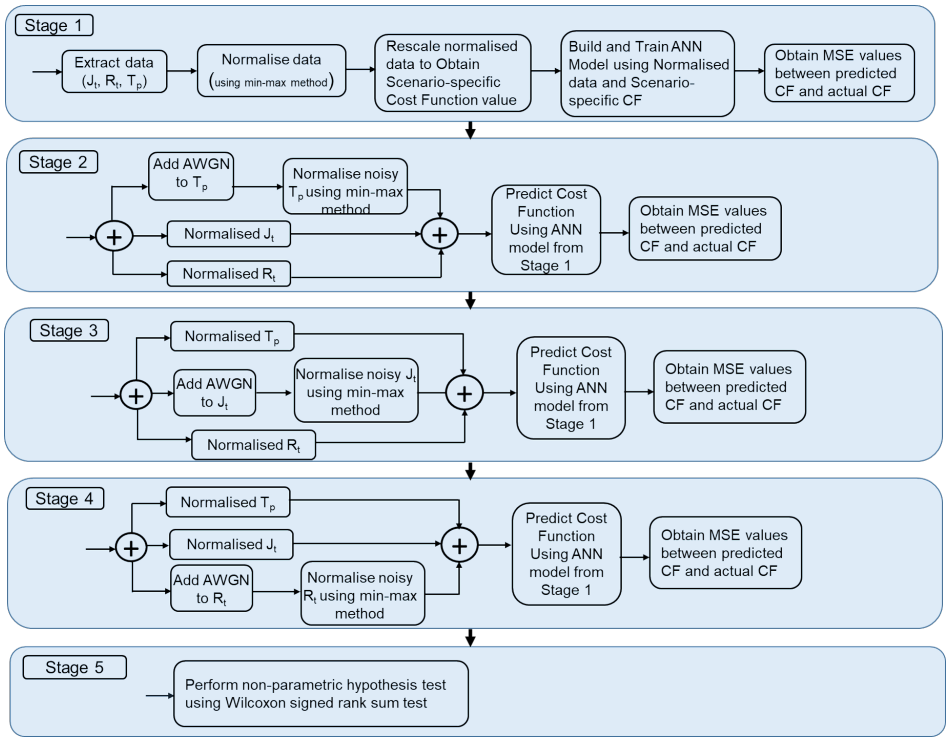

Figure 4: LSA methodology flow diagram

- **Stage 1**: At stage 1, throughput, jitter and response time features are extracted and the data is normalised using min-max method. The normalised data is used to obtain cost function values which are then fed as input training data into the ANN where MSE values are obtained.

- **Stage 2 - 4**: In these stages, additive white Gaussian noise(AWGN) is added based on one-at-a-time basis. AWGN is added to throughput(T_p)to have noisy T_p while other factors (Jitter (J_t) and response time (R_t)) were kept constant in stage 2. This process is repeated for noisy J_t (Stage 3) and noisy R_t (Stage 4) while other parameters are kept constant and new cost function values are predicted, respectively in each stage(i.e., Stages 2-4).

- **Stage 5**: Hypothesis test is carried out at this stage to statistically validate any of the inferences made from the deviations.

7.1. Data Normalisation

The data were normalised such that each system parameter contributes similar relative numerical weight in order to minimise data redundancy and ensure all target input values have an agreeable metric scale. The data normalisation process employs min-max normalisation method. Min-Max normalisation is a strategy which linearly transforms variable 'X' so that the entire range of values of X from minimum to maximum varies between 0 and 1. It can be expressed mathematically as:

$$X_{normalised} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{3}$$

where X_{min} and X_{max} are the minimum and maximum values in X respectively. Tables 6 - 9 show the descriptive statistics of the normalised values for tables presented in section 6

Table 6: Descriptive statistics of normalised T_p, R_t and J_t(over 900 data samples) for without attack scenario

Metric	Min	Max	Average	Median	Standard Deviation
T_p	0.9917	1	0.9972	0.9969	0.0015
R_t	7.1728e-06	0.0013	1.1440e-04	1.0640e-04	7.6849e-05
J_t	0	0.0265	0.0121	0.0103	0.0051

Table 7: Descriptive statistics of normalised T_p, R_t and J_t (over 900 data samples) for TCP attack scenario

Metric	Min	Max	Average	Median	Standard Deviation
T_p	0	1	0.0057	2.4818e-04	0.0740
R_t	1.4645e-04	0.4053	0.1807	0.1787	0.0661
J_t	0	0.0265	0.0121	0.0103	0.0051

Table 8: Descriptive statistics of normalised T_p, R_t and J_t (over 900 data samples) for UDP attack scenario

Metric	Min	Max	Average	Median	Standard Deviation
T_p	0.9917	1	0.9972	0.9969	0.0015
R_t	1.0640e-04	0.0491	0.0157	0.0148	0.0044
J_t	0.4970	1	0.5702	0.5519	0.0570

Table 9: Descriptive statistics of normalised T_p, R_t and J_t (over 900 data samples) for HTTP attack scenario

Metric	Min	Max	Average	Median	Standard Deviation
T_p	0	1	0.0077	0	0.0875
R_t	0	1	0.0294	0.0142	0.0544
J_t	0	0.0265	0.0121	0.0103	0.0051

7.2. Cost function value evaluation

We use the normalised data in section 7.1 to build Input-Output correspondence and the normalised values are scaled to the following four scenarios:

- **Scenario 1**: a scale of 1 is assigned to represent without attack
- **Scenario 2**: a scale of 2 is assigned to represent with TCP attack
- **Scenario 3**: a scale of 3 is assigned to represent with UDP attack
- **Scenario 4**: a scale of 4 is assigned to represent with HTTP attack

in order to reflect scenario-specific targets from their corresponding values,a mathematical cost function in terms of throughput, jitter and response time is introduced. The proposed cost function tends toward unity for the worst case scenario (SDN under severe attack) and approaches zero for the best

case scenario (SDN without attack). The cost function can be represented mathematically by:

$$C_F = (abs(T_p - J_t) * R_t) * L_w \tag{4}$$

where C_F represents cost function, T_p = Throughput, J_t = Jitter , R_t = Response time and L_w = Weight or Scale.

For the best case scenario (i.e.,normal SDN state), the throughput is maximum, response time is minimum and jitter is minimum. Therefore, T_p approaches 1, J_t and R_t approach zero after normalisation. So, we have the following condition for the best case scenario.

$$Bestcase = \begin{cases} T_p \rightarrow 1 \\ J_t \rightarrow 0 \\ R_t \rightarrow 0 \end{cases} \tag{5}$$

Similarly, for the worst case scenario (i.e., SDN under severe attack), the throughput is minimum, response time is maximum and jitter is maximum. Therefore, T_p approaches 0, J_t and R_t approach 1 after normalisation. Hence, we have the following condition for the worst case scenario.

$$Worstcase = \begin{cases} T_p \rightarrow 0 \\ J_t \rightarrow 1 \\ R_t \rightarrow 1 \end{cases} \tag{6}$$

Substituting the values in equations 5 and 6 into equation 4 , C_F approaches null (zero) for normal network state and approaches unity (one) when the SDN is under attack. A descriptive statistics of the cost function value over 3600 samples for normal, UDP,TCP, and HTTP flooding attack scenarios is presented in Table10.

As shown in Figure 5, the cost function value(C_F) associated with normal (without attack) network traffic hovers around zero. This value satisfies the condition for our best case scenario. The other attack scenario has cost function value well above zero with peak values of 0.78 and 0.63 recorded for HTTP and UDP flood traffic respectively.

The cost function value(C_F) simply indicates that the system parameters experience changes due to attacks. To ascertain the most sensitive of these parameters due to attacks, local sensitivity analysis is carried out.

Table 10: Descriptive Statistics of the Cost function value (over 3600 data samples) for normal, TCP, UDP and HTTP attack scenarios

Metric	Min	Max	Average	Median	Standard Deviation
Normal	7.0837e-05	0.0124	0.0011	0.0010	7.5756e-04
TCP	0	0.1528	0.0435	0.0382	0.0251
UDP	0.0014	0.6656	0.2018	0.1968	0.0654
HTTP	0	0.7728	0.0147	0.0065	0.0336

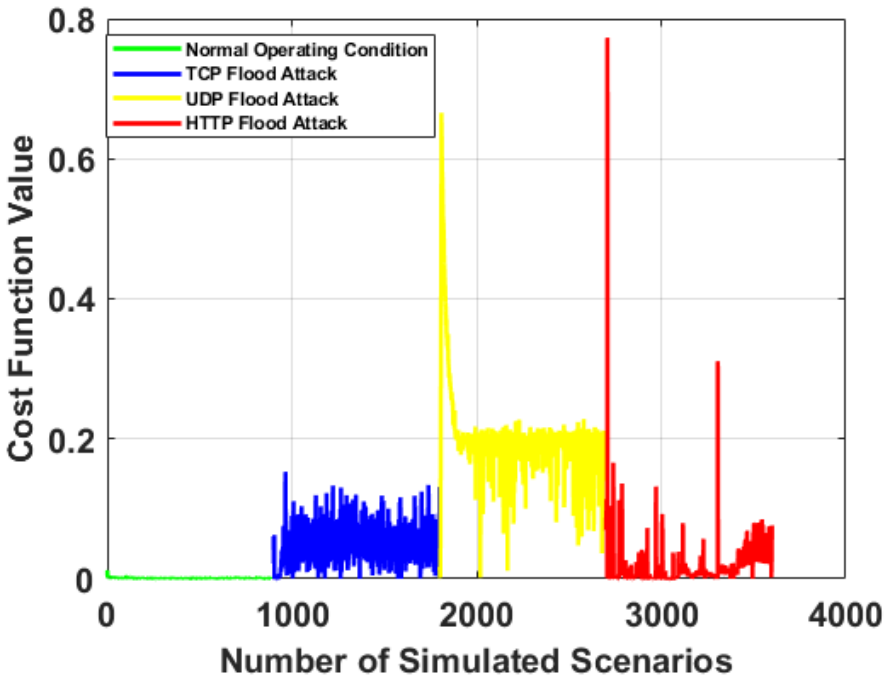

Figure 5: Variation in cost function value versus attack

7.3. AWGN and MSE

additive white Gaussian noise (AWGN) is a statistical noise with a Probability Density Function equal to that of standard normal distribution. AWGN is characterised with bell-shaped curve as shown in Figure 6 with a mean value of zero, standard deviation value of 1 and total area under the curve is 1. For the local sensitivity analysis, AWGN is added to throughput, jitter and response time,respectively, as shown in stages 2 - 4 of the LSA method-

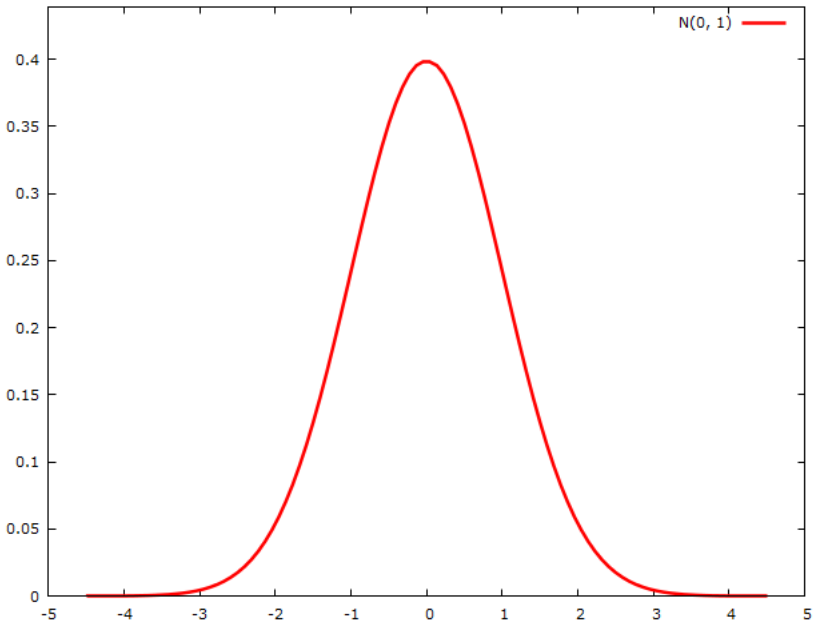

Figure 6: AWGN distribution

ology flow chart(see figure 4). Mean Squared Error(MSE) values over 50 runs is obtained afterward. MSE measures the average squared difference between the predicted values and the actual value. MSE is expressed mathematically as:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y_i})^2 \tag{7}$$

7.4. ANN training

A prediction model is built using ANN with the normalised value discussed in Section 7.1 as input and the cost function values described in Table10 as the target values. The ANN model is trained with the three inputs metrics (i.e., T_p, R_t and J_t), 10 hidden layers and a single output (i.e.,C_F)under 9 iterations as shown in Figure 7. The best validation performance is at epoch 3 (see Figure 8). The number of processing elements per layer, as well as the number of layers greatly influence the training process. Too few processing elements can slow down the learning process and too many can lead to overfitting of the training dataset [35][36].For our experiment, 2520 data samples (70% of the total data samples) have been used

as the training data set, 540 data samples (15% of the total data samples) have been used as the validation data set and 540 data samples (15% of the total data samples) have been used as the test data set according to the data portioning approach recommended in several works [37][38]. Figure 8 shows that the ANN model is correct and acceptably accurate.

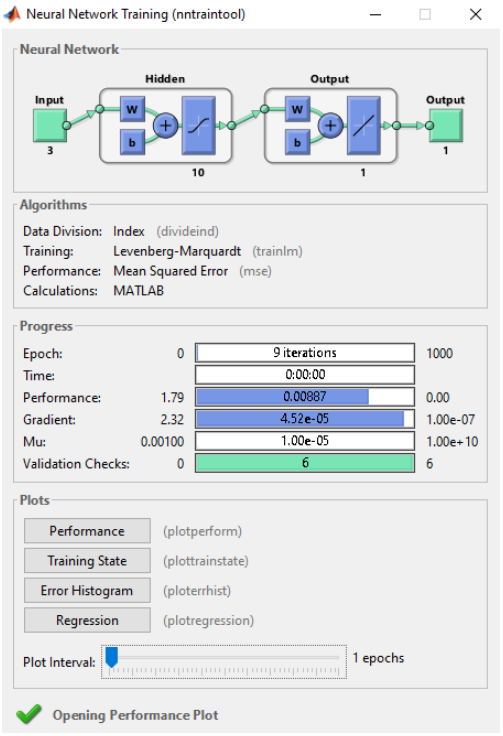

Figure 7: ANN training model

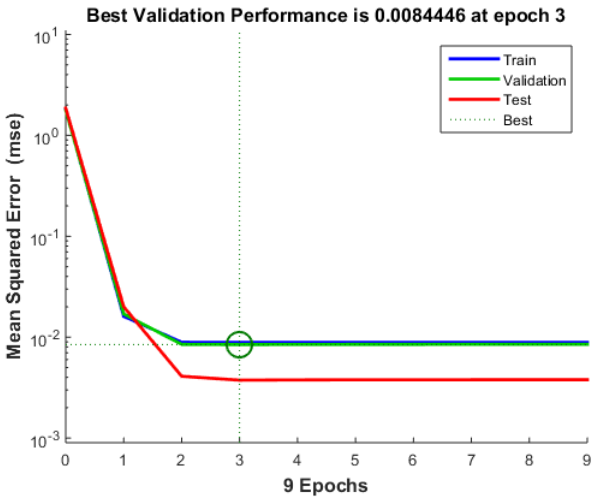

Figure 8: A typical plot of MSE vs number of Epochs

8. Results and Discussion

The sensitivity of throughput, jitter and response time is evaluated using the deviation of newly predicted target value from actual target values obtained and the MSE value of the prediction model. Tables 11 - 13 show the impact of adding AWGN to the impact metrics. For the 50 independent statistical runs to validate the statistical significance of this experiment, it can be seen that jitter's standard deviation value is considerably more than what is obtainable in the T_p noisy and R_t noisy respectively (See the appendices for the complete tables).

Table 11: Local sensitivity analysis for noisy T_p, normalised R_t, normalised J_t (over 3600 data samples) for 50 statistical runs

No of runs	Min	Max	Average	Median	Standard Deviation
1	-0.2907	0.6820	-0.0230	-0.0632	0.0905
2	-0.1138	0.6911	0.0090	-0.0281	0.0908
3	-0.0876	0.7104	0.0102	-0.0265	0.0915
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
48	-0.1589	0.7035	0.0113	-0.0242	0.0917
49	-0.2439	0.6930	-0.0182	-0.0603	0.0894
50	-0.0753	0.7132	0.0036	-0.0390	0.0893

Table 12: Local sensitivity analysis for noisy R_t, normalised J_t, normalised T_p (over 3600 data samples) for 50 statistical runs

No of runs	Min	Max	Average	Median	Standard Deviation
1	-0.1461	0.6995	-0.0017	-0.0445	0.0898
2	-0.0896	0.6993	-0.0107	-0.0528	0.0895
3	-0.0943	0.7007	-0.0090	-0.0513	0.0893
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
48	-0.2157	0.6997	-0.0091	-0.0500	0.0895
49	-0.2500	0.7009	-0.0100	-0.0521	0.0895
50	-0.0836	0.6922	-0.0093	-0.0511	0.0894

Table 13: Local sensitivity analysis for noisy J_t, normalised R_t, normalised T_p (over 3600 data samples) for 50 statistical runs

No of runs	Min	Max	Average	Median	Standard Deviation
1	-0.2783	0.6505	-0.0323	-0.0518	0.1307
2	-0.4036	0.7082	-0.0770	-0.0655	0.1249
3	-0.3362	0.6227	-0.0239	-0.0536	0.0962
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
48	-0.4165	0.7079	0.0191	-0.0179	0.0912
49	-0.1906	0.7496	0.0007	-0.0350	0.0922
50	-0.0896	0.7296	0.0247	-0.0012	0.0998

8.1. Hypothesis test

Wilcoxon test [39] is carried out over 50 runs when J_t, T_p, and R_t are noisy and when they are not noisy. Since the sample size is sufficiently large (that is, 50 in this case), a z-statistic can be used to approximate the probability value (p-value) of the test [40]. This is why Wilcoxon test[39] is an appropriate test for statistical significance in this case. The Wilcoxon test is a non-parametric test that obeys the central limit theorem. It tests the null hypothesis that the normalised data and its noisy version are from continuous distributions with equal medians. A common significance level of 0.05 (i.e., 5%) is selected. If the resultant p-value is equal to or less than 0.05, then, there is strong evidence against the null hypothesis. The p-value obtained from the Wilcoxon test is shown in table 14. From table 14, it can

be seen that the null hypothesis is rejected for all cases. This indicates that noisy J_t, T_p, and R_t are all statistically sensitive.

Table 14: Descriptive Statistics of the MSE Values Over 50 Statistical Runs

Metric	Min	Max	Average	Median	Standard Deviation	p-value
S1	0.00797	0.00826	0.0080	0.00803	0.000047	N/A
S2	0.00797	0.01145	0.0084	0.00824	0.000634	2.4225e-09
S3	0.00797	0.00843	0.0081	0.00807	0.000066	1.7330e-17
S4	0.00799	0.02951	0.0115	0.02951	0.004858	1.7330e-17

- S1: normalised T_p, normalised R_t, normalised J_t
- S2: noisy T_p, normalised R_t, normalised J_t
- S3: noisy R_t, normalised J_t, normalised T_p
- S4: noisy J_t, normalised R_t, normalised T_p

A plot of MSE values against the number of runs is shown in Figure 9. Using rank sum, the result shows that jitter is the most sensitive to flooding attack followed by throughput and then response time. The work presented in [41] also shows that jitter may severely degrade systems performance. It is worthy of note that all parameters evaluated are sensitive to DDoS attacks. Hence, adequate prevention and mitigation schemes can be deployed in SDN controller if these features are embedded in the attack detection scheme.

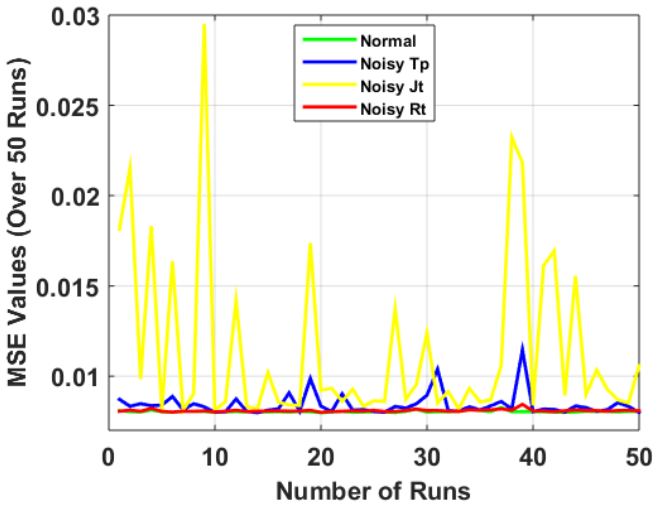

Figure 9: Sensitivity analysis of throughput (T_p), jitter (J_t) and response time(R_t)

9. Conclusion

In this paper, LSA is implemented on real SDN traffic to identify the key metrics that mainly influence the prediction of whether an SDN is under attack or secure. The SDN traffic dataset considered are throughput, response time and jitter, and they are generated from a modelled tree topology in Mininet. The SDN is subjected to a DDoS flooding attack launched using LOIC. An ANN prediction model is built using a min-max feature scaling to derive actual target values from the normalised input parameters. The sensitivity of throughput, jitter and response time is then evaluated using the deviations of newly predicted target values from actual target values when an AWGN is added to the respective SDN traffic dataset. Results obtained show that throughput, jitter and response time are all statistically sensitive to a DDoS flooding attack on the SDN, and jitter is the most sensitive of all the impact metrics considered. In the future, global sensitivity analysis paradigms will be developed to analyse the impact of varying network metrics simultaneously and uniformly over their full range of possible values at a reduced computational overhead.

References

[1] J. Xu, L. Wang, Z. Xu, An enhanced saturation attack and its mitigation mechanism in software-defined networking, Computer Networks 169 (2020) 107092.

[2] A. Sangodoyin, B. Mohammed, M. Sibusiso, I. Awan, J. P. Disso, A framework for distributed denial of service attack detection and reactive countermeasure in software defined network, in: 2019 7th International Conference on Future Internet of Things and Cloud (FiCloud), IEEE, 2019, pp. 80–87.

[3] G.-Y. Chan, C.-S. Lee, S.-H. Heng, Discovering fuzzy association rule patterns and increasing sensitivity analysis of xml-related attacks, Journal of Network and Computer Applications 36 (2) (2013) 829–842.

[4] Y. Al-Hammadi, U. Aickelin, J. Greensmith, Dca for bot detection, in: 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), IEEE, 2008, pp. 1807–1816.

[5] R. L. Iman, W. Conover, Small sample sensitivity analysis techniques for computer models. with an application to risk assessment, Communications in statistics-theory and methods 9 (17) (1980) 1749–1842.

[6] R. L. Iman, J. C. Helton, An investigation of uncertainty and sensitivity analysis techniques for computer models, Risk analysis 8 (1) (1988) 71–90.

[7] R. d. S. M. Júnior, A. P. Guimaraes, K. M. Camboim, P. R. Maciel, K. S. Trivedi, Sensitivity analysis of availability of redundancy in computer networks, CTRQ 2011 (2011) 122.

[8] S. Hashem, Sensitivity analysis for feedforward artificial neural networks with differentiable activation functions, in: [Proceedings 1992] IJCNN International Joint Conference on Neural Networks, Vol. 1, IEEE, 1992, pp. 419–424.

[9] W. W. Ng, R. K. Chang, D. S. Yeung, Dimensionality reduction for denial of service detection problems using rbfnn output sensitivity, in: Proceedings of the 2003 International Conference on Machine Learning

and Cybernetics (IEEE Cat. No. 03EX693), Vol. 2, IEEE, 2003, pp. 1293–1298.

[10] R. H. Jhaveri, N. M. Patel, Y. Zhong, A. K. Sangaiah, Sensitivity analysis of an attack-pattern discovery based trusted routing scheme for mobile ad-hoc networks in industrial iot, IEEE Access 6 (2018) 20085–20103.

[11] A. Saltelli, S. Tarantola, F. Campolongo, M. Ratto, Sensitivity analysis in practice: a guide to assessing scientific models, Chichester, England (2004).

[12] X. Zhou, H. Lin, Local Sensitivity Analysis, Springer International Publishing, Cham, 2017, Ch. Local Sensitivity Analysis, pp. 1130–1131.

[13] J. M. Zurada, A. Malinowski, I. Cloete, Sensitivity analysis for minimization of input data dimension for feedforward neural network, in: Proceedings of IEEE International Symposium on Circuits and Systems-ISCAS94, Vol. 6, IEEE, 1994, pp. 447–450.

[14] A. Hunter, L. Kennedy, J. Henry, I. Ferguson, Application of neural networks and sensitivity analysis to improved prediction of trauma survival, Computer methods and programs in biomedicine 62 (1) (2000) 11–19.

[15] J. Kirch, C. Thomaseth, A. Jensch, N. E. Radde, The effect of model rescaling and normalization on sensitivity analysis on an example of a mapk pathway model, EPJ Nonlinear Biomedical Physics 4 (1) (2016) 3.

[16] J. M. Zurada, A. Malinowski, S. Usui, Perturbation method for deleting redundant inputs of perceptron networks, Neurocomputing 14 (2) (1997) 177–193.

[17] P. A. Kowalski, M. Kusy, Sensitivity analysis for probabilistic neural network structure reduction, IEEE transactions on neural networks and learning systems 29 (5) (2017) 1919–1932.

[18] A. Saltelli, P. Annoni, How to avoid a perfunctory sensitivity analysis, Environmental Modelling and Software 25 (12) (2010) 1508–1517.

[19] A. Saltelli, Sensitivity analysis for importance assessment, Risk analysis 22 (3) (2002) 579–590.

[20] A. Saltelli, M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana, S. Tarantola, Global sensitivity analysis: the primer, John Wiley & Sons, 2008.

[21] B. Iooss, P. Lemaître, A review on global sensitivity analysis methods, in: Uncertainty management in simulation-optimization of complex systems, Springer, 2015, pp. 101–122.

[22] K. G. Link, M. T. Stobb, J. Di Paola, K. B. Neeves, A. L. Fogelson, S. S. Sindi, K. Leiderman, A local and global sensitivity analysis of a mathematical model of coagulation and platelet deposition under flow, PloS one 13 (7) (2018) e0200917.

[23] B. Widrow, M. A. Lehr, 30 years of adaptive neural networks: perceptron, madaline, and backpropagation, Proceedings of the IEEE 78 (9) (1990) 1415–1442.

[24] M.-J. Kang, J.-W. Kang, Intrusion detection system using deep neural network for in-vehicle network security, PloS one 11 (6) (2016) e0155781.

[25] J. B. Ali, N. Fnaiech, L. Saidi, B. Chebel-Morello, F. Fnaiech, Application of empirical mode decomposition and artificial neural network for automatic bearing fault diagnosis based on vibration signals, Applied Acoustics 89 (2015) 16–27.

[26] A. Chojaczyk, A. Teixeira, L. C. Neves, J. Cardoso, C. G. Soares, Review and application of artificial neural networks models in reliability analysis of steel structures, Structural Safety 52 (2015) 78–89.

[27] S. J. Russell, P. Norvig, Artificial intelligence: a modern approach, Malaysia; Pearson Education Limited,, 2016.

[28] M. T. Hagan, M. B. Menhaj, Training feedforward networks with the marquardt algorithm, IEEE transactions on Neural Networks 5 (6) (1994) 989–993.

[29] H. Yu, B. M. Wilamowski, Levenberg-marquardt training, Industrial electronics handbook 5 (12) (2011) 1–16.

[30] Mininet, Download/Get Started with Mininet (2019).
URL http://mininet.org/download/

[31] Floodlight, Floodlight controller (2019).
URL http://www.projectfloodlight.org/

[32] LOIC, Praetox Technologies Low Orbit Ion Cannon (2019).
URL https://github.com/NewEraCracker/LOIC/

[33] S. Suthaharan, Big data classification: Problems and challenges in network intrusion prediction with machine learning, ACM SIGMETRICS Performance Evaluation Review 41 (4) (2014) 70–73.

[34] R. Sommer, V. Paxson, Outside the closed world: On using machine learning for network intrusion detection, in: 2010 IEEE symposium on security and privacy, IEEE, 2010, pp. 305–316.

[35] M. K. S. Alsmadi, K. B. Omar, S. A. Noah, et al., Back propagation algorithm: the best algorithm among the multi-layer perceptron algorithm, International Journal of Computer Science and Network Security 9 (4) (2009) 378–383.

[36] C. Aldrich, Exploratory analysis of metallurgical process data with neural networks and related methods, Vol. 12, Elsevier, 2002.

[37] I. Brown, C. Mues, An experimental comparison of classification algorithms for imbalanced credit scoring data sets, Expert Systems with Applications 39 (3) (2012) 3446–3453.

[38] D. Chicco, Ten quick tips for machine learning in computational biology, BioData mining 10 (1) (2017) 35.

[39] F. Wilcoxon, Individual comparisons by ranking methods, in: Breakthroughs in statistics, Springer, 1992, pp. 196–202.

[40] J. D. Gibbons, S. Chakraborti, Nonparametric Statistical Inference: Revised and Expanded, CRC press, 2014.

[41] M. Long, C.-H. Wu, J. Y. Hung, Denial of service attacks on network-based control systems: impact and mitigation, IEEE Transactions on Industrial Informatics 1 (2) (2005) 85–96.

Appendix A: Local sensitivity analysis for T_p noisy, normalised R_t, normalised J_t (over 3600 data samples) for 50 statistical runs

No of runs	Min	Max	Average	Median	Standard Deviation
1	-0.2907	0.6820	-0.0230	-0.0632	0.0905
2	-0.1138	0.6911	0.0090	-0.0281	0.0908
3	-0.0876	0.7104	0.0102	-0.0265	0.0915
4	-0.5090	0.6848	-0.0166	-0.0583	0.0900
5	-0.1144	0.6929	-0.0195	-0.0619	0.0895
6	-0.1281	0.7314	-0.0024	-0.0315	0.0942
7	-0.1177	0.6969	-0.0114	-0.0529	0.0895
8	-0.1248	0.6730	-0.0141	-0.0516	0.0910
9	-0.1677	0.7188	-0.0065	-0.0457	0.0909
10	-0.2079	0.6964	-0.0054	-0.0469	0.0893
11	-0.1413	0.7068	-0.0036	-0.0449	0.0895
12	-0.1582	0.6808	-0.0221	-0.0578	0.0908
13	-0.0767	0.7080	0.0012	-0.0400	0.0897
14	-0.0764	0.7064	-0.0014	-0.0427	0.0893
15	-0.1062	0.7064	0.0001	-0.0420	0.0901
16	-0.2337	0.6871	-0.0126	-0.0543	0.0896
17	-0.1832	0.6967	-0.0237	-0.0620	0.0923
18	-0.1964	0.6935	-0.0072	-0.0483	0.0894
19	-0.2736	0.7796	0.0292	-0.0042	0.0950
20	-0.1658	0.6703	0.0019	-0.0355	0.0913
21	-0.1491	0.7115	-0.0012	-0.0438	0.0896
22	-0.1402	0.6364	-0.0170	-0.0478	0.0935
23	-0.1023	0.7022	0.0012	-0.0391	0.0901
24	-0.1567	0.7126	-0.0123	-0.0535	0.0894
25	-0.1739	0.6962	-0.0088	-0.0496	0.0893
26	-0.1628	0.6986	-0.0062	-0.0478	0.0892
27	-0.0838	0.7068	0.0095	-0.0278	0.0907
28	-0.0949	0.7559	0.0092	-0.0290	0.0903
29	-0.1309	0.7172	-0.0091	-0.0451	0.0916
30	-0.1619	0.7354	-0.0079	-0.0392	0.0942
31	-0.3041	0.7020	0.0162	-0.0231	0.1006
32	-0.0808	0.6946	-0.0086	-0.0499	0.0895
33	-0.0943	0.7100	0.0005	-0.0411	0.0898
34	-0.5080	0.6945	-0.0121	-0.0549	0.0903
35	-0.2063	0.6948	-0.0107	-0.0527	0.0895
36	-0.1662	0.7003	-0.0161	-0.0581	0.0900
37	-0.9084	0.7033	-0.0169	-0.0585	0.0912
38	-0.1037	0.7497	0.0053	-0.0349	0.0903
39	-0.2242	0.7425	-0.0207	-0.0351	0.1050
40	-0.1298	0.7061	-0.0028	-0.0450	0.0896
41	-0.1020	0.7055	-0.0071	-0.0482	0.0902
42	-0.1680	0.6896	0.0004	-0.0371	0.0903
43	-0.1595	0.6956	-0.0042	-0.0474	0.0894
44	-0.1175	0.6898	-0.0100	-0.0505	0.0908
45	-0.1828	0.7074	0.0107	-0.0279	0.0902
46	-0.1252	0.7173	-0.0016	-0.0435	0.0897
47	-0.1248	0.7038	-0.0100	-0.0521	0.0897
48	-0.1589	0.7035	0.0113	-0.0242	0.0917
49	-0.2439	0.6930	-0.0182	-0.0603	0.0894
50	-0.0753	0.7132	0.0036	-0.0390	0.0893

Appendix B: Local sensitivity analysis for R_t noisy, normalised J_t, normalised T_p (over 3600 data samples) for 50 statistical runs

No of runs	Min	Max	Average	Median	Standard Deviation
1	-0.1461	0.6995	-0.0017	-0.0445	0.0898
2	-0.0896	0.6993	-0.0107	-0.0528	0.0895
3	-0.0943	0.7007	-0.0090	-0.0513	0.0893
4	-0.5218	0.6938	-0.0113	-0.0531	0.0899
5	-0.1139	0.6994	-0.0083	-0.0499	0.0893
6	-0.0945	0.6979	-0.0071	-0.0487	0.0892
7	-0.0981	0.6940	-0.0096	-0.0507	0.0892
8	-0.1146	0.6971	-0.0098	-0.0506	0.0892
9	-0.1631	0.6996	-0.0095	-0.0512	0.0893
10	-0.1984	0.7058	-0.0057	-0.0466	0.0894
11	-0.1032	0.6975	-0.0080	-0.0494	0.0894
12	-0.1517	0.7066	-0.0099	-0.0519	0.0895
13	-0.0881	0.6998	-0.0084	-0.0498	0.0893
14	-0.0793	0.6946	-0.0093	-0.0503	0.0893
15	-0.1021	0.6980	-0.0090	-0.0505	0.0894
16	-0.2043	0.7023	-0.0098	-0.0506	0.0893
17	-0.1486	0.7044	-0.0007	-0.0415	0.0898
18	-0.1946	0.6948	-0.0096	-0.0503	0.0893
19	-0.1317	0.6960	-0.0104	-0.0519	0.0895
20	-0.1294	0.6967	-0.0050	-0.0461	0.0893
21	-0.1607	0.7009	-0.0087	-0.0499	0.0893
22	-0.0946	0.6924	-0.0101	-0.0515	0.0892
23	-0.1305	0.7038	-0.0056	-0.0483	0.0897
24	-0.1373	0.7020	-0.0080	-0.0496	0.0894
25	-0.1825	0.6964	-0.0092	-0.0506	0.0896
26	-0.1729	0.6990	-0.0078	-0.0491	0.0893
27	-0.0916	0.7021	-0.0081	-0.0496	0.0893
28	-0.1033	0.7043	-0.0096	-0.0514	0.0895
29	-0.1202	0.7047	-0.0118	-0.0539	0.0897
30	-0.1071	0.7102	-0.0088	-0.0505	0.0895
31	-0.2968	0.6928	-0.0096	-0.0504	0.0895
32	-0.0793	0.6947	-0.0088	-0.0496	0.0893
33	-0.0814	0.6985	-0.0099	-0.0512	0.0892
34	-0.5114	0.6970	-0.0102	-0.0509	0.0896
35	-0.1939	0.6960	-0.0112	-0.0525	0.0894
36	-0.1635	0.7019	-0.0101	-0.0509	0.0896
37	-0.7064	0.6965	-0.0091	-0.0505	0.0900
38	-0.1086	0.7015	-0.0082	-0.0487	0.0897
39	-0.1278	0.7052	-0.0159	-0.0573	0.0905
40	-0.1604	0.6974	-0.0096	-0.0506	0.0893
41	-0.1781	0.7041	-0.0084	-0.0500	0.0895
42	-0.0909	0.6905	-0.0082	-0.0499	0.0892
43	-0.1863	0.6982	-0.0038	-0.0456	0.0898
44	-0.0860	0.7011	-0.0084	-0.0501	0.0897
45	-0.1440	0.6934	-0.0107	-0.0521	0.0893
46	-0.1094	0.6955	-0.0108	-0.0519	0.0893
47	-0.0889	0.6941	-0.0102	-0.0514	0.0892
48	-0.2157	0.6997	-0.0091	-0.0500	0.0895
49	-0.2500	0.7009	-0.0100	-0.0521	0.0895
50	-0.0836	0.6922	-0.0093	-0.0511	0.0894

Appendix C: Local sensitivity analysis for J_t noisy, normalised R_t, normalised T_p (over 3600 data samples) for 50 statistical runs

No of runs	Min	Max	Average	Median	Standard Deviation
1	-0.2783	0.6505	-0.0323	-0.0518	0.1307
2	-0.4036	0.7082	-0.0770	-0.0655	0.1249
3	-0.3362	0.6227	-0.0239	-0.0536	0.0962
4	-0.5330	0.7538	0.0612	0.0708	0.1207
5	-0.1335	0.7024	-0.0008	-0.0401	0.0900
6	-0.3098	0.6035	-0.0655	-0.0718	0.1099
7	-0.1327	0.6900	-0.0092	-0.0515	0.0896
8	-0.1724	0.6709	-0.0271	-0.0663	0.0911
9	-0.5250	0.6866	-0.1055	-0.0780	0.1356
10	-0.2824	0.6954	-0.0008	-0.0413	0.0900
11	-0.1689	0.6948	-0.0204	-0.0610	0.0905
12	-0.3031	0.7189	-0.0511	-0.0571	0.1076
13	-0.1432	0.6935	-0.0101	-0.0510	0.0907
14	-0.1075	0.6804	-0.0133	-0.0568	0.0895
15	-0.1131	0.7846	0.0043	-0.0342	0.1009
16	-0.2020	0.6969	0.0063	-0.0280	0.0923
17	-0.2129	0.6884	-0.0096	-0.0487	0.0912
18	-0.1135	0.7454	0.0056	-0.0320	0.0914
19	-0.3325	0.6425	-0.0747	-0.0896	0.1086
20	-0.1571	0.7642	0.0068	-0.0265	0.0958
21	-0.2269	0.6976	-0.0288	-0.0698	0.0922
22	-0.1246	0.7236	-0.0044	-0.0384	0.0926
23	-0.2143	0.6970	0.0207	-0.0112	0.0941
24	-0.1207	0.7043	-0.0066	-0.0431	0.0911
25	-0.1758	0.6967	-0.0123	-0.0432	0.0921
26	-0.1305	0.6945	0.0141	-0.0185	0.0917
27	-0.0850	0.7126	0.0452	0.0298	0.1082
28	-0.1426	0.6511	-0.0113	-0.0409	0.0927
29	-0.1435	0.6817	-0.0368	-0.0753	0.0904
30	-0.2022	0.7572	0.0150	-0.0066	0.1103
31	-0.1580	0.6980	-0.0162	-0.0568	0.0909
32	-0.2035	0.6875	-0.0215	-0.0599	0.0932
33	-0.0918	0.6951	-0.0092	-0.0505	0.0900
34	-0.6251	0.6365	-0.0271	-0.0646	0.0927
35	-0.1935	0.6802	-0.0224	-0.0655	0.0898
36	-0.3200	0.6875	-0.0080	-0.0365	0.0929
37	-0.8707	0.6955	-0.0220	-0.0444	0.1005
38	-0.1098	0.8213	0.0793	0.0825	0.1302
39	-0.2530	0.7218	0.0422	0.0240	0.1416
40	-0.4004	0.6906	-0.0036	-0.0431	0.0915
41	-0.1805	0.7446	0.0564	0.0499	0.1138
42	-0.2959	0.7020	-0.0802	-0.0986	0.1025
43	-0.1863	0.6472	-0.0226	-0.0638	0.0918
44	-0.0888	0.6865	0.0600	0.0537	0.1093
45	-0.2275	0.6672	-0.0260	-0.0657	0.0912
46	-0.2108	0.7184	-0.0326	-0.0568	0.0963
47	-0.2364	0.6950	-0.0108	-0.0379	0.0957
48	-0.4165	0.7079	0.0191	-0.0179	0.0912
49	-0.1906	0.7496	0.0007	-0.0350	0.0922
50	-0.0896	0.7296	0.0247	-0.0012	0.0998