Parameter Sensitivity of AADRNN-Based Attack Detection

Mohammed Nasereddin
Inst. Theoretical & Appl. Informatics
Polish Academy of Sciences (IITIS-PAN)
44-100 Gliwice, Poland
ORCID: 0000-0002-3740-9518
Email: mnasereddin@iitis.pl

Erol Gelenbe, *Life Fellow, IEEE*
Inst. Theoretical & Appl. Informatics
Polish Academy of Sciences (IITIS-PAN)
44-100 Gliwice, Poland
Lab. I3S CNRS Université Côte d'Azur
06100 Nice, France
ORCID: 0000-0001-9688-2201
Email: seg@iitis.pl

***Abstract*—The growing prevalence of cyberattacks against network-connected systems increases the need for accurate intrusion and attack detection (AD) systems that can react rapidly at the network access ports. Machine learning has often been used for AD because of the high accuracy it can offer, and its fast testing speed for real-time attack detection. On the other hand, the data sampling methods used to transform incoming traffic streams into data for real-time testing by AD systems can have an important effect on the detection accuracy, and are seldom discussed in the literature. Here, we address this often neglected aspect, with the Auto-Associative Dense Random Neural Network (AADRNN), a state-of-the-art AD system. After briefly reviewing the literature and comparing the AADRNN attack detector with other state-of-the-art approaches, this paper focuses on choosing the values of the data sampling parameters and the decision threshold to obtain the best performance from this AD method. Experiments on Botnet attack detection using the Mirai dataset, as well as synthetic flood attacks conducted on an experimental IoT test-bed are used to illustrate the results, and to guide the parameter choices that maximize the AADRNN's True Positive Rate and minimize its False Positive Rate.**

***Index Terms*—Attack Detection and Mitigation, IoT Security, Flood Attacks, Botnets, Parameter Sensitivity, Machine Learning, Random Neural Network**

I. Introduction

Network-connected systems, and in particular Internet of Things (IoT) infrastructures, are increasingly vulnerable to cyberattacks such as Denial-of-Service (DoS) and Distributed DoS (DDoS) attacks, in which large numbers of compromised devices (Botnets) are exploited remotely without the owners' knowledge [1]. Such attacks overwhelm network resources, including servers, routers, or switches with excessive traffic, thereby blocking legitimate access to IoT devices or entire networks.

This paper investigates parameter-aware deployment of an AD system based on the AADRNN model [2]–[4]. Rather than focusing solely on peak detection performance, we explicitly analyze the sensitivity of the detector to internal decision parameters that directly affect operational trade-offs in real deployments. Building on our previous AADRNN-based AD design [5], we focus in this analysis on key parameters that are often tuned empirically and rarely characterized systematically, including the decision-making threshold used for attack classification and also the choice of values of the data sampling parameters. Larger values of these parameters can increase computational overhead and delay detection decisions, while overly conservative choices may lead to missed attacks. Experimental results show that the AADRNN-based AD maintains high detection performance and low false positive rates within stable operating regions of these parameters, even when relatively small sampling windows are used, which is particularly important for resource-constrained and delay-sensitive IoT environments.

A key advantage of the AADRNN is that it is trained exclusively on benign traffic, which reduces dependence on labeled attack data and avoids attack-pattern leakage into the training process. In addition, it operates on a compact set of packet-level metrics, simplifying feature engineering and reducing runtime overhead. Furthermore, the measured processing time is on the order of a few milliseconds per packet on a simple server [6], reflecting the lightweight nature of the approach. On the other hand, as discussed in Subsection III-B, it provides high accuracy with false positives that are clearly lower than several other baseline detection methods.

The remainder of this paper is organized as follows. First, Section II reviews prior work and highlights gaps related to parameter sensitivity. Next, Section III describes the AADRNN-based AD, while Section IV outlines the experimental setup. Then, Section V presents and analyzes the measurement results. Finally, Section VI concludes the paper and suggests directions for future work.

II. Related Work

The design of internal parameters critically influences the performance of ADs. A key example is proper threshold tuning, which is essential for reliable decision-making. Dynamic thresholds have been reported to significantly reduce false alarms compared to fixed ones [7], performing well across both low and high-rate attacks [8], while threshold optimization also affects accuracy and responsiveness considerably [9], [10]. In [11], the authors confirmed that an adaptive threshold based on Rényi entropy improves resilience across different attack intensities. Moreover, in [12], they found that both

detection sensitivity and robustness are highly dependent on the statistical threshold and window size. Several studies have examined the impact of other internal parameters. For instance, sliding-window entropy methods were applied in [13], showing that detection accuracy, false-positive rate, and timeliness can significantly change with window length, as small windows increase sensitivity and false positives, whereas large windows smooth transient anomalies but increase detection delay. Similarly, it was demonstrated in [14] that the size of the training window strongly affects the generalization and stability of the model whose parameters have been obtained via learning.

Numerous AD models have been proposed using machine learning, deep learning, ensemble methods, and distributed architectures. For evaluation, authors have typically relied on benchmark datasets such as KDD-99, NSL-KDD, CICIDS2017, Kyoto, and WSN-DS. Among these, the Mirai Botnet dataset [15] has been widely adopted, with several approaches reporting high and comparable detection accuracy for various cyberattacks [16]–[18]. Recent top-tier works highlighted complementary detection objectives beyond raw accuracy. For example, HorusEye [19] achieved robust detection under extremely low false-positive budgets by leveraging high-throughput processing and programmable network components. In parallel, recent survey efforts synthesized trends in IoT botnet DDoS attacks and detection techniques, emphasizing challenges related to scalability, deployment feasibility, and robustness under evolving attack behaviors [20].

Other recent studies explore supervised machine learning for Mirai detection on specialized IoT traffic datasets, such as IoT-POT [21], which employed classical classifiers including K-nearest Neighbors (KNN), Decision Trees (DT), and Random Forests (RF). Furthermore, the author in [22] demonstrated that accurate Mirai detection can be achieved directly on resource-constrained IoT devices with low memory usage and inference latency.

On the other hand, it is well known that traffic characteristics have a major impact on network performance [23] and, unsurprisingly, it has been also shown to be the case in the way attacking traffic can compromise, overload, and block the network components that are meant to protect the network-attached equipment and system software, including the attack detection software itself [24] at network gateways. Thus, recent research has focused not only on attack detection, but also on traffic-shaping and attack mitigation [25] to limit the effects of attacks through traffic-shaping and optimum dropping of attacking traffic [26].

Despite this progress, the existing literature primarily focuses on peak detection accuracy under fixed configurations, offering limited insight into how internal design choices affect practical deployment in real systems. In this paper, we analyze the impact of such key internal parameters on the detection performance of an AADRNN-based AD system across different attack scenarios. In addition, we experimentally compare AADRNN against several representative baseline models.

III. The AD and its Performance

The AD system operates on traffic data to compute three specific metrics, as shown on the left-hand-side of Figure 1. The system is based on the Dense Random Neural Network model with Auto-Associative Learning (AADRNN). It processes incoming traffic to obtain a set of M metrics, which are used both to tune the AADRNN parameters on benign traffic exclusively and to subsequently analyze traffic for possible attacks. The learning method is described in Appendix VI.

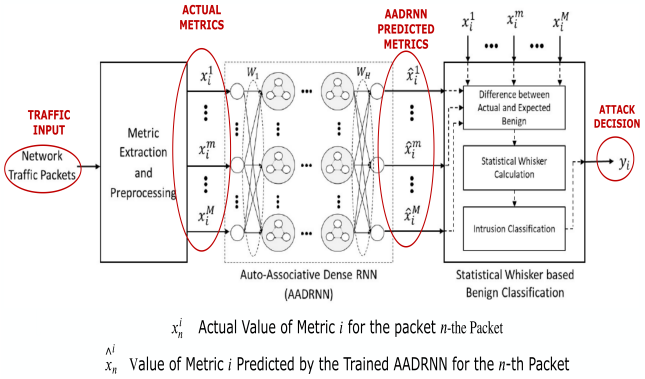


Fig. 1. Structure of the AD system generating a binary decision y_n for the n-th packet using the AADRNN from M network traffic metrics $(x_n^1, ..., x_n^M)$. The AADRNN is an H-layer feedforward network, where each node is a densely coupled neuron cluster. The decision unit compares the network output with its input to classify packets as benign or attacks.

A. Traffic Metrics and Decision Making

The traffic metrics are meant to capture the signatures of DDoS and Botnet attacks. For this purpose, we restrict our attention to three metrics, i.e. $M = 3$, defined as follows: Let I be a fixed positive integer (sliding-window), and T a fixed time duration (in milliseconds). Let p_n denote the n-th packet in chronological order in the dataset $\mathcal{D}$, which is used for training and testing the AD, let t_n denote the time-stamp in its header, which represents the approximate time when the n-th packet is transmitted from its source, and let b_n denote its length in bytes. First, define z_n^1 as the total size of the last I packets up to and including packet n, while z_n^2 is the average inter-transmission time of the last I packets up to and including packet n:

$$z_n^1 = \sum_{j=n-I+1}^{n} b_j, \; z_n^2 = \frac{t_n - t_{n-I}}{I} \; . \tag{1}$$

z_n^3, the total number of packets transmitted in the last T seconds up to the transmission of the n-th packet, is:

$$z_n^3 = \sum_{k=1}^{\infty} \mathbf{1}\big[\, t_n - T \le t_k < t_n \,\big] \;, \; where \tag{2}$$
$$\mathbf{1}\big[X\big] = 1 \; if \; X = true, \; \mathbf{1}\big[X\big] = 0 \; otherwise.$$

Using the dataset $\mathcal{D}_n = \{p_j : \; 1 \le j \le n\}$, these quantities are then normalized:

$$x_n^m = \min\Big[\ 1\ ,\ \frac{z_n^m - \min_{p_j \in \mathcal{D}_n} z_j^m}{\max_{p_j \in \mathcal{D}_n} z_j^m - \min_{p_j \in \mathcal{D}_n} z_j^m}\ \Big]\ . \quad (3)$$

From the output $\hat{x}_n$ of the AADRNN with input x_n, the binary decision variable y_n is obtained using a threshold $0 < \gamma < 1$:

$$y_n = 1\ (ATTACK)\ if\ \frac{\sum_{m=1}^{3} |x_n^m - \hat{x}_n^m|}{3} \geq \gamma, \quad (4)$$
$$y_n = 0\ (NOT - ATTACK),\ otherwise.$$

B. Baseline Comparison and Cost–Performance Analysis

To contextualize the performance of the AADRNN-based detector, we compare it against several representative baseline models under identical experimental conditions. Specifically, we used Isolation Forest (IF) and a Vanilla Autoencoder (AE) as standard unsupervised baselines, operating on the same three packet-level traffic metrics and evaluated using the same training and testing splits. All models were trained using 10% of benign traffic from the Mirai dataset, while the remaining benign traffic and all attack traffic were used for testing. We use three standard metrics (expressed as percentages): $Accuracy$, which measures overall classification correctness; True Positive Rate (TPR), which reflects the ability to correctly detect attacks; and False Positive Rate (FPR), which quantifies the proportion of normal packets incorrectly classified as attacks. These metrics are defined as follows:

$$Accuracy = (\frac{TP + TN}{TP + TN + FP + FN}) \times 100\%, \quad (5)$$
$$TPR = (\frac{TP}{TP + FN}) \times 100\%, \quad (6)$$
$$FPR = (\frac{FP}{FP + TN}) \times 100\%, \quad (7)$$

where TP, TN, FP, and FN denote the number of true positives, true negatives, false positives, and false negatives, respectively.

The results in Table I show that the AADRNN consistently outperforms standard unsupervised baselines in terms of accuracy and attack detection effectiveness. Compared to IF, which relies on isolating individual anomalous points, the AADRNN achieves a significantly higher TPR under sustained botnet attacks, while simultaneously exhibiting a lower FPR, indicating improved discrimination between attack and benign traffic. Relative to AE, the AADRNN also provides improved detection performance, while maintaining comparable training and inference costs.

The table additionally reports the performance of a supervised Long Short-Term Memory (LSTM)-based detector trained on benign traffic augmented with a limited set of 10k labeled attack samples. While the LSTM attains slightly higher detection rates, the performance gap with the AADRNN is marginal. In contrast, the LSTM incurs substantially higher training and inference overhead, and requires labeled attack data, which may not always be available in practice. These results highlight that the AADRNN can approach the detection performance of LSTM, while offering significantly lower computational cost and greater suitability for lightweight and cold-start IoT deployments.

IV. Experimental Setup and Network Traffic Sources

We adopted a two-pronged evaluation strategy. First, we designed a scalable IoT-based experimental test-bed, illustrated in Figure 2 (Below), consisting of two IoT devices represented by Raspberry Pi 4 Model B Rev. 1.2 and a Server interconnected via an Ethernet switch. Each RPi is equipped with a 1.5 GHz ARM Cortex-A72 quad-core CPU and 2 GB LPDDR4 RAM, running Raspbian GNU/Linux 11. The Server, responsible for packet receiving, processing, and logging, is equipped with an Intel Core i7 (8 cores, 3.10 GHz) processor and 16 GB RAM, running Ubuntu Linux 5.15.0-60-generic. User Datagram Protocol (UDP) was used due to its suitability for real-time data transmission. Second, we evaluated the AD using the widely studied Mirai Botnet Kitsune dataset [15], which contains traffic generated by multiple compromised devices over diverse protocols (TCP, UDP, ARP, ICMP).

In the test-bed experiments, in the absence of an attack, both RPis were programmed to transmit normal traffic, periodically sending their CPU temperatures to the Server at fixed UDP rates specified within the context of each experiment. In the attack scenario, attacks were emulated using the open-source MHDDoS toolset [27], which supports 56 attack types and configurable parameters such as destination IP, thread count, attack duration, and packet size. This flexibility allowed the generation of high-rate flooding traffic targeting bandwidth saturation and service disruption. During evaluation, RPi2 was used solely to generate normal traffic, while RPi1 acted as a compromised node. It first transmitted normal packets and then launched a UDP flood attack using fixed-size packets (1032 bytes) at high transmission rates. MHDDoS provides

TABLE I
COMPARISON OF AADRNN WITH BASELINE DETECTION MODELS ON THE MIRAI BOTNET DATASET.

AD Model	Accuracy (%)	TPR (%)	FPR (%)	Training Time (CPU)	Inference Time / Packet
Isolation Forest (IF)	97.49	96.92	1.30	$5 - 30$ s	$0.05 - 0.3$ ms
Vanilla Autoencoder (AE)	98.71	98.62	0.84	$1 - 5$ min	$0.3 - 2$ ms
LSTM (Supervised)	99.53	99.59	0.77	$5 - 30$ min	$2 - 10$ ms
AADRNN	99.34	99.31	0.51	$1 - 5$ min	$0.5 - 3$ ms

fixed control over packet size. We chose a large packet size to maximize the flood's effect by overwhelming the Server's input buffer and thereby paralyzing its operation, while also enabling us to examine scenarios that are distinct from Mirai attacks, which employ varying packet sizes and transport protocols.

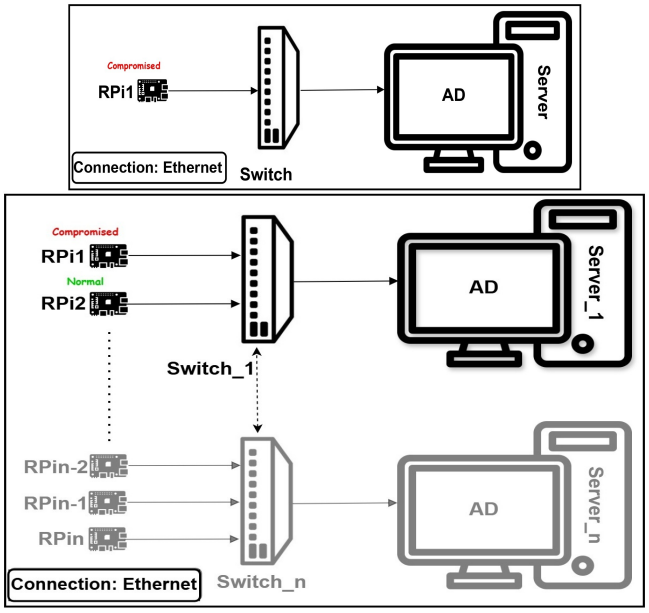


Fig. 2. Above, we show a simple network architecture with a single device (RPi1) connected to the Server via a switch; here RPi1 acts as a compromised device that transmits normal traffic and later switches to launching a UDP flood attack with a very high transmission rate. Below, we show a scalable network architecture where a large number of IoT devices communicate with multiple Servers via network switches. In this case, our experiments connect two devices (RPi1 and RPi2) to Server_1. RPi2 continuously sends normal traffic, while the compromised RPi1 initially transmits normal data before launching a UDP flood attack.

V. Performance Measurements and Discussion

In this section, we evaluate the AD through a series of controlled experiments under different traffic conditions using the performance metrics defined in Subsection III-B. Experiments are conducted using both a simple and a more complex network architecture to analyze the AD behavior before, during, and after flood attacks. In the first preliminary experiment, a simple network architecture was used in which RPi1 transmitted normal UDP packets to the Server before launching a UDP flood attack, as shown in Figure 2 (Above). The normal traffic consisted of CPU temperature measurements sent once per second. After 100 seconds of benign operation, RPi1 initiated a UDP flood attack using the MHDDoS toolset [27], transmitting approximately 15,000 packets/s for 10 consecutive seconds, resulting in a backlog of about 153,000 packets at the Server. During this experiment, the AD achieved an $Accuracy$ and TPR of 99.99%. Moreover, all 99 normal packets transmitted before the attack were correctly classified, resulting in an FPR of 0.0%. Immediately after the attack, 38 normal packets were briefly misclassified before the system stabilized.

A second experiment was conducted using the modified network architecture shown in Figure 2 (Below), where both RPi1 and RPi2 transmitted normal traffic at a higher rate of 100 packets/s. After 100 seconds, RPi1 launched a 10-second flood attack. Before the attack, the AD correctly classified all 20,000 normal packets, achieving an FPR of 0.0%. During the attack, the AD achieved an $Accuracy$ of 98.84%, a TPR of 99.42%, and an FPR of 96.53%, corresponding to 35 correctly classified normal packets out of approximately 1,000 received during the attack. After the attack, the AD rapidly stabilized, with 80 normal packets initially misclassified, resulting in an FPR of 0.40% over the subsequent 20,000 packets. This is a reasonable and expected outcome at this level, considering that the Server continued to receive a high packet rate (100 packets/s from each device), and the transition back to normal operation inherently requires time due to the processing characteristics of neural networks.

We further evaluated the AD using the Mirai Botnet dataset, which contains 764,137 labeled samples generated over multiple protocols from different compromised sources. The model was trained using only 10% of the benign traffic and tested on the remaining benign samples together with all attack traffic. This evaluation achieved an $Accuracy$ of 99.34%, a TPR of 99.31%, and an FPR of 0.51%, demonstrating reliable detection performance with a low false alarm rate.

A. Impact of Threshold Variation on AD Performance

We varied the decision threshold γ in the range $0 < \gamma < 1$, as shown in Figure 3 and Table II. For $0 \leq \gamma \leq 0.28$, the AD showed consistently stable performance. TPR remained above 99.0% (from 99.93% down to 99.30%), while FPR decreased from 1.03% to 0.48%, indicating that increasing γ progressively reduced false alarms without compromising attack detection. This range represents the optimal operating region, where high detection accuracy is maintained alongside a low false positive rate.

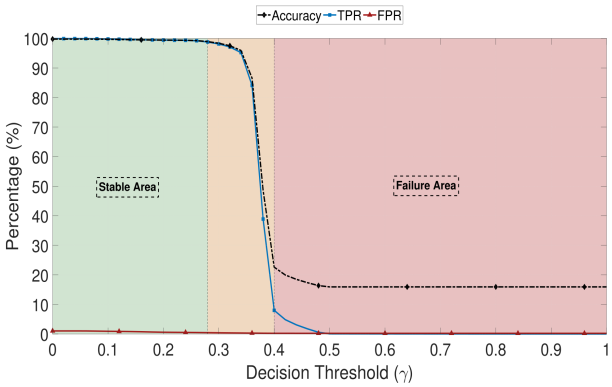


Fig. 3. AD performance in terms of $Accuracy$, TPR and FPR with different values of γ. The stable operational region is between $0 < \gamma < 0.28$.

When γ increased to 0.29–0.37, both $Accuracy$ and TPR declined noticeably to around 90.0%, while FPR continued to decrease slightly. This behavior indicates that the AD

TABLE II
AD PERFORMANCE WITH DIFFERENT VALUES OF γ.

γ	**Accuracy (%)**	**TPR (%)**	**FPR (%)**
0.05	99.78	99.93	1.03
0.10	99.70	99.81	0.91
0.15	99.54	99.60	0.79
0.20	99.43	99.44	0.59
0.25	99.34	99.31	0.51
0.30	98.36	98.12	0.40
0.35	94.70	93.75	0.31
0.40	22.60	7.99	0.23
0.45	17.92	2.42	0.22
0.50	15.93	0.05	0.22

started missing a larger fraction of attack packets as the decision boundary became stricter, despite producing fewer false alarms. At $\gamma = 0.40$, performance collapsed, with $Accuracy$ dropping to 22.60% and TPR to 7.99%. For $\gamma \geq 0.40$, TPR remained close to 0%, while FPR stayed below 0.23%, indicating that the AD essentially stopped detecting attacks and converged to trivial classifications of no practical use. These patterns were consistent across both the test-bed and the Mirai Botnet dataset, showing that the threshold effect is independent of attack type. Based on these measurements, we fixed $\gamma = 0.25$ for all experiments reported in this paper, as it provides the best balance between high attack detection performance and a low false positive rate.

B. Effect of Parameter I on AD Performance

The AD applies a sliding-window that processes successive packets sequentially and classifies them as either "ATTACK" or "NOT-ATTACK" using the traffic metrics described earlier. The internal parameter I plays a critical role in this decision process. In this subsection, we vary the value of I to analyze its impact on performance. The test-bed experiments were conducted using the complex setup described previously, and each experiment was repeated 10 times to ensure consistency.

Table III summarizes the findings. Before any attack, the AD correctly classified all normal packets for every value of I, resulting in an FPR of 0.0%. During the attack, performance improved steadily as I increased: $Accuracy$ rose from 77.61% at $I = 1$ to 99.22% at $I = 100$, while TPR increased from 78.11% to 99.81%. At the same time, the FPR decreased noticeably as I increased, indicating a reduction in false alarms on normal packets. However, when I exceeded 400, both $Accuracy$ and TPR declined sharply, as the AD's view of the traffic became overly smoothed, making it less sensitive to sudden changes and delaying attack detection. Meanwhile, FPR decreased slightly for larger values of I, reducing the loss of normal traffic during attack periods. After the attack, the FPR continued to decrease further, reaching 0.13% at $I = 100$.

A similar pattern was observed on the Mirai Botnet dataset, as shown in Table IV, where $Accuracy$ and TPR increased with larger values of I and peaked around $I = 500$, before dropping significantly for larger values. This difference in the critical value of I relative to the test-bed experiments can be attributed to differences in traffic characteristics. The high-rate UDP flood attack generates highly bursty traffic with rapid temporal variations, making detection more sensitive to large window sizes, whereas the Mirai dataset exhibits more gradual and heterogeneous patterns, allowing the system to tolerate slightly larger values of I.

These results confirm that relatively small window sizes, roughly in the range of $I = 10$–20, can provide highly comparable detection performance while maintaining low false alarm rates, without increasing detection delay or computational overhead.

VI. CONCLUSIONS AND FUTURE WORK

Network-connected systems increasingly demand robust protection against cyberattacks, driving extensive research on AD methods, many of which rely on machine learning and require careful training and validation. However, their performance is affected by multiple internal parameters which are seldom mentioned and studied. In this work, we analyze an AADRNN-based AD system trained on benign traffic and systematically evaluate how key internal parameters and traffic metrics influence its performance. Our findings provide practical insight for adapting the system to different deployment contexts and highlight the value of detailed parameter sensitivity studies in guiding the design and improvement of AD systems. In future work, this analysis can be extended to other detection models and parameters, and to different attack types in real-world deployments.

ACKNOWLEDGMENT

This research was supported by the European Commission H2020 Program through the DOSS Research and Innovation Action, Grant Agreement No. 101120270.

APPENDIX: AADRNN LEARNING

The AADRNN is an extension of the Random Neural Network [28] with soma-to-soma triggering between neurons in any cluster [2], in addition to the usual excitatory and inhibitory weights. As a result, when the size of each neuronal cluster in the layers becomes large, the neuronal cluster activation function is:

$$\zeta(\lambda) = \frac{[r(1-p) - p\lambda^+]\Big[1 \pm \sqrt{1 - \frac{4p(\lambda+\lambda^-)[\lambda^+ - r - \lambda - \lambda^-]}{r(1-p)-p\lambda^+}}\Big]}{2p(\lambda + \lambda^-)},$$

where r is the total firing rate of each neuron in a cluster, λ^+ and λ^- are the external excitatory and inhibitory spike rates arriving at each neuron, and p is the probability that any other neuron in the cluster fires when a given neuron fires, representing soma-to-soma interactions. These parameters are set to: $r = 0.001$, $\lambda^+ = \lambda^- = 0.1$, $p = 0.05$, and $T = 100$ms.

In the present case, the AADRNN is organized in $l \in \{1, \ldots, H\}$ feed-forward layers (see Figure 1), with $H = 3$.

TABLE III
PERFORMANCE EVALUATION OF THE AD USING DIFFERENT VALUES OF THE PARAMETER I IN THE TEST-BED EXPERIMENTS.

Value of I	Before Attack		During Attack				After Attack	
	No. of Packets	FPR (%)	No. of Packets	Accuracy (%)	TPR (%)	FPR (%)	No. of Packets	FPR (%)
1	20000	0.00	153764	77.61	78.11	98.88	20000	34.11
5	20000	0.00	154210	92.16	92.74	97.47	20000	4.88
6	20000	0.00	153953	93.39	93.81	97.28	20000	3.98
7	20000	0.00	158465	95.83	95.88	97.09	20000	3.09
8	20000	0.00	153944	98.09	98.20	96.91	20000	1.63
9	20000	0.00	156892	98.41	99.03	96.70	20000	0.82
10	20000	0.00	158984	98.84	99.42	96.53	20000	0.40
15	20000	0.00	148276	98.81	99.45	95.33	20000	0.33
20	20000	0.00	152832	98.87	99.49	94.80	20000	0.27
25	20000	0.00	158379	98.96	99.55	93.99	20000	0.20
50	20000	0.00	152621	99.02	99.63	93.15	20000	0.18
75	20000	0.00	150988	99.10	99.71	92.39	20000	0.16
100	20000	0.00	154536	99.22	99.81	91.45	20000	0.13
250	20000	0.00	161020	99.29	99.85	89.76	20000	0.09
300	20000	0.00	151276	94.88	95.44	89.03	20000	0.08
400	20000	0.00	155878	91.26	91.78	88.67	20000	0.08
500	20000	0.00	150398	17.36	16.88	3.06	20000	0.07
600	20000	0.00	150439	12.89	12.34	1.72	20000	0.07
700	20000	0.00	157843	12.02	11.50	1.12	20000	0.07
800	20000	0.00	152186	11.56	10.99	0.88	20000	0.07
900	20000	0.00	155287	10.77	10.21	0.35	20000	0.07
1000	20000	0.00	151220	7.29	6.65	0.09	20000	0.09

TABLE IV
PERFORMANCE EVALUATION OF THE AD FOR VARYING VALUES OF THE PARAMETER I ON THE MIRAI BOTNET DATASET.

Value of I	Accuracy (%)	TPR (%)	FPR (%)
1	82.13	82.91	21.99
5	98.91	98.85	0.80
6	98.99	98.94	0.74
7	99.08	99.03	0.68
8	99.17	99.13	0.63
9	99.29	99.26	0.55
10	99.34	99.31	0.51
15	99.40	99.37	0.46
20	99.46	99.43	0.40
25	99.52	99.48	0.27
50	99.64	99.61	0.23
75	99.73	99.72	0.20
100	99.80	99.79	0.18
250	99.81	99.80	0.17
300	99.82	99.81	0.17
400	99.82	99.81	0.16
500	99.83	99.82	0.15
600	88.87	86.78	0.12
700	66.42	60.08	0.11
800	52.61	43.66	0.10
900	28.92	15.49	0.10
1000	23.93	9.55	0.08

Weight matrices W_l connect the clusters of layer l to those of layer $l+1$, and are learned to create the desired auto-associative memory. For the input vector $x_n = (x_n^1, x_n^2, x_n^3)$, the forward pass of the AADRNN is:

$$\hat{x}_{n,l} = \zeta([\hat{x}_{n,l-1}, 1]\, W_{l-1}),\ 1 \le l \le H,\ \hat{x}_n = \hat{x}_{n,L-1} W_{H-1},$$

where $\hat{x}_{n,l} = (\hat{x}_n^1, \hat{x}_n^2, \hat{x}_n^3)$ is the output vector of layer l for input x_n, $\hat{x}_{n,0} = x_n$, and $[\hat{x}_{n,l}1]$ indicates that 1 is concatenated to the output of each layer l as a multiplier of the bias. Since we use auto-associative learning, only benign traffic is used to learn the weights W_l, and the well-known Fast Iterative Shrinkage-Thresholding Algorithm (FISTA) [29] is used for optimization:

$$W_l = \arg \min_{\{W:\, W \ge 0\}} \qquad (8)$$
$$[\ ||adj(\zeta(\hat{X}_{l-1}^{\text{train}} W_R))W - \hat{X}_{l-1}^{\text{train}}||_{H_2}^2 + ||W||_{H_1}],$$

where $\hat{X}_l^{\text{train}}$ is the matrix of outputs of layer l resulting from the training dataset $\mathcal{D}_{\text{train}}$:

$$\hat{X}_l^{\text{train}} = \{\hat{x}_i^l\}_{i \in \mathcal{D}_{\text{train}}}. \qquad (9)$$

To avoid bias in the AD decision, only benign packets were used for training in all experiments, while testing combined the remaining benign and attack packets. This strict separation prevents attack patterns from leaking into training, which would otherwise inflate performance through memorization rather than genuine anomaly detection. In the experiments reported in this paper, AADRNN learning is carried out separately for the two following cases:

1) Test-bed Experiments: A small dataset consisting of the first 5000 UDP packets received by the Server was used for training during the "cold-start" phase, which contained only benign traffic. The duration of this phase depends on the packet arrival rate (1–5 minutes). Larger datasets can further improve accuracy by capturing richer patterns and reducing overfitting, up to a saturation point beyond which additional packets offer little benefit. We intentionally used a small dataset to

demonstrate the AD's effectiveness under limited training data, while also taking into account training time and computational overhead as much as possible.

2) Mirai Botnet dataset [15]: Of the total 764,137 packets, 121,621 are benign; 10% of these benign packets were used for training, while the remaining 90% were combined with attack traffic for testing. Mirai provides detailed packet logs with 118 attributes (e.g., packet number, packet length, flow frequency, timestamp, source/destination, protocol). It exemplifies large-scale IoT malware, where compromised devices are leveraged to launch coordinated DDoS attacks [30], producing characteristic traffic such as repetitive scanning and brute-force login attempts with hardcoded credentials.

REFERENCES

[1] J. G. Almaraz-Rivera, J. A. Perez-Diaz, and J. A. Cantoral-Ceballos, "Transport and application layer ddos attacks detection to iot devices by using machine learning and deep learning models," *Sensors*, vol. 22, no. 9, p. 3367, 2022.

[2] E. Gelenbe and Y. Yin, "Deep learning with random neural networks," in *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2016, pp. 1633–1638.

[3] ——, "Deep learning with dense random neural networks," in *Man-Machine Interactions 5*. Cham: Springer International Publishing, 2018, pp. 3–18.

[4] O. Brun, Y. Yin, and E. Gelenbe, "Deep learning with dense random neural network for detecting attacks against iot-connected home environments," *Procedia Computer Science*, vol. 134, pp. 458–463, 2018.

[5] M. Nasereddin, M. Nakıp, and E. Gelenbe, "Measurement based evaluation and mitigation of flood attacks on a lan test-bed," in *The 48th IEEE Conference on Local Computer Networks (LCN) October 1-5, 2023, Daytona Beach, Florida, USA*. IEEEXpress, 2023, pp. 1–4.

[6] E. Gelenbe and M. Nasereddin, "Protecting iot servers against flood attacks with the quasi deterministic transmission policy (best paper award)," in *2023 IEEE 22nd International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. IEEE, 2023, pp. 379–386.

[7] F. M. Salem, H. Youssef, I. Ali, and A. Haggag, "A variable-trust threshold-based approach for ddos attack mitigation in software defined networks," *Plos one*, vol. 17, no. 8, p. e0273681, 2022.

[8] Z. Hemmati, G. Mirjalily, and Z. Mohtajollah, "Entropy-based ddos attack detection in sdn using dynamic threshold," in *2021 7th International Conference on Signal Processing and Intelligent Systems (ICSPIS)*. IEEE, 2021, pp. 1–5.

[9] K. Singh, K. S. Dhindsa, and D. Nehra, "T-cad: A threshold based collaborative ddos attack detection in multiple autonomous systems," *Journal of Information Security and Applications*, vol. 51, p. 102457, 2020.

[10] H. Qiu, T. Dong, T. Zhang, J. Lu, G. Memmi, and M. Qiu, "Adversarial attacks against network intrusion detection in iot systems," *IEEE Internet of Things Journal*, vol. 8, no. 13, pp. 10 327–10 335, 2020.

[11] M. A. Aladaileh, M. Anbar, A. J. Hintaw, I. H. Hasbullah, A. A. Bahashwan, and S. Al-Sarawi, "Renyi joint entropy-based dynamic threshold approach to detect ddos attacks against sdn controller with various traffic rates," *Applied Sciences*, vol. 12, no. 12, p. 6127, 2022.

[12] J. Wang, L. Wang, and R. Wang, "A method of ddos attack detection and mitigation for the comprehensive coordinated protection of sdn controllers," *Entropy*, vol. 25, no. 8, p. 1210, 2023.

[13] G. Baldini, "On the application of entropy measures with sliding window for intrusion detection in automotive in-vehicle networks," *Entropy*, vol. 22, no. 9, p. 1044, 2020.

[14] J. G. Medina-Arco, R. Magán-Carrión, R. A. Rodríguez-Gómez, and P. García-Teodoro, "Methodology for the detection of contaminated training datasets for machine learning-based network intrusion-detection systems," *Sensors*, vol. 24, no. 2, p. 479, 2024.

[15] "Kitsune Network Attack Dataset," August 2020. [Online]. Available: https://www.kaggle.com/ymirsky/network-attack-dataset-kitsune

[16] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection," in *The Network and Distributed System Security Symposium (NDSS) 2018*, 2018, pp. 274–288.

[17] A. Alabdulatif, S. S. H. Rizvi, and M. A. Hashmani, "Optimal machine learning models for kitsune to detect mirai botnet malware attack," *Journal of Hunan University Natural Sciences*, vol. 48, no. 6, p. 91, 2021.

[18] E. Y. Güven and Z. Gürkas-Aydin, "Mirai botnet attack detection in low-scale network traffic." *Intelligent Automation & Soft Computing*, vol. 37, no. 1, p. 419, 2023.

[19] Y. Dong, Q. Li, K. Wu, R. Li, D. Zhao, G. Tyson, J. Peng, Y. Jiang, S. Xia, and M. Xu, "Horuseye: A realtime iot malicious traffic detection framework using programmable switches," in *32nd USENIX Security Symposium (USENIX Security'23)*, 2023, pp. 571–588.

[20] M. Gelgi, Y. Guan, S. Arunachala, M. Samba Siva Rao, and N. Dragoni, "Systematic literature review of iot botnet ddos attacks and evaluation of detection techniques," *Sensors*, vol. 24, no. 11, p. 3571, 2024.

[21] A. Sharma and H. Babbar, "Iot-pot: Machine learning-based detection of mirai botnet attacks in iot," in *2024 First International Conference on Innovations in Communications, Electrical and Computer Engineering (ICICEC)*. IEEE, 2024, pp. 1–6.

[22] Z. Alwaisi, "Memory-efficient and robust detection of mirai botnet for future 6g-enabled iot networks," *Internet of Things Journal*, p. 101621, 2025.

[23] E. Gelenbe, X. Mang, and R. Önvural, "Diffusion based statistical call admission control in atm," *Performance Evaluation*, vol. 27, pp. 411–436, 1996.

[24] J. Bergquist, E. Gelenbe, M. Nasereddin, and K. Sigman, "Mitigating massive access with quasi-deterministic transmission: Experiments and stationary analysis," *Performance Evaluation*, vol. 170, no. 2025, p. 102512, 2025.

[25] E. Gelenbe and M. Nasereddin, "Data driven optimum cyberattack mitigation," in *IEEE DSAA'25: Data Science and Advanced Analytics Conference, Birmingham, UK*. IEEE, 2025, pp. 1–8.

[26] ——, "Adaptive attack mitigation for iov flood attacks," *IEEE Internet of Things Journal*, vol. 12, no. 5, p. 4701, 2025.

[27] MHDDoS, "DDoS Attack Script With 56 Methods," May 2022. [Online]. Available: https://github.com/MatrixTM/MHDDoS

[28] E. Gelenbe, "Random neural networks with negative and positive signals and product form solution," *Neural computation*, vol. 1, no. 4, pp. 502–510, 1989.

[29] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM journal on imaging sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[30] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis *et al.*, "Understanding the mirai botnet," in *26th USENIX security symposium (USENIX Security'17)*, 2017, pp. 1093–1110.