G-Networks Can Detect Different Types of Cyberattacks

Erol Gelenbe
Institute of Theoretical and Applied Informatics
Polish Academy of Sciences, IITIS-PAN
44-100 Gliwice, Poland
& Lab. I3S CNRS, Université Côte d'Azur,
Grand Château, 06103 Nice Cecex 2, France
& Yaşar University
Bornova, Izmir, Turkey
ORCID: 0000-0001-9688-2201

Mert Nakıp
Institute of Theoretical and Applied Informatics
Polish Academy of Sciences, IITIS-PAN
44-100 Gliwice, Poland
ORCID: 0000-0002-6723-6494

***Abstract*—Malicious network attacks are a serious source of concern, and machine learning techniques are widely used to build Attack Detectors with off-line training with real attack and non-attack data, and used online to monitor system entry points connected to networks. Many machine learning based Attack Detectors are typically trained to identify specific types attacks, and the training of such algorithms to cover several types of attacks may be excessively time consuming. This paper shows that G-Networks, which are queueing networks with product form solution and special customers such as negative customers and triggers, can be trained just with "non-attack" traffic, can accurately detect several different attack types. This is established with a special case of G-Networks with triggerred customer movement. A DARPA attack and non-attack traffic repository is used to train and test the the G-Network, yielding comparable or clearly better accuracy than most known attack detection techniques.**

***Index Terms*—Gelenbe-Networks (G-Networks), Multiple Attack Detection, Random Neural Networks, Queueing Networks with Negative and Positive Customers, Auto-Associative Deep Random Neural Network**

I. Introduction

THE edge of the Internet is populated with a wide range of devices that are part of Mobile Networks, the Internet of Things (IoT) as well as various servers and local area networks. Since 50% or more of these devices [1] are simple and hence of low-cost and low-maintenance, it is difficult (if not totally impossible) to burden them with complex security functionalities [2], and are prime targets for cyberattacks [3]–[5], including simple Denial of Service (DoS) attacks [6] which seriously interfere with the operation of numerous devices that have few resources to spare [7], [8]. Systemic approaches to securing cyber-physical systems have been suggesed [9], [10], but these may not be suitable for highly distributed systems composed of simple devices.

More harmful Distributed DoS (DDoS) attacks [11] can overwhelm large networks by using proxy victims that have been attacked and turned into "Bots". In turn, they flood devices with attacks and overwhelming traffic, such as in the "Mirai" attack [12], and compromising Netflix, Reddit, Spotify, and Twitter [13], [14], and harming millions of IP nodes [15], [16].

Thus much research has been conducted in designing Attack Detectors (ADs) using conventional statistics or Machine Learning (ML). These can be trained on-line, or off-line with validated instances of non-attack and attack data collected during long usage periods or during substantial network attacks. The trained AD is then used online to monitor IP addresses and network ports, so as to raise an alarm when malicious incoming traffic is detected.

Because of the severe effects of Botnet attacks, much work has addressed their characteristics [12], [17] and Mirai attack source code was also studied [18]. The detection of Mirai attacks from incoming traffic has used different ML methods including K-Nearest Neighbours (KNN), Support Vector Machines (SVM), Decision Trees (DT), Multi-Layer Perceptrons (MLP) [19], Classification and Regression Trees [20], Gradient Boosting and Random Forests [21], Deep MLPs [22], Long-Short Term Memory [23], [24], and their comparison was also conducted [25]. Related research includes [26]–[29], and adaptive network routing to avoid nodes or paths that are subject to unusal events or an attack [30], [31] has also been investigated.

G-Networks [32] are stochastic queueing networks with product form solution, and a special case of this model, known as the Random Neural Network (RNN) [33]–[35] has been shown to be effective in detecting denial of service SYN attacks [36], after being trained via gradient descent learning with both "normal" and attack data.

This is a consequence of an important mathematical property that we exploit in this paper, namely the ability of G-Networks to approximate all continuous and bounded functions, with arbitrarily close error values that depend on the size of the network in number of queues or neurons [37], [38]. On the empirical side of things, the RNN has been very

This research has been supported by the European Commission H2020 Program under the IoTAC Research and Innovation Action, under Grant Agreement No. 952684.

successfully used previously by several authors, both for attack or intrusion detection [39], [40], and in many other application areas where machine learning is often used [41]–[49].

Another special case of G-Networks, known as a Deep Random Neural Network (DRNN) [50], when trained with only "normal" (i.e., non-attack data) in Auto-Associative mode, so that its output with "normal" non-atack data matches closely the input data [51], can effectively recognize SYN attacks [52]. In other recent work [53], it has again been shown that this Auto-Associative Deep Random Neural Network (AADRNN), provides very accurate Botnet attack detection despite being trained with only "normal" (non-attack) traffic. In the case of Mirai Botnet attack detection, the AADRNN's performance was compared against three well-known state-of-the-art machine learning techniques: Least Absolute Shrinkage and Selector Operator (Lasso), K-Nearest Neighbors Regressor (KNN), and Multi-Layer Perceptron (MLP), showing that AADRNN performs significantly better, resulting in the following standard metrics: Balanced Accuracy 99.84%, Sensitivity 99.82% and Specificity 99.98%.

Thus in this paper we extend previous work [52], [53] to evaluate whether the AADRNN – using auto-associative training with only normal traffic, i.e., without the use of attack data – can actually detect a wide spectrum of attacks. In Section II we describe the structure of the machine learning model that we use. Section III discusses preprocessing and the post-processing (decision) phase related to the KDD dataset [54] that is used for learning and testing. Section IV uses this well established attack dataset that contains multiple instances of 41 types of attacks, to provide experimental results that indicate that the AADRNN achieves very accurate detection for many – but not all – of these numerous attack types.

Two variants of the AADRNN are used, both with three layers, but one with a $40-20-21$ structure shown in Figure 2, while the second uses a "rectangular" structure as shown in Figure 1. We see that the latter provides results that are somewhat less accurate than the former, while the former structure appears to outperform most other attack detection methods that have recently been discussed in the literature.

II. The AADRNN

The central part of the method we use is composed of an AADRNN that is structured in n**-neuron clusters** of densely coupled neuronal cells that are represented as G-Networks as detailed in this section. They are shown schematically in Figure 1.

The specific G-Network used has "triggers" [32], [55], where each "queue" represents a single neuron. All of these neurons or queues are satistically identical and if a queue's queue length is positive (i.e., that neuron is "excited"), then it fires a spike (sends a "trigger") after an exponentially distributed random interval of parameter r to the set of other queues (neurons) in the same cluster. The trigger will:

- Either cause a drop by -1 of the receiving queue's length, and an immediate transmission of another trigger by the receiving neuron to some other neuron in the cluster with probability p, creating another identical step in the chain or events,
- Or the receiving queue (neuron) absorbs the arriving trigger with probability $(1-p)$ and it's queue length increases by $+1$.
- The target neuron of an arriving trigger is selected at random with equal probability $\frac{1}{n}$.
- In addition to the triggers, each queue (neuron) in a cluster receives an external Poisson flow of excitatory spikes at rate λ_{fl}, and a Poisson flow of inhibitory spikes at rate λ_{fl}. An arriving excitatory spike increases by one $(+1)$ the length of the receiving queue; the arrival of an inhibitory spike reduces the queue length by one (-1) if the queue length is positive, and otherwise have no effect.
- All queues (neurons) in the cluster of layer l (column, going left to right), in row number f as shown in Figure 1, will receive inhibitory inputs (negative customers) of spikes from all neurons in the previous layer: $\sum_{g=1}^{\mathcal{F}_{l-1}} w^-_{g,f,l-1} q_{g,l-1}$, where $\mathcal{F}_l$ is the number of clusters in layer l, and $w^-_{g,f,l-1} \geq 0,\ 1 \leq g \leq \mathcal{F}_{l-1}$.
- All n neurons in a given cluster are statistically identical, and their probability of excitation (probability that the queue length is positive) is denoted $q_{f,l}$.

Based on this description, $q_{f,l}$ is expressed as [32]:

$$q_{f,l} = \tag{1}$$

$$\frac{\Lambda_{fl} + \frac{r'_{fl}\ q_{f,l}(n-1)(1-p)}{n} S_{f,l}(p,n)}{r_{fl} + \lambda^-_{fl} + \frac{r'_{fl} q_{f,l}(n-1)p}{n} S_{f,l}(p,n)}, \tag{2}$$

$$where\ r_{fl} = r'_{fl}\ + \sum_{g=1}^{\mathcal{F}_l}\ w^-_{f,g,l}\ \ , \tag{3}$$

$$\lambda^-_{fl} = \lambda_{fl} + \sum_{g=1}^{\mathcal{F}_{l-1}} [w^-_{g,f,l-1} \times q_{g,l-1}],\ and \tag{4}$$

$$S_{f,l}(p,n) = \sum_{i=0}^{\infty} \big[\frac{q_{f,l}p(n-1)}{n}\big]^i$$

$$= \frac{n}{n - q_{f,l}p(n-1)}. \tag{5}$$

$$Thus:$$

$$q_{f,l} = \frac{\Lambda_{fl} + \frac{r'_{fl}\ q_{f,l}(n-1)(1-p)}{n-q_{f,l}p(n-1)}}{r_{fl} + \lambda^-_{fl} + \frac{r'_{fl}\ q_{f,l}(n-1)p}{n-q_{f,l}p(n-1)}}\ . \tag{6}$$

Since $q_{f,l}$ is a probability, its maximum allowable value is 1. Because one can show that it is an increasing function of Λ_{fl}, it follows that:

$$\Lambda_{fl} \leq r_{fl} + \lambda^-_{fl} - \frac{r'_{fl}(n-1)(1-2p)}{n-p(n-1)}, \tag{7}$$

so that to guarantee that $q_{f,l} \leq 1$ it suffices to set:

$$\Lambda_{fl} \leq r'_{fl} + \lambda_{fl} - \frac{r'_{fl}(n-1)(1-2p)}{n-p(n-1)}. \tag{8}$$

The equations (1) to (8) represent the multiple layer AADRNN as a single nework, where inhibitry weights project

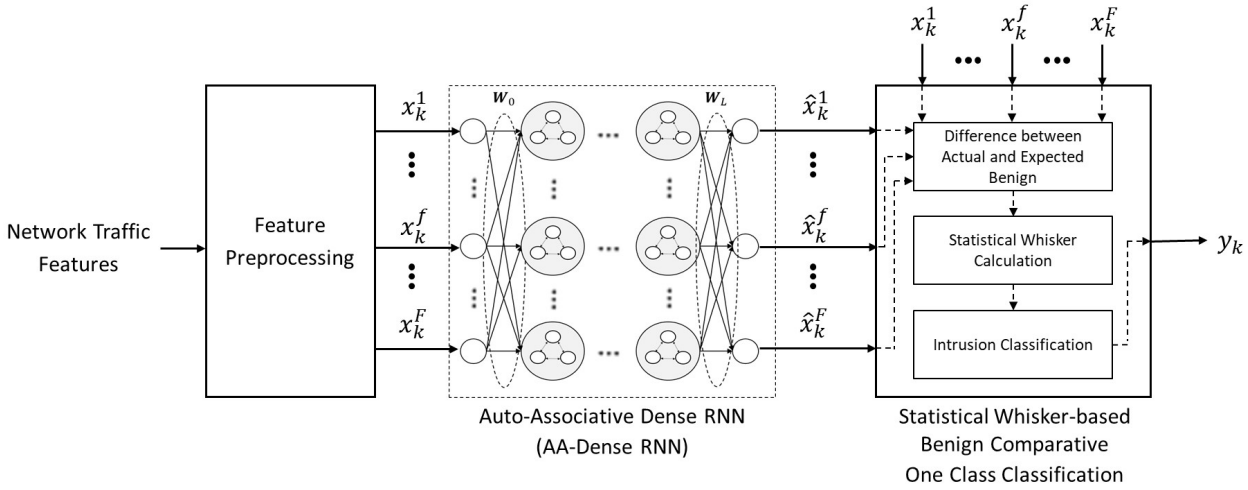

Fig. 1: Overall architecture of the AADRNN for Attack Detection with the KDD [54] Dataset.

from each cell in a given layer to the cells in the next layer. However our computations are based on a simplification, where each cluster of cells in a given layer is a RNN that receives an inhibitory input from clusters in the previous layer, but does not send out spikes to the subsequent layer so that it does not project into the subsequent layer, (3) becomes $r_{fl} = r'_{fl}$. The projections from a layer to a subsequent layer are then treated as a numerical transformation of the outputs of each cluster, providing a numerical value to each λ^-_{fl}, while:

$$q_{fl} = \frac{\Lambda_{fl} + \frac{r_{fl}\ q_{f,l}(n-1)(1-p)}{n - q_{f,l}p(n-1)}}{r_{fl} + \lambda^-_{fl} + \frac{r_{fl}\ q_{f,l}(n-1)p}{n - q_{f,l}p(n-1)}} \ , \tag{9}$$

which for large n becomes:

$$q_{fl} \approx \frac{\Lambda_{fl} + \frac{r_{fl}\ q_{f,l}(1-p)}{1 - q_{f,l}p}}{r_{fl} + \lambda^-_{fl} + \frac{r_{fl}\ q_{f,l}p}{1 - q_{f,l}p}} \ . \tag{10}$$

Since q_{fl} is a probability, by setting $q_{fl} \leq 1$ one obtains,

$$\Lambda_{fl} \leq \lambda^-_{fl} + r_{fl} \ . \frac{p}{1-p} \ , \tag{11}$$

and we see can initialize the AADRNN parameters as follows:

$$\Lambda_{fl} = r_{fl} + \frac{p}{1-p} \ , \tag{12}$$

and set $r_{fl} = 1$ throughout the network, for large n and small p. The we can define the vectors and matrices:

$$Q_{l-1} \ = \ [q_{1,l-1},\ \ldots\ , q_{\mathcal{F}_{l-1},l-1} \ , \tag{13}$$

$$W_{l-1} \ = \ [w^-_{g,f,l-1}]_{[\mathcal{F}_l \times \mathcal{F}_l} \ , \tag{14}$$

We can write the expression (10) as:

$$\begin{aligned} &q_{fl} \approx \zeta(Q_{l-1}, W_{l-1}),\ \textit{or in vector form}: \\ &Q_l \approx \zeta(Q_{l-1}, W_{l-1}),\ Q_0 = [x^1,\ \ldots\ , x^F] = x \ , \\ &\hat{x} = [\hat{x}^1,\ \ldots\ , \hat{x}^F] = Q_L \ . \end{aligned}$$

The matrices W_l are initially drawn at random, and then progressively optimized step by step during auto-associative training to minimize the error function $||x - \hat{x}||$.

III. Extraction of Input Features from Data

In order to prepare the traffic features for the input of AADRNN, we preprocess (via simple/lightweight operations) the features extracted from the network traffic and available in the data set. First, the non-numerical (categorical) features are encoded into numerical features. To this end, for each non-numerical feature f, the possible set of unique values, denoted by $\mathcal{U}_f$, is determined. Then, for each unique value $u \in \mathcal{U}_f$, a positive integer in $[1, |\mathcal{U}_f|]$ is assigned. As soon as each non-numerical feature is converted to numerical, the value of the numerical feature, denoted by x^f_k, is normalized by min-max scaling as

$$x^f_k \leftarrow \frac{x^f_k - \min\limits_{k \in \mathcal{K}} x^f_k}{\max\limits_{k \in \mathcal{K}} x^f_k - \min\limits_{k \in \mathcal{K}} x^f_k} \tag{15}$$

where $\mathcal{K}$ is the set of all traffic (samples) available in the data set.

A. Statistical Whisker based Non-Attack Classification

In order to classify the traffic features as attack or benign, we train the ADRNN using only benign traffic. This classifier makes a decision by evaluating the traffic features compared against the output of AADRNN based on the statistical whisker.

During the decision making for a packet (or any sample) k, first one calculates the absolute difference between the actual features and the predicted features by the AADRNN:

$$z^f_k = |x^f_k - \hat{x}^f_k| \qquad \forall f \in \mathcal{F}. \tag{16}$$

Then, the total number of features with abnormal values, denoted by ζ_k, is computed:

$$\zeta_k = \sum_{f \in \mathcal{F}} \mathbf{1}(z^f_k > w_f), \tag{17}$$

where w_f is the value of whisker that is learned during the training, where $\mathbf{1}(\Xi) = 1$ if the statement Ξ is true, and $\mathbf{1}(\Xi) = 0$ otherwise. That is, in (17), the feature f is considered to have an abnormal value indicative of an attack when $z_k^f > w_f$.

Finally, the packet k is considered an attack if there are more than θ features with abnormal values:

$$y_k = \mathbf{1}(\zeta_k > \theta). \tag{18}$$

Using the training data, $\mathcal{K}_{\text{train}}$, which consists of only benign traffic features, we determine the values of w_f and θ, respectively. To this end, first, the value of the absolute difference z_k^f is computed for each $k \in \mathcal{K}_{\text{train}}$ using (16). For each feature f, the lower quartile Q_l^f and upper quartile Q_u^f of $\{z_k^f\}_{k\in\mathcal{K}_{\text{train}}}$ are calculated as

$$\mathcal{K}^f_{\text{lower}} \equiv \{k : z_k^f < \text{median}(\{z_k^f\}_{k\in\mathcal{K}_{\text{train}}})\}, \quad \forall f \in \mathcal{F} \tag{19}$$

$$Q_l^f = \text{median}(\{z_k^f\}_{k\in\mathcal{K}^f_{\text{lower}}}), \quad \forall f \in \mathcal{F}, \tag{20}$$

and

$$\mathcal{K}^f_{\text{upper}} \equiv \{k : z_k^f > median(\{z_k^f\}_{k\in\mathcal{K}_{\text{train}}})\}, \quad \forall f \in \mathcal{F} \tag{21}$$

$$Q_u^f = \text{median}(\{z_k^f\}_{k\in\mathcal{K}^f_{\text{upper}}}), \quad \forall f \in \mathcal{F} \tag{22}$$

Using Q_l^f and Q_u^f, the whisker w_f for the upper quartile is calculated as

$$w_f = Q_u^f + 1.5(Q_u^f - Q_l^f) \qquad \forall f \in \mathcal{F} \tag{23}$$

Since the training data is based on benign traffic, θ must be selected to classify training samples as benign traffic. However, we should also consider that the training data may include some outlier samples. Thus, we set the value of θ to the mean of the number of abnormal features plus the standard deviation of abnormal features in the training data:

$$\theta = \mu_\zeta + 2\sigma_\zeta, \tag{24}$$

where

$$\mu_\zeta = \frac{\sum_{k\in\mathcal{K}_{\text{train}}} \zeta_k}{|\mathcal{K}_{\text{train}}|}, \quad \sigma_\zeta = \sqrt{\frac{\sum_{k\in\mathcal{K}_{\text{train}}} (\zeta_k - \mu_\zeta)^2}{|\mathcal{K}_{\text{train}}|}} \tag{25}$$

IV. Evaluating the AADRNN

The KDD Cup'99 data set [54] includes both "normal" or benign traffic as well as attack (intrusion) traffic. It contains three different subsets: the training set, the smaller training set reduced to 10% of the total training set, and the test set, comprised of "$4,898,431$", "$494,021$", and "$311,029$" samples respectively, with 41 features related to network traffic.

We first evaluate the ADRNN with three layers and $\mathcal{F}_\infty = 41,\ \mathcal{F}_\infty = 20,\ \mathcal{F}_\infty = 41$, where the number of clusters in the second layer is simply chosen to be roughly one half of the number of 41 input and output clusters.

To be comparable with recent state-of-the-art, we use the small training set. Thus the AADRNN is:

- Trained with only the $97,278$ benign samples in the smaller training set, and
- Evaluated using all samples in the test set.
- All experiments are performed on a workstation with 32 Gb RAM and an AMD 3.7 GHz (Ryzen 7 3700X) processor.

The structure of the AADRNN includes three layers with 41, 20 and 41 neurons respectively, where 41 corresponds to the number of features. In addition, we set $r_{fl} = 1$ for all clusters, $\lambda_{fl} = \lambda_{fl} = 0.005$, and $p = 0.01$.

We compare the performance of the AADRNN against the unsupervised state-of-the-art one class classification technique based on the Support Vector Machine - One Class Classifier (SVM-OCC). In addition, we compare the performance of AADRNN with that of several supervised machine learning techniques using the same dataset, namely the Linear Regression (LR), K-Nearest Neighbours (KNN), Decision Tree (DT) and Random Forest (RF), which are detailed in [56].

A. Results for the $41-20-41$ Cluster Three Layer AADRNN

In Figure 3, the performance is presented with respect to Accuracy, True Negative Rate (TNR), True Positive Rate (TPR), Precision, and F1 Score on the KDD dataset as a whole, showing that AADRNN achieves aproximately 93% accuracy with high TNR (97%) and TPR (92%). According to these results, the AADRNN of Figure successfully classifies both benign and attack traffic as a whole for all the test data and attack types in the KDD dataset, although it has been trained only with a small benign traffic dataset.

As indicated earlier, the KDD Cup'99 dataset contains a wide variety of attack types, so that the performance of the ADRNN can differ for different attack types. To this end, Figure 4 displays the performance of AADRNN for each individual type of attack in the test set. The results show that the prediction accuracy is less than or equal to 50% for 10 out of 37 attack types while it is above 98% for 21 out of 37 attack types.

1) Comparison with Unsupervised and Supervised Techniques: We now compare the performance of the $41-20-41$ cluster three layer AADRNN first with the performance of the unsupervised technique based on the Support Vector Machine SVM-OCC, and then with other supervised state-of-the-art machine learning techniques. Figure 5 compares AADRNN with SVM-OCC with respect to Accuracy, TNR and TPR. The results in this figure show that of the $41-20-41$ cluster three layer AADRNN clearly outperforms this state-of-the-art unsupervised one class classifier for detecting both benign and attack traffic, and the performance difference is significant especially for detecting attack traffic.

In addition Table I, presents the computation times of AADRNN and SVM-OCC. One may see that AADRNN is two orders of magnitude faster than SVM-OCC. Moreover, since the training of AADRNN requires only benign traffic and takes $1.49\ s$ on average, AADRNN can also be trained online for some applications.

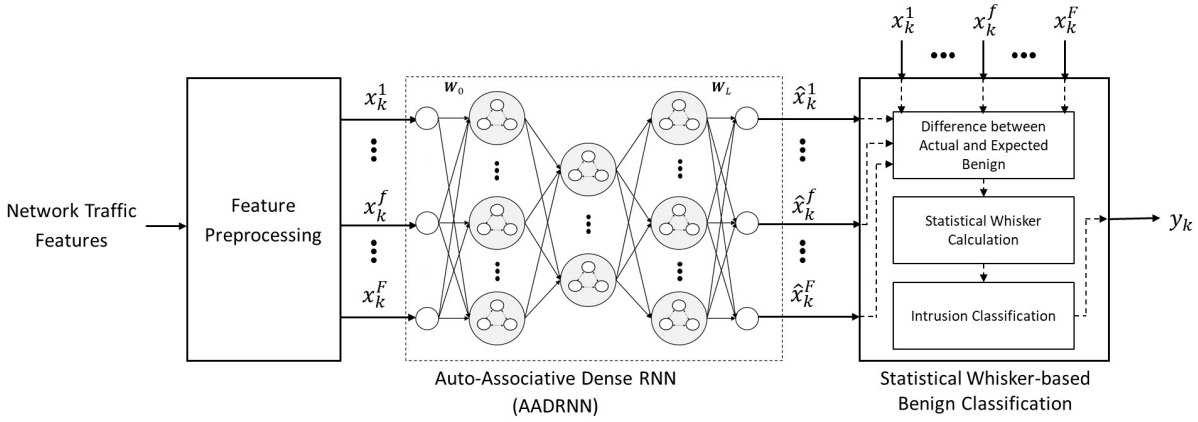

Fig. 2: Architecture of the $41-20-41$ three layer AADRNN for Attack Detection with the KDD [54] Dataset.

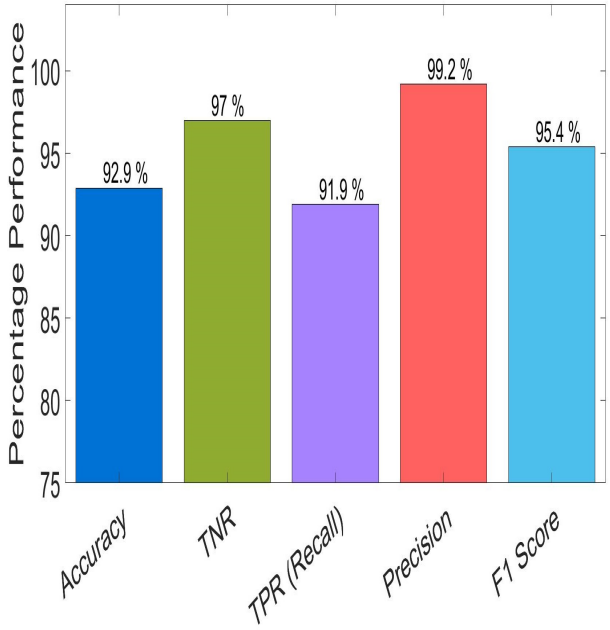

Fig. 3: Performance of the $41-20-41$ cluster three layer AADRNN with respect to Accuracy, TNR, TPR, Precision and F1 Score metrics for the KDD dataset taken as a whole for all attack types.

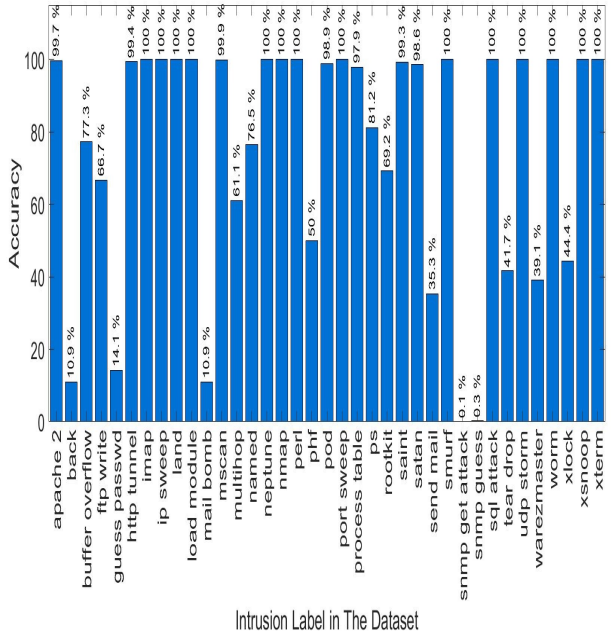

Fig. 4: The $41-20-41$ cluster three leayer ADRNN's performance for each attack type in the KDD dataset.

B. *Evaluation of the* $41-41-41$ *AADRNN*

Finally we also briefly evaluate the three layer and $41-41-41$ cluster AADRNN with the same small training dataset and the whole KDD testing dataset. In Figure 6 we observe its average performance over all of the attack types in the dataset, and observer a slightly lower performance than that of the $41-20-41$ cluster network that is given in Figure 3.

In Figure 7 we summarize the performance of the three layer $41-41-41$ cluster AADRNN for each of the individual attack types, and again we see a somewhat lower performance than that of the $41-20-41$ network that is shown in Figure 4. Hence we see that the mapping of the data into a sub-space, as offered by the $41-20-41$ network, provides better performance.

V. Conclusions

This paper shows that a special case of G-networks [32], [55], the AADRNN, can be used as a detector for a broad range of packet based network attacks. To this effect, this work uses the function approximation capacity of G-Networks [37].

Another novelty of this work is that the AADRNN is *not* trained with the different attack data sets that it is expeted to detect. Rather, the AADRNN is **only trained with benign non-attack traffic** as an autassociative neural network, and

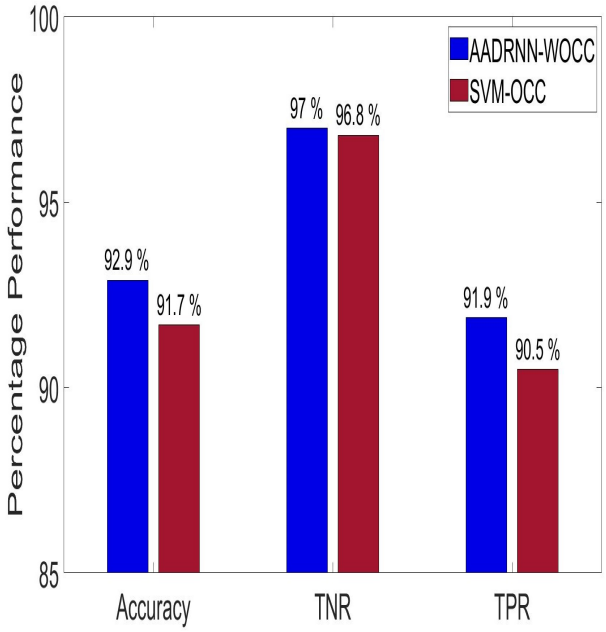

Fig. 5: Comparison of the $41-20-41$ cluster three layer AADRNN against the state-of-the-art one class classifier SVM-OCC on the KDD dataset.

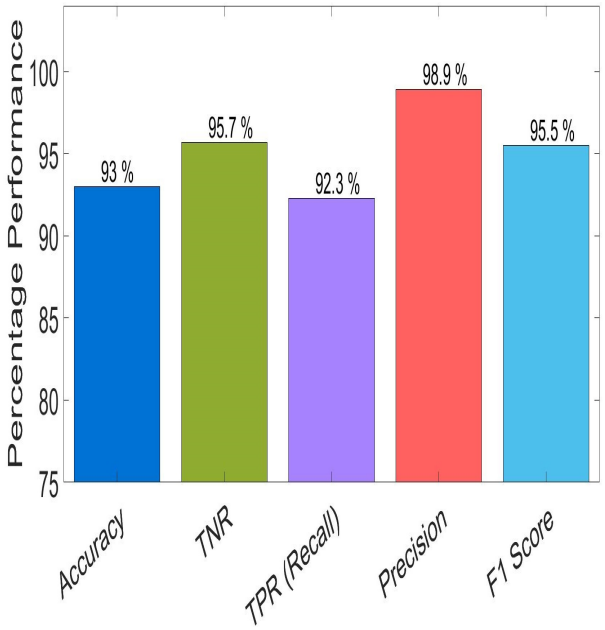

Fig. 6: The $41-41-41$ cluster three layer ADRNN's average performance for all attack types in the KDD dataset.

TABLE I: Comparison of the $41-20-41$ three layer AADRNN against a state-of-the-art one class classifier, the SVM-OCC with respect to computation time.

Model	**Total Training Time (seconds)**		**Execution Time per Sample (secs)**	
	Mean	**St-Dev**	**Mean**	**St-Dev**
AADRNN	1.49	0.03	5.13	0.08
SVM-OCC	249.7	9.44	326.3	7.59

despite this limited training it is able to detect many different attack types.

The AADRNN is also limited in this case to two different three-layer networks. Learning and training is conducted with the well known DARPA KDD [54] dataset. The first AADRNN is a $41-20-41$ cluster structure, and the second one is a $41-41-41$ cluster structure.

The $41-20-41$ cluster structure is observed to achieve better performance, which is summarized in Table II, and compared to other well-known machine learning techniques: the unsupervised SVM-OCC, and the supervised MLP, LR, KNN, DT, RF systems whose performance was recently reported in [56].

When evaluated with the KDD dataset, the $41-20-41$ cluster AADRNN outperforms these other techniques except

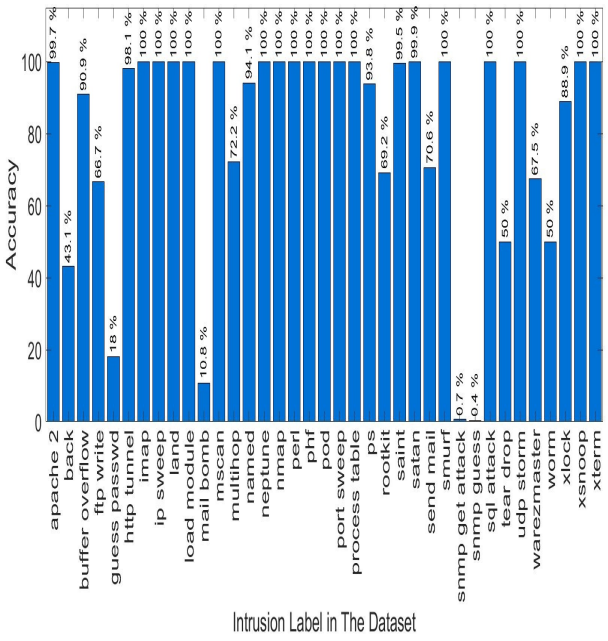

Fig. 7: The $41-41-41$ cluster three layer ADRNN's performance for each attack type in the KDD dataset.

for MLP and DT. A comparison with MLP and DT shows that the AADRNN achieves almost the same performance, but its Recall metric is higher and its Precision is lower, while it yields slightly lower False Negatives and slightly higher False Positives.

In future work, we plan to extend these comparisons to other available real or synthetic datasets. In particular we plan to use the NSL-KDD dataset [57], and some validated synthetic datasets which may also available, while other forms of G-Networks may also be tested.

REFERENCES

[1] Cisco, *Cisco Annual Internet Report (2018–2023)*, Mar. 2020. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

[2] "Hp study reveals 70 percent of Internet of Things devices vulnerable to attack," Accessed on 25.01.2022. [Online]. Available: https://www.hp.com/us-en/hp-news/press-release.html%3Fid=1744676

[3] M. Frustaci, P. Pace, G. Aloi, and G. Fortino, "Evaluating critical security issues of the IoT world: Present and future challenges," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2483–2495, 2018.

[4] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1125–1142, 2017.

[5] Y. Yang, L. Wu, G. Yin, L. Li, and H. Zhao, "A survey on security and privacy issues in internet-of-things," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1250–1258, 2017.

[6] S. Benzarti, B. Triki, and O. Korbaa, "A survey on attacks in Internet of Things based networks," in *2017 International conference on engineering & MIS (ICEMIS)*. IEEE, 2017, pp. 1–7.

[7] CISA, "Understanding Denial-of-Service attacks." [Online]. Available: https://us-cert.cisa.gov/ncas/tips/ST04-015

[8] G. Carl, G. Kesidis, R. Brooks, and S. Rai, "Denial-of-Service attack-detection techniques," *IEEE Internet Computing*, vol. 10, no. 1, pp. 82–89, 2006.

[9] G. Matta, S. Chlup, A. M. Shaaban, C. Schmittner, A. Pinzenöhler, E. Szalai, and M. Tauber, "Risk management and standard compliance for cyber-physical systems of systems," *Infocommunications Journal*, vol. 13, no. 2, pp. 32–39, June 2021. [Online]. Available: https://doi.org/10.36244/ICJ.2021.2.5

[10] S. Maksuti, M. Zsilak, M. Tauber, and J. Delsing, "Security and autonomic management in system of systems," *Infocommunications Journal*, vol. 13, number = 3, month = September, year = 2021, pages = 66-75, url = https://doi.org/10.36244/ICJ.2021.3.7.

[11] C. Douligeris and A. Mitrokotsa, "DDoS attacks and defense mechanisms: classification and state-of-the-art," *Computer networks*, vol. 44, no. 5, pp. 643–666, 2004.

[12] J. Margolis, T. T. Oh, S. Jadhav, Y. H. Kim, and J. N. Kim, "An in-depth analysis of the mirai botnet," in *2017 International Conference on Software Security and Assurance (ICSSA)*. IEEE, 2017, pp. 6–12.

[13] J. Biggs, "Hackers release source code for a powerful DDoS app called Mirai," *TechCrunch*, October 2018. [Online]. Available: https://techcrunch.com/2016/10/10/hackers-release-source-code-for-a-powerful-ddos-app-called-mirai/

[14] R. Hackett, "Why a hacker dumped code behind colossal website-trampling botnet," October 2016.

[15] N. Statt, "How an army of vulnerable gadgets took down the web today," October 2016. [Online]. Available: https://www.theverge.com/2016/10/21/13362354/dyn-dns-ddos-attack-cause-outage-status-explained

[16] B. Tushir, H. Sehgal, R. Nair, B. Dezfouli, and Y. Liu, "The impact of dos attacks onresource-constrained iot devices: A study on the mirai attack," *arXiv preprint arXiv:2104.09041*, 2021.

[17] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou, "Understanding the Mirai Botnet," in *Proceedings of the 26th USENIX Security Symposium*, 2017. [Online]. Available: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis

[18] H. Sinanović and S. Mrdovic, "Analysis of mirai malicious software," in *2017 25th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*. IEEE, 2017, pp. 1–5.

[19] R. Doshi, N. Apthorpe, and N. Feamster, "Machine learning ddos detection for consumer internet of things devices," in *2018 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2018, pp. 29–35.

[20] C. S. Htwe, Y. M. Thant, and M. M. S. Thwin, "Botnets attack detection using machine learning approach for iot environment," in *Journal of Physics: Conference Series*, vol. 1646, no. 1. IOP Publishing, 2020, p. 012101.

[21] M. Banerjee and S. Samantaray, "Network traffic analysis based iot botnet detection using honeynet data applying classification techniques," *International Journal of Computer Science and Information Security (IJCSIS)*, vol. 17, no. 8, 2019.

[22] S. Sriram, R. Vinayakumar, M. Alazab, and K. Soman, "Network flow based iot botnet attack detection using deep learning," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2020, pp. 189–194.

[23] C. D. McDermott, F. Majdani, and A. V. Petrovski, "Botnet detection in the Internet of Things using deep learning approaches," in *2018 international joint conference on neural networks (IJCNN)*. IEEE, 2018, pp. 1–8.

[24] G. D. L. T. Parra, P. Rad, K.-K. R. Choo, and N. Beebe, "Detecting Internet of Things attacks using distributed deep learning," *Journal of Network and Computer Applications*, vol. 163, p. 102662, 2020.

[25] T. A. Tuan, H. V. Long, R. Kumar, I. Priyadarshini, N. T. K. Son *et al.*, "Performance evaluation of botnet ddos attack detection using machine learning," *Evolutionary Intelligence*, pp. 1–12, 2019.

TABLE II: Performance of the $41 - 20 - 41$ three layer AADRNN compared to several state-of-the-art unsupervised and supervised techniques with the KDD dataset.

Model		**Metrics (%)**			
		Accuracy	**Recall**	**Precision**	**F1 Score**
Unsupervised	**AADRNN-WOCC**	92.9	91.9	99.2	95.4
	SVM-OCC	91.7	90.5	99.1	94.6
Supervised [56]	**MLP with 4 layers**	93	91.5	99.8	95.5
	LR	81.1	76.9	99.4	86.7
	KNN	92.5	90.9	99.8	95.2
	DT	92.9	91.5	99.7	95.4
	RF	92.7	91.1	99.9	95.3

[26] A. Kumar and T. J. Lim, "Early detection of mirai-like iot bots in large-scale networks through sub-sampled packet traffic analysis," in *Future of Information and Communication Conference*. Springer, 2019, pp. 847–867.

[27] Y. N. Soe, Y. Feng, P. I. Santosa, R. Hartanto, and K. Sakurai, "Machine learning-based iot-botnet attack detection with sequential architecture," *Sensors*, vol. 20, no. 16, p. 4372, 2020.

[28] M. Chatterjee, A. S. Namin, and P. Datta, "Evidence fusion for malicious bot detection in iot," in *2018 IEEE International Conference on Big Data (Big Data)*, 2018, pp. 4545–4548.

[29] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi, "DÏot: A federated self-learning anomaly detection system for iot," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, 2019, pp. 756–767.

[30] E. Gelenbe and E. C.-H. Ngai, "Adaptive qos routing for significant events in wireless sensor networks," in *2008 5th IEEE International Conference on Mobile Ad Hoc and Sensor Systems*. IEEE, 2008, pp. 410–415.

[31] F. Francois and E. Gelenbe, "Optimizing secure sdn-enabled inter-data centre overlay networks through cognitive routing," in *2016 IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*. IEEE, 2016, pp. 283–288.

[32] E. Gelenbe, "G-networks with triggered customer movement," *Journal of Applied Probability*, vol. 30, no. 3, pp. 742–748, September 1993. [Online]. Available: https://doi.org/10.2307/3214781

[33] E. Gelenbe and A. Stafylopatis, "Global behavior of homogeneous random neural systems," *Applied mathematical modelling*, vol. 15, no. 10, pp. 534–541, 1991.

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

[35] ——, "Learning in the recurrent random neural network," *Neural Computation*, vol. 5, no. 1, pp. 154–164, 1993.

[36] S. Evmorfos, G. Vlachodimitropoulos, N. Bakalos, and E. Gelenbe, "Neural network architectures for the detection of syn flood attacks in IoT systems," in *Proceedings of the 13th ACM International Conference on PErvasive Technologies Related to Assistive Environments*, 2020, pp. 1–4.

[37] E. Gelenbe, Z.-H. Mao, and Y.-D. Li, "Function approximation with spiked random networks," *IEEE Transactions on Neural Networks*, vol. 10, no. 1, pp. 3–9, 1999.

[38] ——, "Function approximation by random neural networks with a bounded number of layers," in *Computer System Performance Modeling In Perspective: A Tribute to the Work of Prof. Kenneth C. Sevcik*. World Scientific, 2006, pp. 35–58.

[39] G. Öke, G. Loukas, and E. Gelenbe, "Detecting denial of service attacks with bayesian classifiers and the random neural network," in *2007 IEEE International Fuzzy Systems Conference*. IEEE, 2007, pp. 1–6.

[40] A. Qureshi, H. Larijani, J. Ahmad, and N. Mtetwa, "A novel random neural network based approach for intrusion detection systems," in *CEEC*. IEEE, 2018, pp. 50–55.

[41] C. E. Cramer and E. Gelenbe, "Video quality and traffic qos in learning-based subsampled and receiver-interpolated video sequences," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 2, pp. 150–167, 2000.

[42] H. Cancela, F. Robledo, and G. Rubino, "A GRASP algorithm with RNN based local search for designing a WAN access network," *Electron. Notes Discret. Math.*, vol. 18, pp. 59–65, 2004.

[43] M. Martínez, A. Morón, F. Robledo, P. Rodríguez-Bocca, H. Cancela, and G. Rubino, "A GRASP algorithm using RNN for solving dynamics in a P2P live video streaming network," in *HIS*. IEEE Computer Society, 2008, pp. 447–452.

[44] S. Mohamed and G. Rubino, "A study of real-time packet video quality using random neural networks," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 12, pp. 1071–1083, December 2002.

[45] A. Javed, H. Larijani, A. Ahmadinia, and D. Gibson, "Smart random neural network controller for hvac using cloud computing technology," *IEEE Transactions on Industrial Informatics*, vol. 13, pp. 351–360, 2017.

[46] O. Brun, L. Wang, and E. Gelenbe, "Big data for autonomic intercontinental overlays," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 3, pp. 575–583, 2016.

[47] E. Gelenbe and Y. Yin, "Deep learning with dense random neural networks," in *ICMMI*, ser. Advances in Intelligent Systems and Computing, vol. 659. Springer, 2017, pp. 3–18.

[48] W. Ingabire, H. Larijani, R. M. Gibson, and A. Qureshi, "Outdoor node localization using random neural networks for large-scale urban iot lora networks," *Algorithms*, vol. 14, no. 11, p. 307, 2021.

[49] S. Y. Shah, H. Larijani, R. M. Gibson, and D. Liarokapis, "Random neural network based epileptic seizure episode detection exploiting electroencephalogram signals," *Sensors*, vol. 22, no. 7, p. 2466, 2022.

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

[51] Y. Yin and E. Gelenbe, "Nonnegative autoencoder with simplified random neural network," *CoRR*, vol. abs/1609.08151, 2016.

[52] 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.

[53] M. Nakip and E. Gelenbe, "MIRAI botnet attack detection with auto-associative dense random neural network," in *IEEE Global Communications Conference (Globecom)*, 2021, pp. 1–6.

[54] "KDD Cup 1999 Data." [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[55] E. Gelenbe, "G-networks: a unifying model for neural and queueing networks," *Annals of Operations Research*, vol. 48, pp. 433–461, 1994.

[56] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41 525–41 550, 2019.

[57] A. Ghorbani, et al., "The nsl-kdd dataset," *Canadian Institute for Cybersecurity*. [Online]. Available: https://www.unb.ca/cic/datasets/nsl.html