Memristor Based Autoencoder for Unsupervised Real-Time Network Intrusion and Anomaly Detection

Md. Shahanur Alam, B. Rasitha Fernando, Yassine Jaoudi, Chris Yakopcic, Raqibul Hasan, Tarek M. Taha, and Guru Subramanyam
Dept. Of Electrical and Computer Engineering, *University of Dayton,* Dayton, OH, USA
{alamm8, fernandob1, jaoudiy1, cyakopcic1, hasanm1, tarek.taha, gsubramanyam1}@udayton.edu

ABSTRACT

Custom low power hardware for real-time network security and anomaly detection are in great demand, as these would allow for efficient security in battery-powered network devices. This paper presents a memristor based system for real-time intrusion detection, as well as an anomaly detection based on autoencoders. Intrusion detection is based on a single autoencoder, and the overall detection accuracy of this system is 92.91% with a malicious packet detection accuracy of 98.89%. The system described in this paper is also capable of using two autoencoders to perform anomaly detection using real-time online learning. Using this system, we show that anomalous data is flagged by the system, but over time the system stops flagging a particular datatype if its presence is abundant. Utilizing memristors in these designs allows us to present extreme low power systems for intrusion and anomaly detection, while sacrificing little accuracy.

CCS CONCEPTS

CCS → Hardware → Emerging technologies → Analysis and design of emerging devices and systems → Emerging architectures

KEYWORDS

Memristor, Autoencoder, NSL-KDD, Neuromorphic

1 Introduction

The incredible ubiquitousness of the internet has led to more diverse protocols, services, and applications [1]. This has also increased the chance of breaches in system security, allowing theft of confidential information. Information is among the most valuable assets for any institution [2]. Thus, any network intrusion which places data confidentiality, integrity, or availability at risk can cause dramatic losses for the organization. According to [3], a continuous surveillance of networks looking in real time for anomalous behavior is necessary to prevent intrusion.

Two of the prominent methods for performing network intrusion detection include rule/signature-based detection, as well as anomaly-based detection. Rule-based automatic intrusion detection systems such as SNORT are widely used. In a rule-based system, an incoming packet is compared with a list of known malicious signatures to determine if a particular data packet contains an attack [4]. However, this does little against attacks that SNORT is not already aware of. Thus, SNORT is unable to detect new types of attacks, typically called 'zero day' attacks [4]. Deep learning systems are promising candidates for the improvement of anomaly-based intrusion detection systems due to their pattern recognition and pattern reconstruction capabilities [5,6]. Deep networks that are adaptable and can perform self-organization to learn new features [4] are needed to detect 'zero day' attacks in real-time.

There are two types of learning in neural networks: supervised and unsupervised. Unsupervised learning does not require labeled data, and thus is a strong candidate for anomaly detection, as real-time network traffic will also not be labeled [7]. A key problem with this approach is that the learning in deep networks requires significant memory, computation time, and power consumption [8]. Thus, these networks are typically implemented using Graphics Processing Units (GPUs) that consume more than 200W of power. Thus, these systems are not practical for edge computing.

In the IoT era, many physical devices are connected to the internet that perform very sophisticated tasks like automation, industrial processes, human health analysis, and environmental monitoring [9]. The integration of real-world objects with the internet brings network security threats into the realm of our daily activities. Real-time anomaly detection has a high demand in the network security industry. Besides intrusion detection, other important applications of these types of systems include malware detection [8].

To enable this anomaly detection on low power devices, building unsupervised deep learning circuits using memristors is a

This work was supported through funding from the National Science Foundation.

ICONS '19, July 23–25, 2019, Knoxville, TN, USA
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-7680-8/19/07…$15.00
https://doi.org/10.1145/3354265.3354267

viable solution [10]. Memristors are a new class of nanoscale semiconductor device. They are non-volatile and can retain their resistance state when powered off. New types of neuromorphic systems have been proposed recently that utilize memristor crossbars to implement deep networks very efficiently [10]. They have high density and extreme low-power, as they perform many multiply-add operations in parallel in the analog domain [10,11].

In this work, we propose a network intrusion and real-time anomaly detection system that utilizes memristor crossbars to construct autoencoders. Autoencoders are able to learn in an unsupervised fashion and can recognize anomalous data quickly and accurately. Furthermore, since we are using memristor devices to carry out the critical computations required of autoencoders, our proposed system will operate at significantly lower power relative to alternative traditional approaches [10]. The system proposed in this work is capable of performing two different processes. The first is a feedforward intrusion detection process. In this case, the system is first trained using a set of normal, benign, network data to determine a baseline. Then the system is tested in a runtime mode as it is supplied a mix of normal and malicious data. This system was able recognize the difference between normal and attack data at greater than 92% accuracy.

The second process capable of execution via the proposed system employs a feedback path to provide self-learning during real-time operation. This system was successfully used to detect anomalous data. Furthermore, if data that was once considered anomalous becomes more prevalent over time, this system is able to recognize this pattern and forget this anomaly. However, as new anomalies are presented to the system, they are still recognized.

This paper is organized as follows: Section II discusses related work, and Section III provides details on the dataset used in this study. Section IV describes the memristor based auto encoder design, and Section V details the experimental method carried out in this work. Section VI provides the results and discussion while Section VII contains a brief conclusion.

2 Related Work

Few papers have been presented that describe using memristor crossbars as part of a neuromorphic intrusion detection system. Work in [12] presents memristor based intrusion detection hardware implemented as a supervised multilayer perceptron (MLP), which achieved a classification accuracy greater than 99% when using the KDD Cup'99 dataset. Work in [13] presents a memristor based deep packet inspection (DPI) system for high speed intrusion detection and classification.

On the other hand, more studies have been conducted that describe intrusion detection based on deep learning that is not tied to a specific hardware design. Network intrusion detection is studied in [7] using unsupervised learning via autoencoders and a restricted Boltzmann machine (RBM) that achieved up to 92.12% detection accuracy with k-means clustering. Work has also been presented using the NSL-KDD dataset for intrusion detection in unsupervised deep learning [14]. An autoencoder with five hidden layers achieved 96.3% classification accuracy with a support vector machine (SVM) on the KDDTest+ data set [14]. Researchers are also exploring convolutional neural networks (CNNs) for intrusion detection, where a preprocessing unit is used to increase the dimensionality of the input network data [15]. A deep autoencoder trained in a greedy layer-wise fashion achieved 94.53% accuracy [16].

As an alternative, in this work we describe an autoencoder as part of a memristor based neuromorphic system with real-time detection. Our results also show that we are able to train our system during runtime to only respond to anomalous data. To the best of our knowledge, no other memristor based system capable of both unsupervised intrusion detection and anomaly detection has yet been published.

3 The NSL-KDD Dataset

NSL-KDD dataset is a revised version of the KDD Cup'99 dataset currently hosted by the University of New Brunswick [17]. In the NSL-KDD dataset, the significant redundancy originally present in the KDD Cup'99 dataset has been removed [17]. The portion of the data in the NSL-KDD dataset dedicated to training has a total of 125,973 packets, each containing one of 23 different datatypes: (1) back, (2) buffer_overflow, (3) ftp_write, (4) guess_passwd, (5) imap, (6) ipsweep, (7) land, (8) loadmodule, (9) multihop, (10) neptune, (11) nmap, (12) normal, (13) perl, (14) phf, (15) pod, (16) portsweep, (17) rootkit, (18) satan, (19) smurf, (20) spy, (21) teardrop, (22) warezclient, and (23) warezmaster [18]. Each packet has 43 attributes, where 42 of the attributes match those found in the KDD Cup'99 data (including the data label), and the 43rd describes the level of classification difficulty of the corresponding packet [18]. The NSL-KDD data used in this study contains 67,343 packets that are considered normal benign data, while the remaining 58,630 packets in the dataset are considered malicious. Fig. 1 presents two sample data packets. Fig. 1 (a) presents a normal data sample and Fig. 1 (b) presents a malicious data packet. Both packets contain numerical data, as well as alphanumeric data within certain attributes, including the 2nd position (protocol/type), the 3rd position (service), the 4th position (flag), and the 42nd position (the attack type).

```
0,tcp,ftp_data,SF,491,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0,0,0,0,1,0,0,
150,25,0.17,0.03,0.17,0,0,0,0.05,0,normal,20
```

(a)

```
0,tcp,ftp_data,SF,334,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,2,2,0,0,0,0,1,0,0,
2,20,1,0,1,0.20,0,0,0,0,warezclient,15
```

(b)

```
0,0.5,0.188,0.629,3.55e07,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.00391,0.003
91, 0,0,0,0,1,0,0,0.588,0.098,0.17,0.03,0.17,0,0,0,0.05,0,0,0.9523
```

(c)

```
0,0.5,0.188,0.629,2.42e07,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0.00391,0.003
9, 0,0,0,0,1,0,0,0.0078,0.078,1,0,1,0.2,0,0,0,0,1,0.714
```

(d)

Figure 1. Examples of a (a) normal data packet, (b) malicious data packet, (c) normalized normal data packet, and (d) normalized malicious data packet.

Before training the network, all packets need to undergo minor preprocessing. The alphanumeric data was converted into numeric data and all data was normalized. The 42nd position in the data set stores the labels of the packets. Normal packet labels are denoted as '0' while all types malicious data have a label of '1'. The three remaining features in the 2nd to 4th positions are alphanumeric and need to be adjusted (see Fig. 1 (a) & (b)). For example, the 2nd position has three possible strings (tcp, udp, icmp), and these can be replaced by 1, 2 and 3 respectively. Then the dataset is normalized according to the largest value contained within each attribute. The processed versions of the packets displayed in Figs. 1 (a) and (b) are shown in Figs. 1 (c) and (d) respectively. The last processing step was to remove the 20th attribute in the dataset, as a zero was present in this position for all packets.

4 Memristor Based Autoencoder

4.1 Autoencoder Design

An autoencoder (AE) is a type of unsupervised neural network. The main purpose of an autoencoder is to perform feature learning in a way that reduces the dimensionality of incoming data. Once the minimum dimensionality is achieved, the autoencoder should then be able to reconstruct the original data with little to no error [7]. The autoencoder layout used in this work is displayed in Fig. 2, where the encoder and decoder layers sit on either side of a bottleneck layer, which produces features in their most compressed form. The encoder ($\emptyset$) and decoder (ϕ) processes are shown in equations (1) and (2).

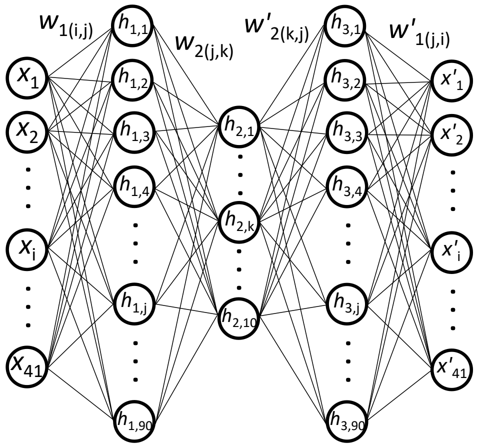

Figure 2. Conceptual diagram of an autoencoder.

The output of the autoencoder described in Fig. 2 can be obtained by carrying out equations (3-6) where b denotes a bias value, and $f(x)$ denotes neuron activation function.

$$\emptyset : X \rightarrow \mathcal{F} \tag{1}$$
$$\phi : \mathcal{F} \rightarrow X \tag{2}$$
$$L_{1j} = f\left(\sum_{i=1}^{41} w_{1(i,j)} . x_i + b_{1j}\right) \tag{3}$$
$$L_{2k} = f\left(\sum_{j=1}^{90} w_{2(j,k)} . h_{1j} + b_{2k}\right) \tag{4}$$
$$L_{3j} = f\left(\sum_{k=1}^{10} w'_{2(k,j)} . h_{2k} + b_{3j}\right) \tag{5}$$
$$L_{4i} = f\left(\sum_{j=1}^{90} w'_{1(j,i)} . h_{3j} + b_{4i}\right) \tag{6}$$

4.2 Memristor Autoencoder

To build on previous work in autoencoder based intrusion detection [5,6], we propose a memristor crossbar-based system for performing unsupervised network intrusion detection. Memristor devices [19] are a strong candidate for the basis of a low power embedded neuromorphic system. Memristors are commonly patterned in what is known as a crossbar structure, which is capable of performing many multiply-add operation in a parallel fashion in the analog domain [20].

Memristors are typically utilized in neuromorphic systems to approximate the concept of synaptic connectivity. Thus, memristors can be used to store the connection strength between a neuron and all incoming connections. This is demonstrated in Fig. 3. This circuit requires two memristors to represent a single weight because the dynamic resistance of a memristor can really only be used to store a single positive bounded value. The left column of memristors represents a positive excitatory connection and the right column represents an inhibitory connection. In a given row, if $\sigma_{i+} > \sigma_{i-}$ then a net positive synaptic weight is observed, otherwise a negative synaptic weight will be present [21].

Assume that the value of the dot product (DP_j) can be calculated according to equation (7) as the voltage difference between the left and right column wires. Thus, each memristor crossbar in this system essentially performs a set of dot product calculations between the neuron input voltages and the net conductance of each memristor pair. The memristor device considered in this work has maximum conductance $\sigma_{max} = 2 \times 10^{-5}\ \Omega^{-1}$ and minimum conductance $\sigma_{min} = 1 \times 10^{-7}\ \Omega^{-1}$.

The output, y_j in Fig. 3 represents the neuron output. When the power rails of the op-amps, V_{DD} and V_{SS} are set to 0 V and 1 V respectively. The sigmoid presented in equation (8) is typically used as the activation function in deep learning. However, in the presented memristor based neuromorphic system the approximated sigmoid function in equation (9) is used, as it is easier to generate using an amplifier circuit [12]. Fig. 4 displays the typical sigmoid function along with the approximation used in this work.

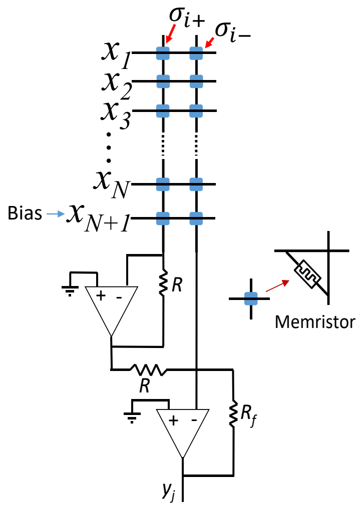

Figure 3. Circuit diagram for a single memristor based neuron.

$$DP_j = \sum_{i=1}^{N+1} x_i \times \left(\sigma_{ij}^{+} - \sigma_{ij}^{-}\right) \quad (7)$$

$$f(x) = \frac{1}{1+e^{-x}} \quad (8)$$

$$g(x) = \begin{cases} 1, & x > 2 \\ 0.25x + 0.5, & |x| \le 2 \\ 0, & x < 2 \end{cases} \quad (9)$$

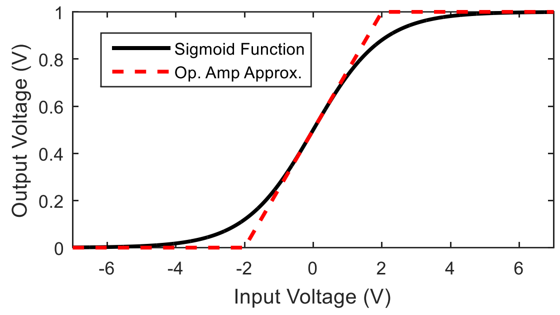

Figure 4. Plot displaying the traditional sigmoid function along with the approximation used in this work.

Fig. 5 displays a circuit diagram that shows how multiple neuron circuits can be patterned using a memristor crossbar. One of these circuits will be required for each layer in the autoencoder. Thus, five crossbars will be needed to implement the autoencoder presented in Fig. 2.

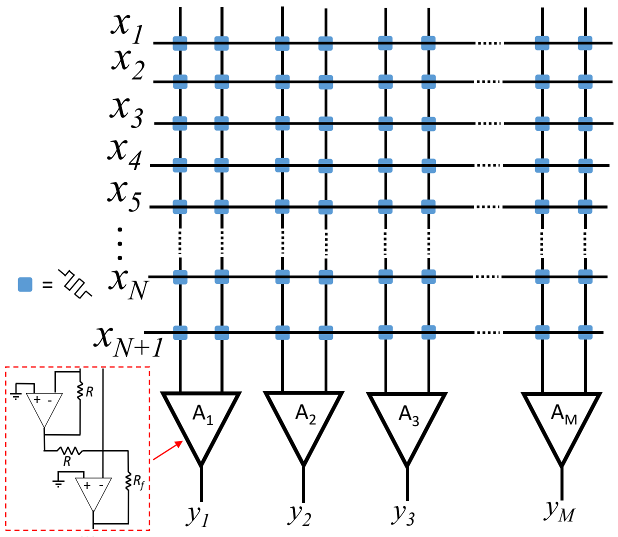

Figure 5. Circuit diagram displaying how a memristor crossbar can implement one layer of the proposed autoencoder.

4.3 Memristor Autoencoder

The training algorithm for the memristor crossbar based autoencoder is conceptually similar to that of a multilayer neural network, as the autoencoder is trained layer by layer. The proposed autoencoder has three hidden layers in addition to the input and output layers. The proposed training algorithm and training circuit has been adopted from [10,12,21-23]:

1) Apply the input pattern x_i to the input layer crossbar. Thus, the crossbar will compute the dot product DP_j for each neuron and propagate the output signal y_j.

2) For each output layer crossbar, the error is computed as the difference between the input x_i and the output y_j as in equation (10).

$$\delta_j = (x_j - y_j) f'(DP_j) \quad (10)$$

3) Backpropagate the error from each hidden layer neuron j as in equation (11).

$$\delta_j = \sum_k \delta_k w_{k,j} f'(DP_j) \quad (11)$$

4) Update the weights according to the error function with a learning rate η. The weight update rule is $\Delta w_j = \eta \delta_j x$.

5) Repeat this process until the error converges to a specified value.

In the proposed system we utilize on-chip learning, meaning that memristor resistance will be tuned during the training process as the result of a learning algorithm. The memristor resistance is altered due to a set of incoming voltage pulses applied to specific devices for specific times. One of the advantages of on-chip training is that it accounts for the variation in resistance present across an array of memristor devices [22]. The memristor device modeled for this study has a resistance ratio of approximately 200 and a write threshold voltage of 1.3V.

5 Memristor Based Autoencoder

An autoencoder is an unsupervised learning neural network. The detection technique is essentially threshold-based detection. Similarly, threshold-based intrusion detection and clustering was used by V. Nikulin [24], and Aron Laszka [25]. In our work, the input layer and output layer feature sizes are the same and can be considered one-dimensional vectors. Fig. 6 shows a block diagram of the proposed method for real-time intrusion detection.

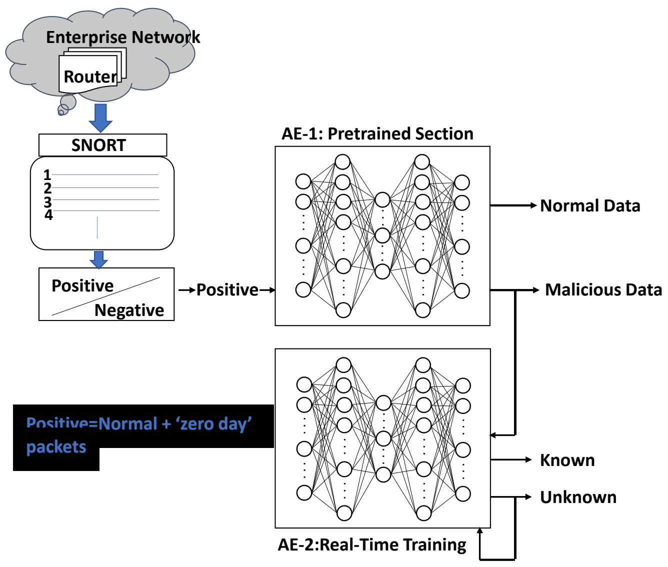

Figure 6. Schematic diagram of AE based real-time intrusion detection. AE-1 is pre-trained with 90% of the normal data and AE-2 is initialized with random weights.

The SNORT system is first used to catch all known attacks. Then all normal data and missed attack packets are sent to a pre-trained autoencoder. This autoencoder is trained with only normal packets. The test data consists randomly dispersed normal and malicious data, and the network does not have any predetermined knowledge of testing data types. However, as data passes through the network, vector distance can be measured between the input and the regenerated output packet. As the network was trained with only normal data packets, the malicious packets likely provide a larger vector distance than that of the normal packets. The network can then detect normal and malicious packets based on a vector distance threshold.

5.1 Real Time Intrusion Detection and Training

Most traditional intrusion detection systems are rule based, and if a zero-day attack arrives, this type of system is usually unable to detect it. So, the network needs some type of system that can protect against these new attacks. Work in [2] describes an extreme learning machine (ELM) for real-time intrusion detection, which includes a clustering manager, decision maker, and an update manager. Work in [3] proposed a hierarchical temporal memory (HTM), which is a machine learning system for real-time anomaly detection in video streams. J. Dromard et. al. implemented an unsupervised anomaly detection system based on knowledge databases [26]. G. Kathareios et. al implemented a behavioral unsupervised real-time anomaly detection system with a shallow autoencoder [27]. Ideally, intrusion detection should be as prompt as possible. Deep networks are presenting an opportunity for newer anomaly detection approaches that can significantly outperform other machine learning techniques. Thus, we developed an unsupervised real-time anomaly detection system using deep learning algorithms on a memristor based neuromorphic system. The system completes three steps that are required to detect unusual data (that the network has not seen before). First, we preprocess the NSL-KDD training data and separate 90% of the normal data for training. Then, we use the remaining 10% of the normal data and 10% of the malicious data for testing the first autoencoder (denoted at AE-1 in Fig. 7). Second, we train the first autoencoder (AE-1) and test it to determine detection accuracy. Third, the second autoencoder (AE-2 in Fig. 7) has no predetermined knowledge and was initialized with a set of random weights. AE-1 is able to detect malicious data types efficiently as it has knowledge of normal features. Furthermore, any detected malicious packets are sent to AE-2. Then, AE-2 will learn malicious data in real-time, and thus will be able to detect new data types as they are presented to the network.

5.2 Pre-Training

In this system, the training process does not use labels for learning the packet types. The training computation tracks the vector distance D between input and output samples as in equation (12) where X_i and Y_i are the input and output vectors respectively.

$$D = \sqrt{\sum(X_i - Y_i)^2} \tag{12}$$

In the last epoch, the mean distance $D_m = D/N$ and standard deviation D_{SD} (see equation (13)) are used to determine the threshold for packet detection. The value N denotes the total number of training samples.

$$D_{SD} = \sqrt{\frac{\sqrt{\sum(D-D_m)^2}}{N}} \tag{13}$$

Fig. 7 shows the real-time intrusion detection algorithm. An incoming packet is passed through the network, and the difference in vector distance Δ between the incoming packet and the mean distance D_m is computed. This difference is compared with the standard deviation D_{SD}. If Δ is larger than D_{SD}, the data packet is determined to be malicious. Likewise, if Δ is smaller than D_{SD}, the packet is determined to be a normal packet. Using this thresholding technique, normal input data will induce a lower value for Δ than when input data is malicious since the network is trained with only normal packet data. If the network determines that a packet is abnormal, it is sent to the next system for learning, weights are continuously updated based on these incoming packets, which leads to more robust real-time anomaly detection. The detection accuracy is defined as in equation (14). Here, N_F represents false detection and includes false positive and false negative cases. The value N_s represents the total number of samples in the test set.

$$Accuracy = \frac{N_S - N_F}{N_S} \times 100\% \tag{14}$$

In the following sections, we show that this thresholding technique is capable of generating strong intrusion detection results. Furthermore, we show the power benefits of using memristors to implement the presented systems.

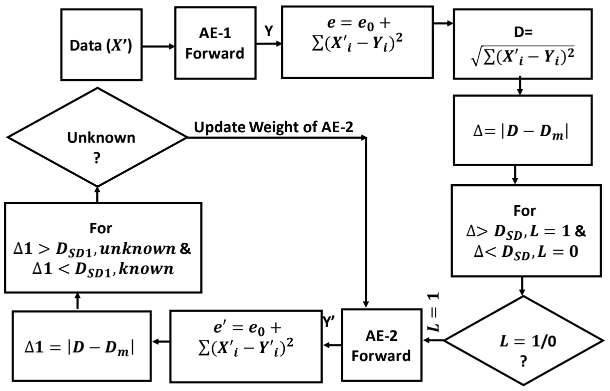
Data (X')
AE-1 Forward
Y
$e = e_0 + \sum(X'_i - Y_i)^2$
D= $\sqrt{\sum(X'_i - Y_i)^2}$
$\Delta = |D - D_m|$
For $\Delta > D_{SD}, L = 1$ & $\Delta < D_{SD}, L = 0$
$L = 1/0$?
$L = 1$
AE-2 Forward
Y'
$e' = e_0 + \sum(X'_i - Y'_i)^2$
$\Delta 1 = |D - D_m|$
For $\Delta 1 > D_{SD1}, unknown$ & $\Delta 1 < D_{SD1}, known$
Unknown ?
Update Weight of AE-2

Figure 7. Flowchart for the proposed real-time intrusion detection system. L = 0 indicates a normal packet and L = 1 indicates a malicious packet.

6 Results and Discussion

6.1 Intrusion Detection Results

We executed the proposed unsupervised training algorithm and tested it in real-time on the NSL-KDD dataset. The preprocessed training data contains a total of 67,343 normal packets and 58,630 malicious data packets. In these experiments, 90% of the normal data packets are separated from the total to be used as the initial training dataset. The last 10% of the normal data packets (a total of 6,734) and 10% of the malicious data packets (a total of 5,863) data were placed in random order and were used as the testing dataset.

An autoencoder with a 41→90→10→90→41 architecture was implemented in two forms. The first implementation is a straightforward software autoencoder to be used for baseline testing. The second implementation was developed to represent a simulated memristor crossbar. Both of these systems were designed and implemented in MATLAB. For the memristor based design, crossbar circuit functionality was built into the simulation for accurate representation of the hardware.

An input sample and its regenerated counterpart are displayed in Fig. 8; the regenerated output was determined using the memristor based AE design. The regenerated sample closely matches the original input sample.

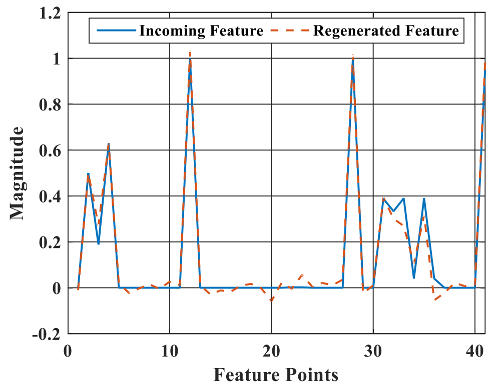

Figure 8. Sample demonstration of packet regeneration using the memristor crossbar based AE.

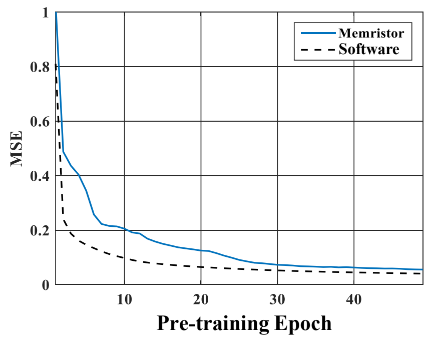

Figure 9. Plot showing MSE vs. pre-training epoch.

Fig. 9 displays the mean squared error (MSE) in the system during training for both the straightforward autoencoder and the memristor based system. The software autoencoder shows a smooth reduction of error whereas learning in the memristor system appears to be more fragmented and does not reach the minimum error value that the software autoencoder is able to attain. This is most likely because of the reduced dynamic range in the memristor weight values compared to the floating point weights in software. Furthermore, in the memristor based autoencoder, the approximated sigmoid function is used as the activation function, so this is likely another source of increased error. In both cases, error reduction diminishes after approximately 50 epochs.

The plot in Fig. 10 displays the standard deviation of the distance between the input and regenerated sample produced by the autoencoder during training. The standard deviation acts as a threshold for classification in the real-time during the testing phase. The software autoencoder shows lower threshold values when compared to the memristor case, and this makes sense because the error is lower in the system that does not possess the complications of memristor hardware. The effect of thresholding can be understood more clearly from the detection accuracy in Fig. 11. For the same initial conditions, the memristor crossbar shows slightly lower accuracy when compared to the software autoencoder.

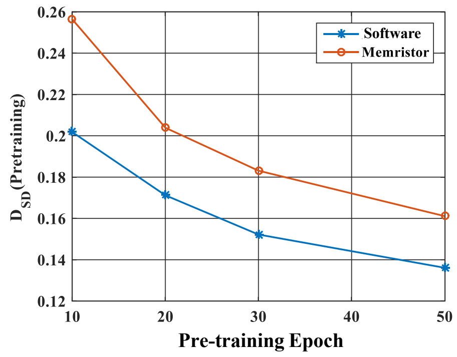

Figure 10. Plot showing the change in the standard deviation threshold for classification during training.

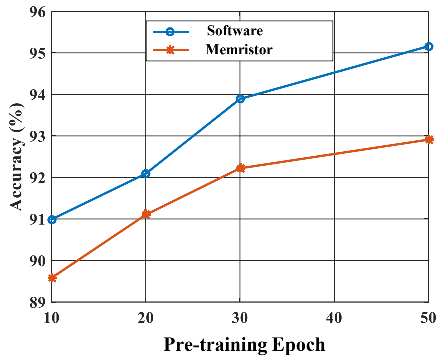

Figure 11. Plot showing intrusion detection accuracy vs. epoch.

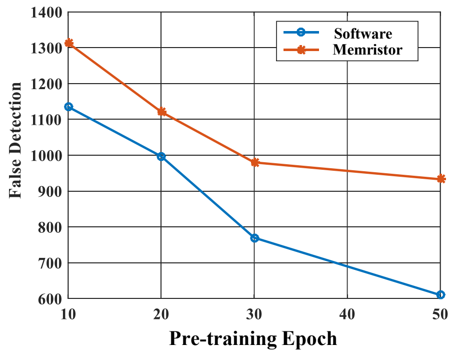

Figure 12. Plot showing false detection number vs. epoch.

The accuracy is 95.22% for the software autoencoder which exceeds the accuracies presented in [7, 16]. The accuracy of the memristor crossbar-based version is 92.91% when the initial conditions and pre-training period are matched to the software case. The accuracy of the memristor system could possibly be increased by implementing a deeper network (as in [14]) that uses five hidden layers with higher numbers of neurons, and we intend to study this in future work.

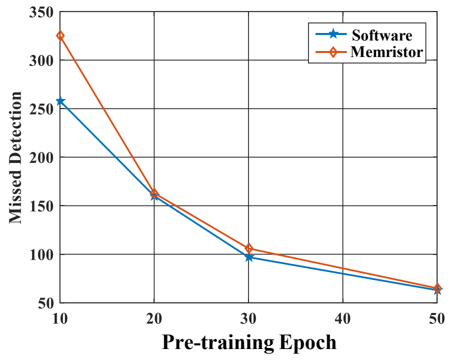

Figure 13. Plot showing the number of missed malicious packets during the pre-training process.

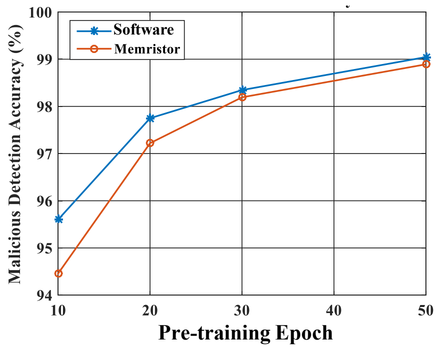

Figure 14. Plot showing the malicious packet detection accuracy using only the pre-trained autoencoder (AE-1).

Fig. 12 displays the total number of incorrect packet detections during training. The number is higher in the memristor based system, which is coherent with the earlier results in this paper. Fig. 13 presents the number of malicious data packets that passed through the autoencoder. From Table 1, we can see that with same initial conditions and the same training periods, the accuracy in the memristor based system is again slightly lower.

Table 1. Intrusion detection accuracy of the autoencoder system when implemented both in software and the simulated memristor crossbar.

Pre-trained Epoch	Global Accuracy	N_{MN}	N_{NM}	N_F	Case
50	95.22%	56	546	602	Software
50	92.91%	65	868	933	Memristor

6.2 Anomaly Detection Results

This experiment demonstrates how the memristor based autoencoder system is capable of real-time unsupervised learning when running an anomaly detection application. Four test sets (T1, T2, T3, and T4) for real-time training have been constructed with five data types, each of which share 100 samples. The packets used in this study comprise of normal packets, as well as four different attack types. A data subset from each of these categories is created where x_1 = *normal*, x_2 = *neptune*, x_3 = *satan*, x_4 = *ipsweep*, and x_5 = *back*. The test sets are then created using these data subsets as in equations (15) through (18). These test sets were developed to simulate the process of introducing anomalies into the system.

$$T_1 = x_1^1, x_2^1, x_1^2, x_2^2, x_1^3, x_2^3, \ldots \quad (15)$$
$$T_2 = x_1^1, x_2^1, x_3^1, x_1^2, x_2^2, x_3^2, \ldots \quad (16)$$
$$T_3 = x_1^1, x_2^1, x_3^1, x_4^1, x_1^2, x_2^2, x_3^2, x_4^2, \ldots \quad (17)$$
$$T_4 = x_1^1, x_2^1, x_3^1, x_4^1, x_5^1, x_1^2, x_2^2, x_3^2, x_4^2, x_5^2, \ldots \quad (18)$$

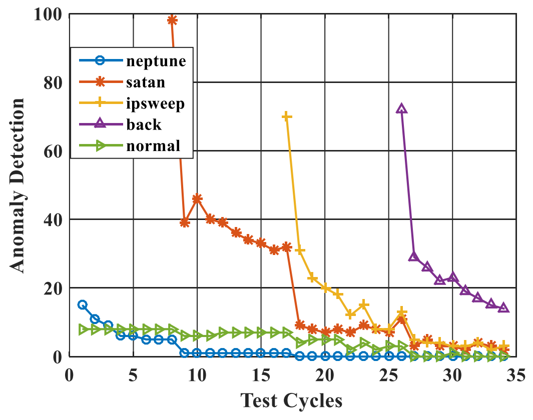

Figure 15. Real-time anomaly detection and anomaly learning over time on selected packet subsets.

Fig. 15 presents the results of the anomaly detection study. In this real-time learning example, the threshold and weight values were updated automatically after each cycle. First, an initialization step was completed to determine a starting point for the threshold and weight values using a training dataset that consisted of the testing dataset from the intrusion detection experiments. Then, the test set T1 was sent to the system, which consists of 100 normal data packets and 100 neptune denial of access (DoS) attacks. The T1 test set was ordered so that normal and neptune packets were alternated.

The AE-1 is very efficient for the detection of malicious data (see Fig.14) as it is trained with only normal data, and all malicious data is then sent to AE-2 for learning first and detection second. The network is continuously learning over time and the number of anomalies recognized decreases as more occurrences of the same anomaly become more prevalent. However, when a new type of packet comes into the network, it is then recognized as an anomaly. For example, when the T2 subset is applied to the autoencoder, it flags the previously unlearned packets 98% of the time (see Fig.

14). Although, the more times an attack type shown to the system, the less likely it is to be flagged as unusual. Thus, this type of packet becomes less interesting to the system as its presence becomes more frequent. The network was then tested with the T3 and T4 datasets. Once again, Fig. 15 shows that new datatypes are flagged by the system with a very high probability. This high probability then decays over time due to the repetition of this new data. This online learning process is continuous and over time, the AE-2 retains its ability to detect anomalies.

6.3 Power, Area, and Timing Analysis

The area, power, and energy have been computed for the memristor crossbars and peripheral circuits. Table II shows a total system area of 0.00271 mm^2. The system consumes 20.6 mW of power during the training phase and 7.56 mW during the recognition phase when running at the maximum clock speed.

Table 2. Power, area, and timing estimates for the proposed neuromorphic anomaly detection system.

Area (mm^2)	0.00271
Training Power (mW)	20.6
Training Time (µs)	4.02
Training Energy: One Sample (nJ)	82
Recognition Power (mW)	7.56
Recognition Time (µs)	0.384
Recognition Energy: One Sample (nJ)	2.90

6 Conclusion

Unsupervised real-time intrusion detection and learning has been studied in a traditional autoencoder, as well as one implemented using memristor crossbar technology. The memristor crossbar designs were able to successfully reproduce the functionality of the software autoencoder with only a slight reduction in accuracy. The memristor crossbar based autoencoder design successfully implemented both network intrusion detection and anomaly detection. Network intrusion detection was performed with an accuracy of 92.91% with a malicious packet detection accuracy of 98.89%. In the case of anomaly detection, unrecognized datatypes were not only recognized, they became familiar to the system as their presence became more frequent. In the future we plan to perform an energy and power comparison to determine more accurately the throughput efficiency benefits of the memristor based system.

REFERENCES

[1] Setareh Roshana, Yoan Michel, Anton Akusokd, Amaury Lendasse, "Adaptive and online network intrusion detection system using clustering and Extreme Learning Machines", Journal of the Franklin Institute Vol. 355, pp. 1752–1779, Iss. 4, March 2018

[2] Setareh Roshana, Yoan Michel, Anton Akusokd, Amaury Lendasse, "Adaptive and online network intrusion detection system using clustering and Extreme Learning Machines", Journal of the Franklin Institute Vol. 355, pp. 1752–1779, Iss. 4, March 2018

[3] Subutai Ahmada, Alexander Lavina, Scott Purdya, Zuha Agha, "Unsupervised real-time anomaly detection for streaming data", Vol. 262, pp. 134-147, Neurocomputing, 2017

[4] Matilda Rhode b, Pete Burnap b, Kevin Jones, 'Early-stage malware prediction using recurrent neural networks', computers & security Vol. 77, August 2017, pp. 578-594

[5] P. K. Prajapati and M. Dixit, "Un-Supervised MRI Segmentation using Self Organised Maps," International Conference on Computational Intelligence and Communication Networks, pp. 471-474, India, December 2016

[6] Raghavendra Chalapathy, Sanjay Chawla, "Deep Learning for Anomaly Detection: A Survry", arXiv:1901.03407v2 [cs.LG], Jan 2019

[7] Md Zahangir Alom, Tarek M. Taha, 'Network intrusion detection for cyber security using unsupervised deep learning approaches', National Aerospace and Electronic Conference (NAECON), June 2017

[8] Md Zahangir Alom and Tarek M. Taha, "Network Intrusion Detection for Cyber Security on Neuromorphic Computing System", 2017 International Joint Conference on Neural Networks (IJCNN), 14-19 May 2017, Anchorage, AK, USA.

[9] Bruno Bogaz Zarpelão, Rodrigo Sanches Miani, Cláudio Toshio Kawakani Sean Carlisto de Alvarenga, "A survey of intrusion detection in Internet of Things", Journal of Network and Computer Applications, Vol. 84, pp. 25-37, 2017

[10] C. Yakopcic, M. Z. Alom, and T. M. Taha, "Memristor Crossbar Deep Network Implementation Based on a Convolutional Neural Network," IEEE/INNS International Joint Conference on Neural Networks (IJCNN), pp. 963-970, Vancouver, BC, July 2016.

[11] Raqibul Hasan, Tarek M. Taha, 'Memristor crossbar based unsupervised training', National Aerospace and Electronic Conference (NAECON), Dayton, USA, June 2015

[12] Yakopcic and T. M. Taha, "Analysis and Design of Memristor Crossbar Based Neuromorphic Intrusion Detection Hardware," IEEE/INNS International Joint Conference on Neural Networks (IJCNN), pp. 1-7, Rio de Janeiro, Brazil, Julys, 2018

[13] Venkataramesh Bontupalli, Chris Yakopcic, Raqibul Hasan, and Tarek M. Taha. 2018. Efficient Memristor-Based Architecture for Intrusion Detection and High-Speed Packet Classification. *J. Emerg. Technol. Comput. Syst.* 14, 4, Article 41 (November 2018), 27 pages

[14] Mahmood Yousefi-Azar, Vijay Varadharajan, Len Hamey and Uday Tupakula, 'Autoencoder-based Feature Learning for Cyber Security Applications', International Joint Conference on Neural Networks (*IJCNN*), Anchorage, USA, May 2017.

[15] Zhipeng Li, Zheng Qin, Kai Huang, Xiao Yang, and Shuxiong Ye, "Intrusion Detection Using Convolutional Neural Networks for Representation Learning", Springer, 2017, pp. 858–866.

[16] Fahimeh Farahnakian, Jukka Heikkonen, 'A Deep Auto-Encoder based Approach for Intrusion Detection System', International Conference on Advanced Communications Technology (ICACT), South Korea, February 2018.

[17] NSL-KDD data: https://www.unb.ca/cic/datasets/nsl.html.

[18] Solane Duquea, Dr. Mohd. Nizam bin Omar b,' Using Data Mining Algorithms for Developing a Model for Intrusion Detection System (IDS)', Procedia Computer Science vol. 61, 46 – 51, November 2015

[19] L. O. Chua, "Memristor—The Missing Circuit Element," IEEE Transactions on Circuit Theory, vol. 18, no. 5, pp. 507–519 (1971).

[20] T. M. Taha, R. Hasan, C. Yakopcic, and M. R. McLean, "Exploring the Design Space of Specialized Multicore Neural Processors," IEEE/INNS International Joint Conference on Neural Networks (IJCNN), pp. 1-8, Dallas, TX, August 2013

[21] R. Hasan, T. M. Taha, and C. Yakopcic, "On-chip Training of Memristor Crossbar Based Multi-layer Neural Networks," Microelectronics Journal, vol. 66, no. 8, pp. 31-40, Aug. 2017.

[22] Raqibul Hasan and Tarek M. Taha, "Enabling Back Propagation Training of Memristor Crossbar Neuromorphic Processors", International Joint Conference on Neural Networks (***IJCNN***), Beijing, July 2014

[23] B. R. Fernando, R. Hasan and M. Tarek Taha, "Low Power Memristor Crossbar Based Winner Takes All Circuit," 2018 IJCNN, Rio de Janeiro, 2018, pp. 1-6.

[24] V. Nikulin, 'Threshold-based clustering with merging and regularization in application to network intrusion detection', Computational Statistics & Data Analysis vol. 51, pages 1184 – 1196, November 2006

[25] A. Laszka, W. Abbas, S. S. Sastry, Yevgeniy Vorobeychik, Xenofon Koutsoukos, 'Optimal Thresholds for Intrusion Detection Systems', Proceeding of the Symposium and Bootcamp on the Science of Security, Pittsburgh, Pennsylvania, USA, April 2016, P. 72-81.

[26] Dromard J., Roudière G., Owezarski P. (2015) Unsupervised Network Anomaly Detection in Real-Time on Big Data. In: Morzy T., Valduriez P., Bellatreche L. (eds) New Trends in Databases and Information Systems. ADBIS 2015. Communications in Computer and Information Science, vol 539. Springer, Cham

[27] G. Kathareios, A. Anghel, Akos M, R. Clauberg, M. Gusat, "Catch It If You Can: Real-Time Network Anomaly Detection With Low False Alarm Rates", IEEE International Conference on Machine Learning and Applications, Cancun, Mexico, December 2017.