Computers, Materials & Continua
DOI:10.32604/cmc.2022.019847
Article

Tech Science Press

User Behavior Traffic Analysis Using a Simplified Memory-Prediction Framework

Rahmat Budiarto[1,*], Ahmad A. Alqarni[1], Mohammed Y. Alzahrani[1], Muhammad Fermi Pasha[2], Mohamed Fazil Mohamed Firdhous[3] and Deris Stiawan[4]

[1]College of Computer Science & IT, Albaha University, Alaqiq, 65779-7738, Saudi Arabia
[2]Malaysia School of Information Technology, Monash University, Bandar Sunway, 47500, Malaysia
[3]Department of Information Technology, University of Moratuwa, Moratuwa, 10400, Sri Lanka
[4]Faculty of Computer Science, Universitas Sriwijaya, Indralaya, 30662, Indonesia
[*]Corresponding Author: Rahmat Budiarto. Email: rahmat@bu.edu.sa
Received: 28 April 2021; Accepted: 15 June 2021

Abstract: As nearly half of the incidents in enterprise security have been triggered by insiders, it is important to deploy a more intelligent defense system to assist enterprises in pinpointing and resolving the incidents caused by insiders or malicious software (malware) in real-time. Failing to do so may cause a serious loss of reputation as well as business. At the same time, modern network traffic has dynamic patterns, high complexity, and large volumes that make it more difficult to detect malware early. The ability to learn tasks sequentially is crucial to the development of artificial intelligence. Existing neurogenetic computation models with deep-learning techniques are able to detect complex patterns; however, the models have limitations, including catastrophic forgetfulness, and require intensive computational resources. As defense systems using deep-learning models require more time to learn new traffic patterns, they cannot perform fully online (on-the-fly) learning. Hence, an intelligent attack/malware detection system with on-the-fly learning capability is required. For this paper, a memory-prediction framework was adopted, and a simplified single cell assembled sequential hierarchical memory (s.SCASHM) model instead of the hierarchical temporal memory (HTM) model is proposed to speed up learning convergence to achieve on-the-fly learning. The s.SCASHM consists of a Single Neuronal Cell (SNC) model and a simplified Sequential Hierarchical Superset (SHS) platform. The s.SCASHM is implemented as the prediction engine of a user behavior analysis tool to detect insider attacks/anomalies. The experimental results show that the proposed memory model can predict users' traffic behavior with accuracy level ranging from 72% to 83% while performing on-the-fly learning.

Keywords: Machine learning; memory prediction framework; insider attacks; user behavior analytics

1 Introduction

Nearly half of the incidents in enterprise security have been triggered by insiders. Sophisticated insider threat attackers include leavers, outsiders, and unknowing innocents. A conventional solution to this problem, i.e., the perimeter security tool, is no longer able to provide proper insights to stop insider attacks. To produce proper insights, the security tool needs to monitor, record, and analyze user behaviors across the entire organization without sacrificing privacy or performance.

A technique called user behavior analytics has become particularly useful in providing solutions that have flexible pattern recognition when rules are unsuitable, which humans simply cannot achieve due to the extremely large amount of data involved. It is difficult to distinguish anomaly/malware from normal behaviors due to sophisticated attack techniques. A compromised machine/node may pretend to be a normal user requesting a service from a server to gain classified information. The main issue with the traditional intrusion detection systems (IDSs) is the use of signature-based files, which increase in size over time and affect the detection accuracy and detection time. Thus, researchers have begun to use intelligent techniques to overcome the issue. As modern network traffic has dynamic patterns and a huge volume, detection systems that use deep-learning techniques require more time to be retrained for new traffic patterns as they cannot fully perform online (on-the-fly) learning. Therefore, an intelligent detection system with an on-the-fly learning capability is required.

Contribution: This research work has various contributions in the domain of user traffic behavior analysis [1].

(1) First, a dataset generated from real traffic and combined with malware traffic was created as an alternative for benchmarking anomaly detection systems.
(2) Second, a memory-prediction framework model has been proposed as a basis for a user behavior traffic analysis in real-time, i.e.,: simplified single cell assembled sequential hierarchical memory (s.SCASHM) model inspired by the modern neuroscience theory on the neocortex.
(3) A new type of memory system called Sequential hierarchical superset (SHS) platform and Single neuronal cell (SNC) model have been introduced.
(4) Lastly, the memory-prediction model is implemented as anomaly detection system. The model will help enterprises to defend their networks from insider attacks as it has on-the-fly learning capability.

2 Related Work

Nowadays, Internet of Things (IoT) and cloud computing networks are dominant in the Internet. The fast development in wireless sensor network technology has made the IoT network becomes more complex in term of applications [2] and routing [3]. Naturally, security turns out to be extremely important, especially in cloud environment [4]. Therefore, more sophisticated tools to analyze cybersecurity and cybercrime are required [5], including the use of artificial intelligence techniques [6].

Stiawan et al. [7] have combined interaction behavior and user attitude as habitual activities with users' known activities, such as browsing the web, accessing database applications, etc., to recognize normal, suspicious, or malicious internal users through the classification of the packets in network traffic. Sun et al. [8] explored and discussed user travel behaviors, traffic satisfaction, key determinants, impacts, development planning, and policies to determine the impact of a

shared human transportation system. Other recent works on insider attacks that consider user behavior analytics (UBA) are presented in [9–11].

Karbab et al. [12] utilized a deep-learning technique for the sequence classification of Android malware and proposed an automatic Android malware detection and attribution framework called MalDozer. Shaukat et al. [13] measured the performance of three machine-learning techniques, i.e.: the support vector machine (SVM), decision tree, and deep belief network, in spam detection, intrusion detection, and malware detection based on frequently used and benchmark datasets. Evolving methods of time-series anomaly detection and the way computational methods can be applied in this domain in the future are investigated in [14]. The authors concluded that most of the existing online anomaly detection systems use supervision and that few use an unsupervised approach. Thus, designing online versions of ensemble and multimodal learning approaches will be an interesting future research direction. Hawkins et al. [15] introduced the concept of a memory-prediction framework in 2005. Following this, more researches on the theoretical concepts of the human memory system were published [16–18]. A novel framework to reveal the human neocortex function was proposed by Hawkins et al. [19,20]. Furthermore, as more experiments were conducted, it was found that grid cells, such as neurons, may also be present in the neocortex [21,22]. The memory-prediction framework has been implemented in various fields, including object identification [23,24], medicine [25], online learning [26,27], real-time network traffic anomaly detection [28–31], and network forensics [32].

Cui et al. [26,27] discussed online sequence learning with an unsupervised neural network model. The authors implemented the Hierarchical Temporal Memory (HTM) spatial pooler. The spatial pooler models how neurons learn feed-forward connections and form efficient representations of input. Ahmad et al. [28] discussed the implementation of the HTM on online/streaming network traffic anomaly detection. The experimental results using the Numenta Anomaly Benchmark dataset [29] showed that the proposed system fulfills the requirements for streaming applications. Nevertheless, the proposed system requires powerful computing resources.

Pasha et al. [32] implemented the human neocortex memory system using a sequential hierarchical superset based on the memory prediction framework as a new way to create brain-like artificial intelligence tools without involving neural networks. The system was applied to perform an automated network forensic analysis on a local area network. This paper adapts the idea from the work in [32] by simplifying the sequential hierarchical superset to speed up the system convergence in learning. The proposed model was then adjusted to specifically recognize user traffic to learn the behavior of user. It is a novel brain-inspired approach to detect insider threats by continuously modeling daily user traffic behaviors compared to one off modeling that commonly used in a deep-learning based approach.

3 Data and Methodology

This section discusses the dataset and the proposed model along with its implementation on user traffic behavior analysis.

3.1 Dataset

The Numenta Anomaly Benchmark is an existing benchmark dataset used for real-time network traffic monitoring; however, the current Numenta Anomaly Benchmark is limited to data streams containing a single metric plus a timestamp, and thus the error analysis from the benchmark indicates that the errors across various detection algorithms are not always correlated [29]. Therefore, we intentionally created a dataset for the experiments. Packets of various specific

applications were captured using packets' sniffing software from the network at the college of Computer Science and IT Building, Albaha University (the network consists of six segments with more than 100 users) for a three-month period. The captured data over a two-month period were used as a training dataset, and the data from the third month were used for testing/experimenting with the user behavior analysis. The captured packets were played back on a local network setup to mimic some application packets, which were later injected to simulate a real corporate network. Anomalous traffic packets were taken from Mid-Atlantic Collegiate Cyber Defense Competition (MACCDC) 2012 dataset [33].

3.2 The Simplified SCASHM (s.SCASHM)

The s.SCASHM is inspired by neuroscience's memory-prediction framework theory on the human neocortex [15]. To model the human neocortex as closely and simply as possible, the s.SCASHM is designed to consist of two main parts: the SHS platform and the SNC model. Fig. 1 depicts the general architecture of the s.SCASHM.

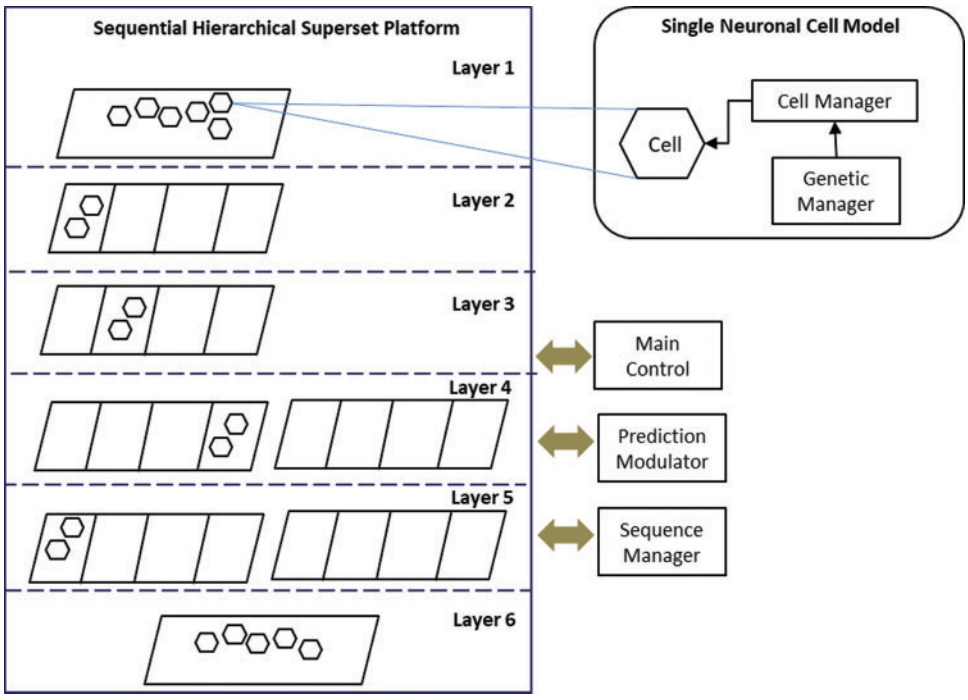

Figure 1: The s.SCASHM architecture

The SNC model is designed to have two different managers: the cell manager and the genetic manager. The cell manager is the core part of the SNC model. It manages the creation, parameters, and operation of every cell. The purpose of having a separate genetic manager is to provide a manageable gene distribution among each cell for performance improvement. A cell produced by the cell manager is arranged as such that it interconnects with other cells in a hierarchy inside the SHS platform. The nature of the SNC model is fundamentally different from neurons commonly known in the artificial cell assembly model or neural network. The data are not stored in the synapse, which needs to be strengthened or trained if the value stored by the neural network is changed. The purpose of examining the connection between cells is to determine which cell may

be activated next when one cell is activated in a particular sequenced assembled cell. The cell itself is a container of data. Thus, when the data need to be updated, the required training time can be significantly reduced as no weights need to be retrained.

The SHS platform consists of five modules: the sequence manager, the prediction modulator, the input-preprocessing, the main control center, and the six-layer hierarchical columnar container. Among these five modules, the main control center is the core module that controls the assembled single neuronal cell model inside the six-layer hierarchical columnar container and manages the data flow between each layer [19,20]. This work proposes a simplified structure of the columnar region because the actual structure of the human neocortex cortical region is highly complex and not well understood yet [15]. The simplified SHS platform is designed to have a region with four rows of a columnar architecture instead of having six rows per region as in the human neocortex. For further simplification, Layer 1 and Layer 6 are designed to have one region with an unlimited number of columns (see Fig. 2), while Layers 2 to 5 can have multiple regions and the number of columns for each region depends on its gene. Fig. 2 shows the four-row columnar region architecture of the s.SCASHM platform.

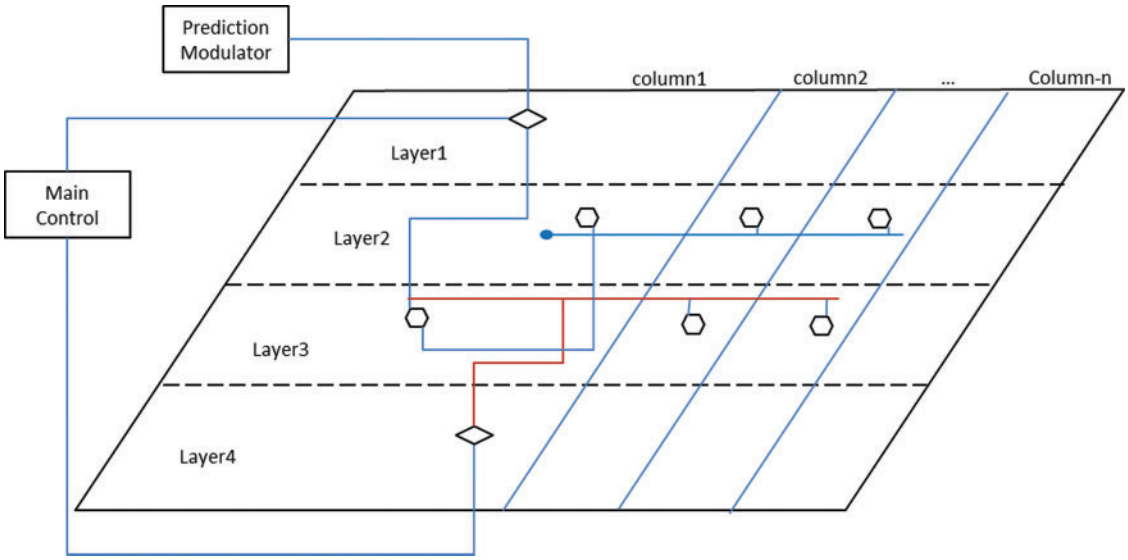

Figure 2: The architecture of the simplified SHS of the s.SCASHM

The six-layer hierarchical columnar container is the actual platform where an assembled SNC model is placed. Similar to the human neocortex, each layer contains a region, and the region has a further layered columnar architecture. The proposed SNC model forms an assembled cell in a sequential manner inside the s.SCASHM memory model without the complexity of training and learning computations. Thus, both the SNC model and the SHS platform are integrated and complement each other with the aim to have a working cell assembly-based memory implementation following the human neocortex columnar pyramidal cell architecture inside its six-layered hierarchical memory structure.

The column's first row and last row are where the input-output handler is placed. Both the column's first row and the last row are connected to the main control center module. In addition, the column's first row is connected to the prediction modulator. The column's third row is the SNC

holder's row, which implements a standard SNC cell that contains the data, and the cell's input gate is connected to the input-output handler in the fourth row as well as to other cells' input gate in the same row defined in the SNC's cell manager connection map. The column's second row is the inhibitory layer, which implements an inhibitory cell. An inhibitory cell is an SNC cell without data. An inhibitory cell's input and output gate is not connected to any cell as it has no data but instead receives an inhibitory signal in its third interface and sends inhibitory signals to other inhibitory cells inside the particular region where the column is located. The purpose of this inhibitory cell is to prevent other cells in different columns that are not part of the current active set of the sequence from being activated. Fig. 3 illustrates the SNC architecture. All the other four modules of the SHS platform connect to the main control center, where one common main algorithm is proposed. The term "common algorithm" is used because the same algorithm is applied at all layers inside the sixth layer of the hierarchical columnar container, which in turn allows it to control and to manage the data flow between each layer.

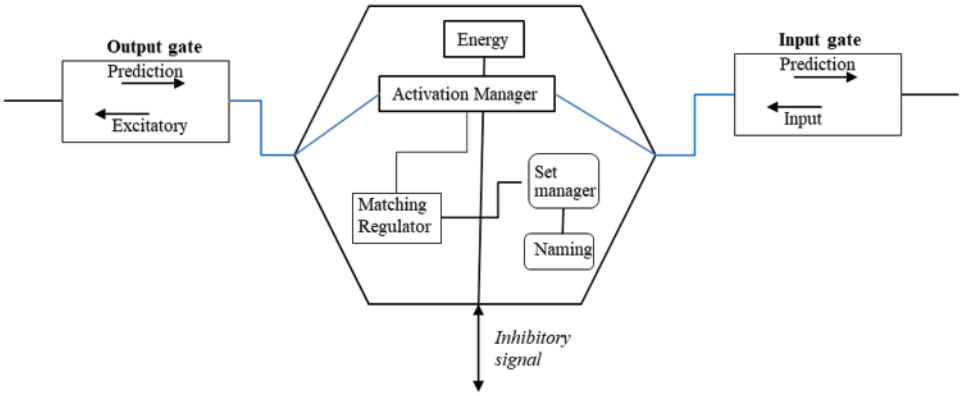

Figure 3: The single neural cell model

To understand the fundamental activities of the s.SCASHM with its SNC and SHS platform, Fig. 4 visualizes the step-by-step workflow of the s.SCASHM when dealing with new, first-time input of the string "Brain Model." After receiving the input string, first, the SHS input-preprocessing module breaks the input into its smallest forms, outputting two sets of inputs. The first input is a set of letters that forms the word "brain," while the second input is a set of letters that forms the word "model."

These two sets of inputs are then sent to the SHS' main control center in sequential order. The SNC cell manager is then called to compute similarity matching on each of the inputs. Because this input is new and has never been presented before, the SNC cell manager then returns no match triggering the SHS main control center to initiate a new memory creation process as part of the SHS' common main process. The new memory creation process includes initiating the cell manager to create new cells to hold the new memory, putting these newly created cells into the SHS platform's lowest layer (Layer 6), and creating the connections with default weights between cells based on its sequence.

The newly created links between cells are stored in the SNC cell manager's connection map table, and the cell's location in the SHS platform is stored in SNC cell manager's information details table. A set of cells that is linked together to form a sequence is then placed in the upper layer (Layer 5) and automatically added to the SHS sequence manager's naming convention table

with an auto-assigned name, which in this case is "Unknown-1" and "Unknown-2" (see Fig. 4). It then continues to form a set of sequences between these two new sets to form a superset placed in the next upper layer (Layer 4) and auto-assigns the name "Unknown-3."

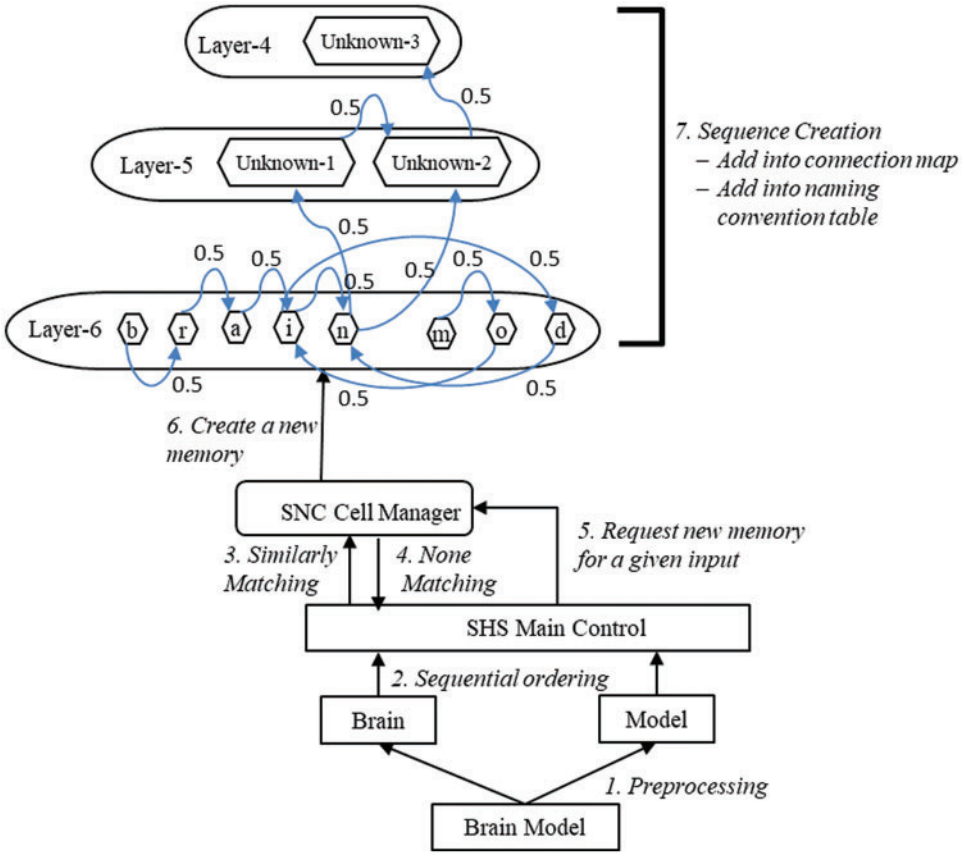

Figure 4: s.SCASHM step-by-step workflow when presented with new input samples

Upon completion, the s.SCASHM has knowledge of a few letters, two words, and one short sentence. The next time, when it is presented with the string "Brain Research," the same process occurs again; however, this time, after the letter "b" is presented, the s.SCASHM, using the SHS prediction modulator, makes a prediction that the next letter might be "r". When the next input letter "r" is presented, the prediction is matched, and the set of the sequence {"b", "r", "a", "i", "n"} becomes a predicted set. The same process occurs in the higher layer (Layer 5), and the superset of sequence {"Unknown-1," "Unknown-2"} is then used by the SHS prediction modulator to predict what the next set might be after the set "Unknown-1." This time, the prediction fails because the next input presented is the word "Research" and not "Model." In the case of prediction failure, the s.SCASHM will go through a new memory creation process again for the word "Research," and a new link in the SNC cell manager's connection map table is created to form the new sequence of {"brain," "model"}. Generally, these are the typical processes that occur in the s.SCASHM.

The s.SCASHM was implemented as a memory prediction framework-based online UBA tool for analyzing users' traffic behavior. In the online UBA tool implementation, the captured raw

traffic stream is preprocessed into the sequence of individual network packets, and each byte inside the network packet is represented by its smallest form in a sequence of bits. A 2048 bits vector is then used as the Sparse Distributed Representation (SDR) implementation to represent this sequence of bits. At each point in time, this 2048 bits vector is fed to the s.SCASHM memory network for analysis. On the output of the s.SCASHM network, additional computations, i.e.,: prediction error and probability or likelihood that the event is in an anomalous state, are executed. A threshold on this likelihood determines whether an anomaly is detected or not. Fig. 5 illustrates the implementation of the proposed online UBA tool.

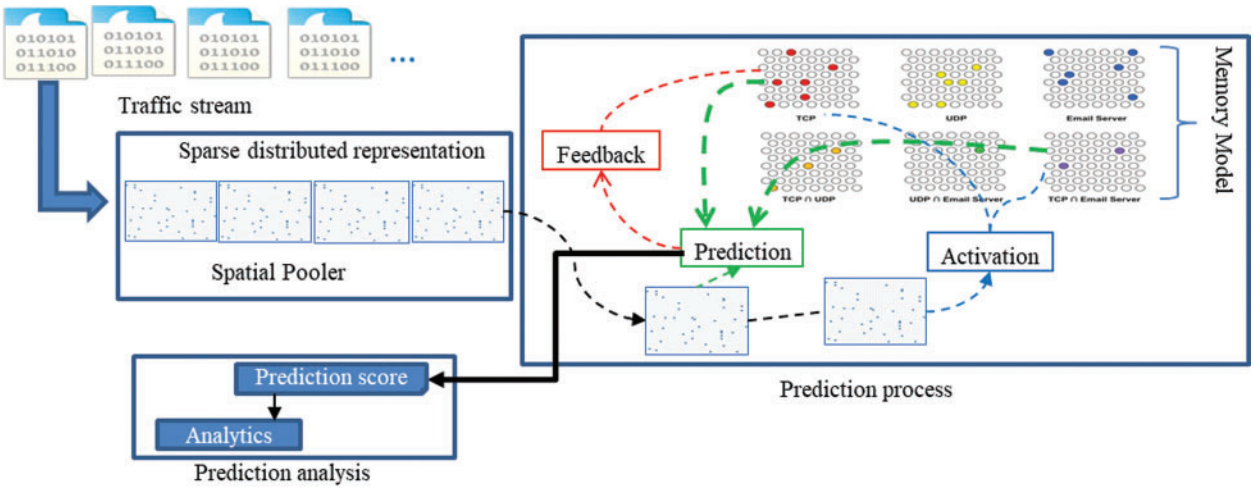

Figure 5: The s.SCASHM for an online UBA implementation diagram

For the purpose of comparison with deep learning, VGG-16 deep learning is used. VGG-16 is a convolutional neural network model introduced by Simonyan et al. [34]. The model is in the top five in test accuracy in ImageNet 2014 as it achieves 92.7% accuracy. The VGG-16 was selected for comparison because it does not use many hyper-parameters, so it is a simpler architecture model.

3.3 Experimental Setup

Topology

The experimental setup for playing back the captured traffic to create the simulation consists of three computers connected to a network switch. PC-1 is used for crafting traffic packets, and a packet generator injects traffic into the network segment. The server is configured to host four virtual servers. The s.SCASHM (as the detection engine) is installed on PC-2. PC-2 is connected to a mirrored port of the switch, and thus the detection engine can see all the traffic in the network segment. PC-1 and PC-2 are computers with specifications of Intel Core i5-8250U processor with 4 GB RAM and 500 GB of storage capacity. The server's specification is an Intel Core i7-10510Y processor with 8 GB RAM and 1 TB of storage capacity. All computers run Windows 10 as their operating system. The memory prediction engine is implemented in Python programming language, and the deep-learning engine is implemented using Tensorflow software.

Scenarios

Once a small local area network with a data mirroring port is set up, the simulated one-month dataset is created as per the following steps:

- Design and plan a one-month traffic simulation (when and what anomalous traffic appears, etc.).
- Set eight users/nodes (four server nodes and four user nodes) for the simulation.
- Craft the required traffic packets manually, including anomalous/malware packets.
- Begin injecting traffic into the network and at the same time capturing the traffic through the mirroring port.
- After the planned traffic for one month has been injected and captured, manually label the simulated anomaly as well as specific application traffic (this label was used during the experiments to verify the results).
- Save the captured traffic into a file in the .pcap format as the dataset for the UBA experiments.

4 Results and Discussion

Experiments on the proposed s.SCASHM were carried out by feeding the raw data traffic flow for continuous learning and detection. In deep-learning experiments, for each month of data, the first 20 days of decoded traffic are used as training data, and the last 10 days are used as testing data. The average is used for the final result.

4.1 Results

Experiment 0*—On cell activity statistics of the proposed s.SCASHM during training*

The analysis of the individual traffic flows and their contents are essential for a complete understanding of network usage. In this experiment, a reconstructive traffic analysis was carried out. The archived three-month traffic flows were analyzed to identify behavior patterns. The captured raw packet was converted into a data string before sending the data to the s.SCASHM. Tab. 1 shows the configuration control file parameter used during the experiment. As the purpose of this experiment was to validate and evaluate the s.SCASHM's memory, the evaluation cycle parameter for the SHS performance improvement evaluation through gene mutations was set to 1000 cycles (where one cycle is equal to the operation of one set of input sequences processed). The evaluation cycle for SNC is set to five (where one cycle is equal to the number state changes of cells from "active" to "tired" to "non-active" and back to "active" again).

Table 1: s.SCASHM configuration control files setup

Parameter	Value
State	"SCASHM" = Activated
SHS_PredictionMatchingDegree	0.75
SHS_SequenceTimePeriod	5000 (msec.)
EvaluationCycle	SHS = 1000, SNC = 5
InputReceiver	SCAHSM
Log	SCAHSM

The training part is a supervision process period of the s.SCASHM's memory. The s.SCASHM implements auto-assigned naming for every object it sees. This labeling process is useful for a better evaluation of the s.SCASHM in terms of how accurate it recognizes a memorized object. Tab. 2 shows a sample of some entries inside the SHS' sequence manager that names convention tables after the whole set of experiments was finished. The table shows a supervised set labeled for easy reference. The content of a set shown is either the smallest-form data or an ID of other cells. A set of actual data is a set that is formed in the layer that connects directly with the input, which is Layer 6. A set of other cells' IDs is a set that contains a sequence of other cells represented by their IDs.

Table 2: Sample of some entries inside SHS' sequence manager naming convention table

ID	Content	Name	Type	Timestamp
1	{08, 00}	ethIP	Std	13:04:45.112
2	{46, 06}	ipver-tcp	Std	13:04:46.046
3	{00, 55}	httpPort	Std	13:04:46.072
4	{1, 2}	tcp	Std-ID	13:06:21.489
5	{1, 2, 3}	http	Std-ID	13:07:33.048
6	{86, dd}	ethIp6	Std	13:07:34.481
7	{6e, fe}	ipv6ver-icmp	Std	13:08:13.653
8	{6, 7}	icmp	Std-ID	13:08:57.203
9	{60, 06}	ipv6ver-tcp	Std	13:10:28.322
10	{6, 9, 3}	http6	Std-ID	13:10:29.406
11	{08, 06}	ethArp	Std-ID	13:13:42.811
12	{00, 01}	arpOptCode	Std	13:15:09.678
13	{11, 2, 12}	arp	Std-ID	13:15:55.576
14	{13, 13, 13, 13}	arp-flow	Std-ID	13:32:19.864

Tab. 3 shows the number of SNC cell statistics inside the SHS platform before and after the supervision is made. From the statistics, it is clear that without supervision, the s.SCASHM memorizes every single set of sequences it is presented with. During the supervision process, unnecessary sets are eliminated, such as the set that contains almost identical content, and also most of the sets are trimmed to have a better representation of the actual packet structure's properties. From 236 cells in Layer 1, after supervision, the number of cells in layer 1 becomes 37 cells only. Similarly, in Layer 6, almost 3.2 million cells become almost 370,000 cells after supervision. Tab. 3 shows that the number of sets of sequences being formed decreases as the s.SCASHM begins to build comprehensive knowledge of the traffic.

Table 3: Number of cells in SNC before and after supervision during training

Supervision	Layer1	Layer2	Layer3	Layer4	Layer5	Layer6
Without	708	3111	41766	283857	947886	3184944
With	111	834	4257	58922	11823	369526

Fig. 6 shows the visualization of a portion of assembled SNC cells after the whole training process is completed. White-colored cells are cells in Layer 6, while blue-colored cells are in Layer 5. Green- and red-colored cells are in Layer 4 and Layer 3, respectively. A solid line represents the connection to the neighboring cell in the same layer and in the same region, while the dashed lines represent a connection to other cells in a different layer. Both connections to the neighboring cells and connections to other cells in different layers have a weight of the particular connection. Cells at Layer 3 and below represent an abstract representation of the traffic behavior.

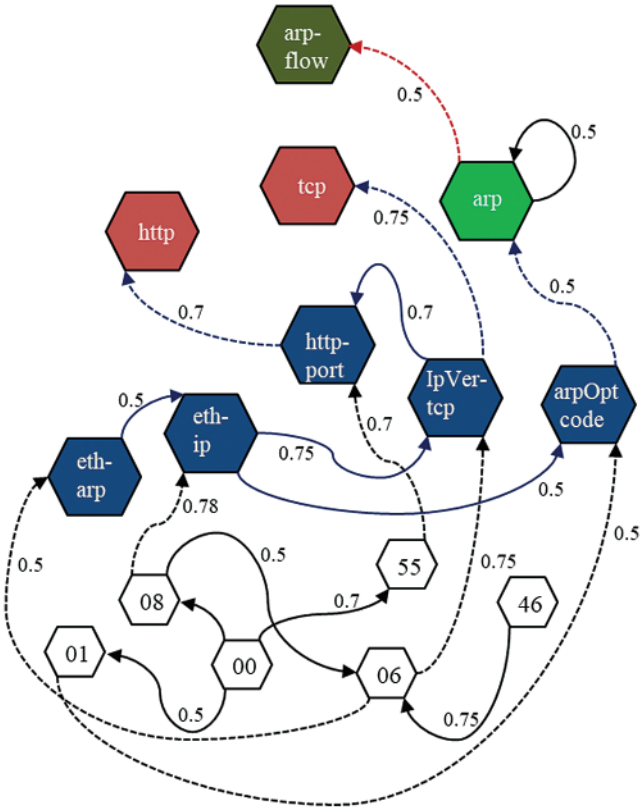

Figure 6: Cell construction and its statistics

Experiment 1*—Automatic user traffic classification*

In this set of experiments, we feed the trained s.SCASHM with the simulated one-month dataset (without being decoded first) to learn and recognize various user traffics. The observation is focused on the number of packets of each user as a feature. The result shows that the proposed s.SCASHM successfully recognizes all top 8 users' traffics in the dataset. Fig. 7 illustrates experiment result of a window time observation.

Experiment 2*—Deep learning user traffic flow prediction vs. the memory prediction framework model*

The aim of this experiment was to compare the ability of the proposed s.SCASHM and the deep-learning technique to predict traffic flow. The result showed that the proposed s.SCASHM is able to maintain consistent prediction accuracy even during the fluctuating period (from 9 am to 5 pm) of daily activity. The s.SCASHM has the ability to learn on-the-fly during the observation,

while deep learning has lower prediction accuracy due to a rather small dataset during training. Fig. 8 shows the result for one-day time window.

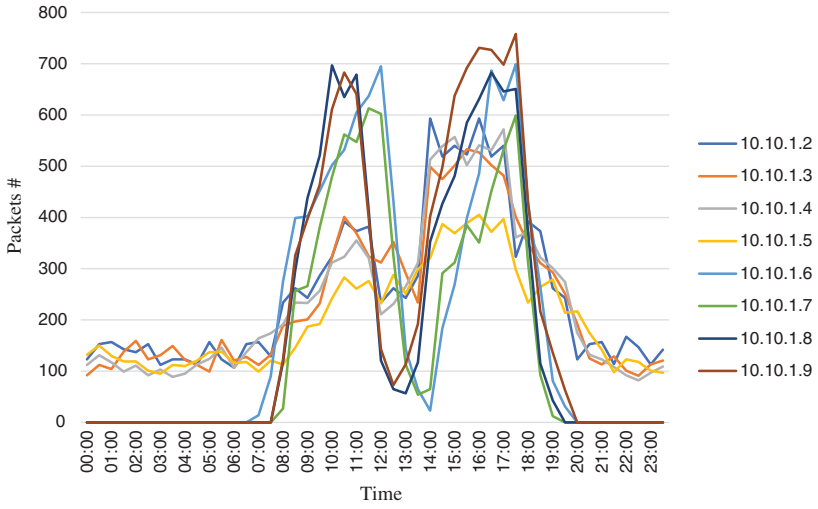

Figure 7: Automatic user traffic classification result

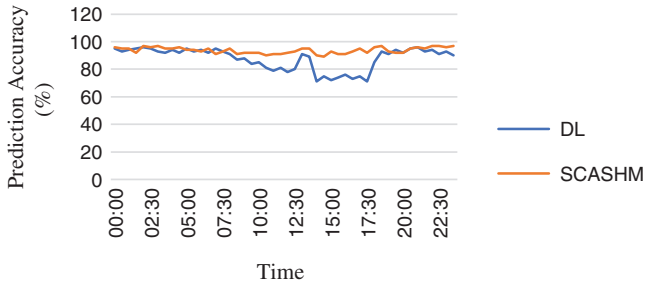

Figure 8: Prediction accuracy compared to the deep-learning technique

Fig. 9 shows the prediction matching statistics. The chart shows the prediction matching statistics of the proposed memory prediction model during the automatic user traffic classification experiments. A node with an IP address of 10.10.1.2 has the lowest matching percentage due to its inactive participation during the experiments.

Experiment 3*—Automatic user application traffic profiling with memory prediction*

The purpose of this experiment was to evaluate the performance of the proposed memory prediction model to profile users' application usage behaviors (as part of the UBA) by observing the number of packets exchanged as a feature. The outcome of this profiling is useful to identify any anomaly in the use of any services/applications by legacy users. The results shown in Fig. 10 are filtered to display only the profiles of four users/nodes. Among the applications/services, the email application records the highest number of packets for each user.

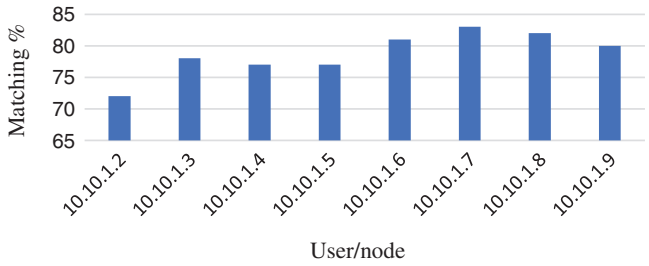

Figure 9: Statistics of the prediction matching of the proposed memory model

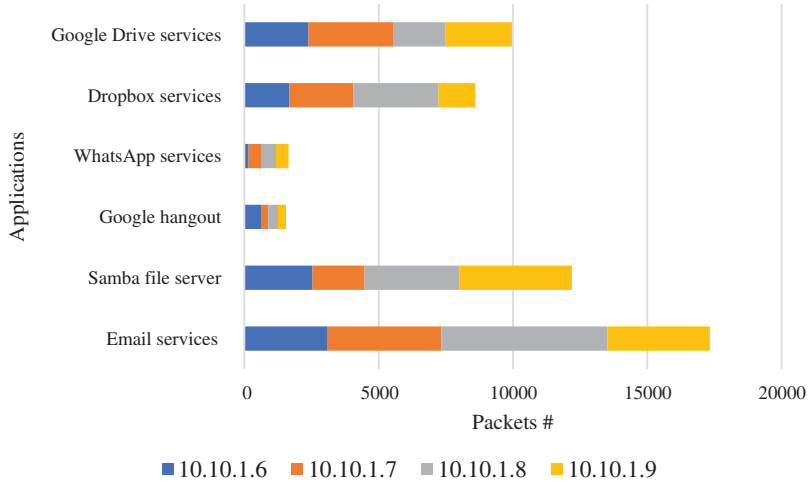

Figure 10: Profiling results of four users/nodes

Experiment 4*—Google Drive access anomaly detection*

In this experiment, the performance of the proposed memory prediction model was compared to the deep-learning model in detecting anomalous Google Drive access. The deep-learning technique uses decoded data for this purpose and relies on the Google Drive packet decoder to recognize Google Drive traffic. In this experiment, one peak-day traffic flow was randomly chosen and observed from 12.00 noon to 00.00 midnight. The true positive (TP), true negative (TN), False Positive (FP), and False Negative (FN) of the detections of the simulated anomalies were recorded, and the results are shown in Tab. 4. Tab. 5 shows the detection performances calculated from the observation results given in Tab. 4. The measurement results in Tab. 5 show that the proposed s.SCASHM outperforms the deep-learning model. Fig. 11 depicts the accuracy measurement of deep learning and the proposed s.SCASHM.

Experiment 5*—Overall traffic suspicious anomalous activity detection*

In this experiment, the performance of the proposed s.SCASHM in detecting anomalous activity in overall traffic at the gateway was compared to the deep-learning model. Observing overall traffic at the gateway is important as an initial indication of whether there is an anomaly/attack. If there was an indication of an anomaly, we continue with a detailed observation of individual traffic flows. The observation was done for a period of two weeks, and the results are shown in Fig. 12. The proposed s.SCASHM detects more anomalies compared to the deep-learning model for each observed day.

Table 4: Observation results on Google Drive access anomaly detection (DL *vs.* s.SCASHM)

Observation window time	TP		TN		FP		FN		Total	
	s.SCA-SHM	DL	s.SCA-SHM	DL	s.SCA-SHM	DL	s.SCA-SHM	DL	s.SCA-SHM	DL
12:00–14:00	491	429	119	113	83	137	7	21	700	700
14:00–16:00	591	472	32	39	78	160	9	39	710	710
16:00–18:00	526	392	51	57	37	143	7	29	621	621
18:00–20:00	617	513	58	53	46	134	6	27	727	727
20:00–22:00	647	541	77	85	53	134	5	22	782	782
22:00–24:00	594	491	81	90	53	131	4	20	732	732

Table 5: Performance metrics measurement on Google Drive access anomaly detection (DL *vs.* s.SCASHM)

Observation window time	Accuracy (%)		Precision (%)		Sensitivity (%)		F1-score (%)		Specificity	
	s.SCA-SHM	DL	s.SCA-SHM	DL	s.SCA-SHM	DL	s.SCA-SHM	DL	s.SCA-SHM	DL
12:00–14:00	87.14	77.42	85.54	75.79	98.59	95.33	91.60	84.44	58.91	45.2
14:00–16:00	87.74	71.97	88.34	74.68	98.50	92.36	93.14	82.58	29.00	19.59
16:00–18:00	92.91	72.30	93.42	73.27	98.68	93.11	95.98	82.00	57.59	28.50
18:00–20:00	92.84	77.85	93.06	79.28	99.03	95.00	95.95	86.43	55.76	28.34
20:00–22:00	92.58	80.05	92.42	80.14	99.23	96.09	95.71	87.39	59.23	38.8
22:00–24:00	92.21	79.37	91.80	78.93	99.33	96.08	95.42	86.67	60.44	40.72

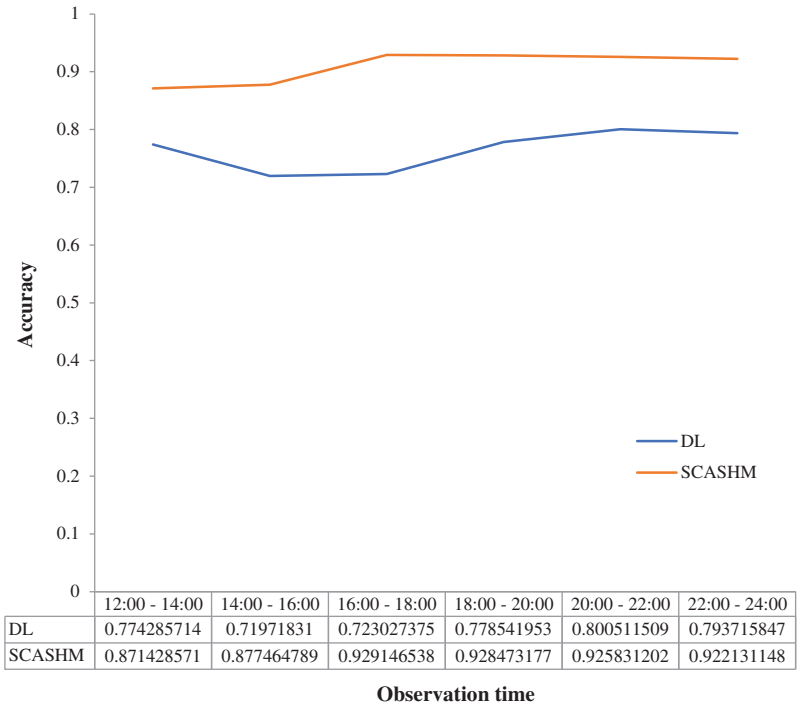

Figure 11: s.SCASHM *vs.* DL on Google Drive access anomaly detection accuracy

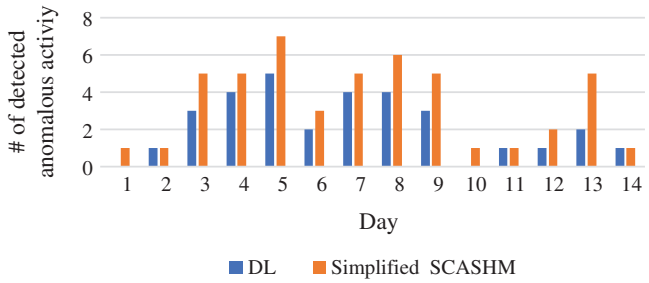

Figure 12: Anomalous activity detection comparison for overall traffic

Experiment 6—*Application-based anomalous traffic detection*

The aim of this experiment was to evaluate the proposed s.SCASHM's performance on other UBA functions, i.e., user-based anomaly detection. Again, the performance was compared to the deep-learning technique. The results shown in Tab. 6 indicate that the deep-learning technique is not able to detect a low-traffic simulated attack due to the small size of the training dataset.

Table 6: Anomalous activity detection comparison for overall traffic

NodeID	Application	# of detected anomaly		NodeID	Application	# of detected anomaly	
		DL	s.SCASHM			DL	s.SCASHM
10.10.1.6	Email services	0	0	10.10.1.8	Email services	0	1
	Samba file server	0	0		Samba file server	0	2
	Google hangout	0	0		Google hangout	0	0
	WhatsApp services	0	0		WhatsApp services	0	0
	Dropbox services	0	0		Dropbox services	0	0
	Google Drive services	0	0		Google Drive services	0	2
10.10.1.7	Email services	10	14	10.10.1.9	Email services	9	10
	Samba file server	23	31		Samba file server	14	17
	Google hangout	7	9		Google hangout	3	4
	WhatsApp services	11	11		WhatsApp services	4	4
	Dropbox services	18	22		Dropbox services	12	13
	Google Drive services	26	34		Google Drive services	21	25

Tab. 7 shows the cell firing activity statistics. It shows the backend statistics of the proposed s.SCASHM cells' activity during each experiment.

Table 7: Statistics (in %) on cell firing activity during the testing experiments

	Experiment #					
	1	2	3	4	5	6
Active cells (%)	85	45	78	81	53	72
Inactive cells (%)	12	35	15	10	38	21
Inhibited cells (%)	3	20	7	9	9	7

Finally, Tab. 8 shows the run-time for the eight nodes' profiling and detecting, i.e., the total time (in seconds) consumed by updating the models and generating the predictions.

Table 8: Running time for profiling and detecting (in seconds)

	Node ID							
	10.10.1.2	10.10.1.3	10.10.1.4	10.10.1.5	10.10.1.6	10.10.1.7	10.10.1.8	10.10.1.9
	Prof. Det.	Prof. Det.	Prof. Det.	Prof. Det.	Prof. Det.	Prof. Det.	Prof. Det.	Prof. Det.
s.SCASHM	0.12 0.01	0.14 0.01	0.14 0.01	0.15 0.01	0.27 0.01	0.44 0.03	0.51 0.06	0.49 0.05
DL	N/A 0.03	N/A 0.04	N/A 0.04	N/A 0.04	N/A 0.06	N/A 0.08	N/A 0.14	N/A 0.11

Note: Prof. = Profiling, Det. = Detection.

4.2 Discussion

A practical performance metric for tasks requiring an immediate prediction just after a few training examples is the online accuracy [35]. As depicted in Fig. 8, the proposed simplified s.SCASHM that uses the memory prediction framework outperforms the deep learning-based technique because it uses streaming data, no training is needed, and the time in which patterns are streamed is very important. Deep learning uses static data, requires training, and is time-invariant. Thus, deep learning involves large training sets, susceptibility to noise, long training times, complex setup, inability to adapt to changing data, and time invariance. The memory prediction framework learns as it goes and adapts to changing data. It may not be applicable to a broad type of problems that deep learning can solve, however the memory prediction framework excels in anomaly detection in streaming data, predicting data, and accurate semantic searches. Referring to Fig. 9, the worse performance on accuracy for the proposed model was observed at 72%.

With regards to learning convergence speed, the proposed memory model is absolutely better than the deep-learning model as it does not need to update multiple cells across the set of activated cells. It can be observed in Tab. 7 that few new cells are created when new information is inputted. In addition, Tab. 8 shows that the node's profiling and detecting time of s.SCASHM is better than the deep-learning model.

While both are inspired by the memory prediction framework, the main difference between Hierarchical Temporal Memory (HTM) [26–29] and the proposed memory model in this paper is the use of s.SCASHM instead of HTM's complex multicell set of a mini-column sequence memory. The proposed model has the advantage of a simpler and faster execution for forecasting data without the need to update multiple cells across the set of activated mini-columns, as is needed for HTM. In addition, the proposed model is not meant for learning long-term dependencies from high order sequences because such features require heavy processing time in the HTM model, which is not required for the application's purposes. Nevertheless, the proposed model is also designed to be scalable in case there is a need to implement a multi-cell architecture to increase the model's ability to recognize complex objects. Considering the running time as well as the complexity of the model, the proposed s.SCASHM has the potential to be adopted as a real-time user behavior analytics model to prevent insider attacks.

4.3 Threats to Validity

Shaukat et al. [36] examined threat validity in the use of machine-learning techniques. The authors compared the performance of supervised as well as unsupervised machine-learning techniques running on different publicly available datasets. To discuss the validity threats of the proposed s.SCASHM on detecting anomalies, first, we considered the threat to internal validity as the selection of the dataset. It also involves a risk related to the knowledge and understanding of the underlying network where the dataset is collected. To mitigate impacts from this internal threat, the simulated traffic was produced by imitating the actual traffic captured from our university network, which we are familiar with. We also simulated anomalous behaviors into the traffic using knowledge gained from the literature.

Second, we used the external validity threat as a measure to evaluate the results from our experiments. We used a simulated dataset in our experiments due to the inability to inject anomalous traffic into our real university network. This limited our options in comparing the results with other approaches using real network traffic. We addressed this issue by also replicating similar experiments using the deep-learning model for comparison.

4.4 Limitations and Future Directions

There is no machine-learning technique that has no limitations in terms of data or processes (e.g., overfitting [36,37]). Based on our comparative experiments, we also identified one possible problem caused by the limitation of the proposed s.SCASHM as an unsupervised, lifelong, continuous learning approach. The problem lies in the fact that the s.SCASHM needs to learn from considerably normal user behaviors and thus cannot be deployed as a diagnostic tool into an already problematic network. This limitation also opens up the possibility for insider attackers to slowly change their traffic behaviors to avoid triggering suspicious alerts. This limitation is known as unsupervised learning from unknowingly anomalous traffic.

Real-time sequence learning from data streams presents unique challenges for machine learning algorithms. Besides prediction accuracy, the development of the following are considered as future research directions.

(1) Ideal algorithm that can learn Markov order automatically and efficiently to make accurate high-order predictions. Real-world sequences contain contextual dependencies that span multiple time steps, i.e.,: the ability to make Markov high-order predictions.
(2) Algorithm that can recognize and learn new patterns rapidly to perform continuous learning for real-time data stream analysis.
(3) Proper sequence learning algorithm that can make multiple predictions simultaneously and evaluate the likelihood of each prediction online, because a given temporal context, there could be multiple possible future outcomes.
(4) Ultimate algorithm that possesses acceptable performance on various problems without any task-specific hyper-parameter tuning. Learning in the cortex is enormously robust for a wide range of tasks. In contrary, most machine learning algorithms require optimizing a set of hyper-parameters for various tasks. Hyper-parameter tuning presents a major challenge for applications that require a high degree of automation, like data stream mining.

5 Conclusions

In this paper, the authors have introduced a new model of a sequential hierarchical superset of a memory-prediction framework. The model was used as an unsupervised learning engine

for a real-time user behavior anomaly detection system. The experimental results show that the s.SCASHM performed well in recognizing spatial as well as temporal anomalies in the network security domain. It was able to detect user traffic behavior with an accuracy ranging from 72% to 83%. Thus, the detection system meets the requirements of real-time, continuous, online detection without supervision.

As a memory prediction model, the proposed s.SCASHM has an advantage compared to the HTM model in terms of a simpler and faster execution for forecasting data without the need to update multiple cells across the set of activated mini-columns. Compared to conventional machine-learning and deep-learning models, the proposed s.SCASHM model has an advantage in terms of its capability to perform on-the-fly learning.

Researchers in neuroscience and artificial intelligence (AI) are now focusing on understanding how the brain works and thus on predicting behavior. When we can comprehend better the brain, we may produce better designs for AI algorithms. For the learning process, the neuroscience-AI collaboration can be developed further, and a precise learning process developed by neuroscience can be used to inform the design of the process for AI. Correspondingly, if AI discovers patterns from large datasets and creates a learning model, neuroscientists could carry out wet experiments to confirm the model. This research work materializes the idea of a partnership between both disciplines.

Considering the limitations and future directions discussed in Section 4.4, for future work, the authors plan to focus on three aims. First, we plan to extend the data representation model and improve the proposed s.SCASHM architecture further by implementing an advanced cortical micro-circuitry region that enhances the connection between the hierarchical layers and also allows for storing more data. An empirical study is planned to validate the proposed s.SCASHM model by utilizing functional magnetic resonance imaging (fMRI) to identify cells' activities when dealing with an episodic and declarative memory within the neocortex's cortical column. The goal is to measure the growth rate of the s.SCASHM model when presented with new information compared to the actual cell growth in the neocortex and also to investigate specific properties of the human neocortex that have not been incorporated into the proposed model, e.g., time. Second, we plan to experiment with a larger organizational network and evaluate the performance of the proposed detection system by performing more user behavioral analytic functionalities. Next, we will address the limitation of learning from unknowingly bad traffic by incorporating a feedback mechanism that allows the s.SCASHM to decide whether to discard the current newly learned data or to adjust the previously learned data.

Funding Statement: This research was funded by Scientific Research Deanship, Albaha University, under the Grant Number: [24/1440].

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] T. M. Alam, K. Shaukat, I. A. Hameed, S. Luo, M. U. Sarwar *et al.,* "An investigation of credit card default prediction in the imbalanced datasets," *IEEE Access*, vol. 8, pp. 201173–201198, 2020.

[2] I. Javed, X. Tang, K. Shaukat, M. U. Sarwar, T. M. Alam *et al.,* "V2X-based mobile localization in 3D wireless sensor network," *Security and Communication Networks*, vol. 2021, pp. 1–13, 2021.

[3] M. U. Hassan, M. Shahzaib, K. Shaukat, S. N. Hussain, M. Mubashir *et al.,* "DEAR-2: An energy-aware routing protocol with guaranteed delivery in wireless ad-hoc networks," in *Recent Trends and*

Advances in Wireless and IoT-enabled Networks. EAI/Springer Innovations in Communication and Computing, M. Jan, F. Khan and M. Alam (Eds.), Cham: Springer, pp. 215–224, 2019.

[4] K. Shaukat and M. U. Hassan, "Cloud security through encryption techniques," *Transylvanian Review*, vol. 25, no. 15, pp. 4037–4042, 2017.

[5] K. Shaukat, A. Rubab, I. Shehzadi and R. Iqbal, "A socio-technological analysis of cyber crime and cyber security in Pakistan," *Transylvanian Review*, vol. 25, no. 16, pp. 4187–4190, 2017.

[6] K. Shaukat, F. Iqbal, T. M. Alam, G. K. Aujla, L. Devnath *et al.,* "The impact of artificial intelligence and robotics on the future employment opportunities," *Trends in Computer Science and Information Technology*, vol. 5, no. 1, pp. 050–054, 2020.

[7] D. Stiawan, A. H. Abdullah and M. Y. Idris, "Classification of habitual activities in behavior-based network detection," *Journal of Computing*, vol. 2, no. 8, pp. 1–7, 2010.

[8] Z. Sun, Y. Wang, H. Zhou, J. Jiao and R. E. Overstreet, "Travel behaviours, user characteristics, and social-economic impacts of shared transportation: A comprehensive review," *International Journal of Logistics Research and Applications*, vol. 24, no. 1, pp. 51–78, 2019.

[9] H. Zhang, M. Wang, L. Yang and H. Zhu, "A novel user behavior analysis and prediction algorithm based on mobile social environment," *Wireless Network*, vol. 25, no. 2, pp. 791–803, 2019.

[10] K. Perichappan, "Greedy algorithm based deep learning strategy for user behavior prediction and decision making support," *Journal of Computer and Communications*, vol. 6, no. 6, pp. 45–53, 2018.

[11] K. Deng, L. Xing, L. Zheng, H. Wu, P. Xie *et al.,* "A user identification algorithm based on user behavior analysis in social networks," *IEEE Access*, vol. 7, pp. 47114–47123, 2019.

[12] E. B. Karbab, V. Debbabi, A. Derhab and D. Mouheb, "MalDozer: Automatic framework for android malware detection using deep learning," *Digital Investigation*, vol. 24, no. 4, pp. S48–S59, 2018.

[13] K. Shaukat, S. Luo, S. Chen and D. Liu, "Cyber threat detection using machine learning techniques: A performance evaluation perspective," in *Proc. of 2020 Int. Conf. on Cyber Warfare and Security*, Islamabad, Pakistan, pp. 1–6, 2020.

[14] K. Shaukat, T. M. Alam, S. Luo, S. Shabbir, I. A. Hameed *et al.,* "A Review of time-series anomaly detection techniques: A step to future perspectives," in *Advances in Information and Communication (FICC 2021). Advances in Intelligent Systems and Computing*, K. Arai (Ed.), vol. 1363. Cham: Springer, pp. 865–877, 2021.

[15] J. Hawkins and S. Blakeslee, *On Intelligence*. New York, USA: Owl Book, 2005.

[16] H. Eichenbaum, "Memory systems," *WIREs Cognitive Science*, vol. 1, no. 4, pp. 478–490, 2010.

[17] R. N. Henson and P. Gagnepain, "Predictive, interactive multiple memory systems," *Hippocampus*, vol. 20, no. 11, pp. 1315–1326, 2010.

[18] F. De Brigard, "Predictive memory and the surprising gap," *Frontiers in Psychology*, vol. 3, no. 420, pp. 23162493, 2012.

[19] J. Hawkins and S. Ahmad, "Why neurons have thousands of synapses, a theory of sequence memory in neocortex?," *Frontiers in Neural Circuits*, vol. 10, no. 177, pp. 23, 2016.

[20] J. Hawkins, M. Lewis, M. Klukas, S. Purdy and S. Ahmad, "A framework for intelligence and cortical function based on cells in the neocortex," *Frontiers in Neural Circuits*, vol. 12, pp. 121, 2019.

[21] S. Ahmad and J. Hawkins, "How do neurons operate on sparse distributed representations? A mathematical theory of sparsity, neurons and active dendrites," ArXiv, abs/1601.00720, 2016.

[22] I. Herrero-Reder, J. M. Peula, C. Urdiales and F. Sondoval, "CBR based reactive behavior learning for the memory-prediction framework," *Neurocomputing*, vol. 250, no. 1–3, pp. 18–27, 2017.

[23] S. J. Garalevicius, "Memory-prediction framework for pattern recognition: Performance and suitability of the Bayesian model of visual cortex," in *Proc. of the Twentieth Int. Florida Artificial Intelligence Research Society Conf.*, Florida, USA, pp. 92–97, 2007.

[24] M. Bundzel and S. Hashimoto, "Object identification in dynamic images based on the memory-prediction theory of brain function," *Journal of Intelligent Learning Systems and Applications*, vol. 2, no. 4, pp. 212–220, 2010.

[25] T. Pham, T. Tran, D. Phung and S. Venkatesh, "DeepCare: A deep dynamic memory model for predictive medicine," in *Advances in Knowledge Discovery and Data Mining. PAKDD 2016. Lecture Notes in Computer Science*, J. Bailey, L. Khan, T. Washio, G. Dobbie, J. Huang, et al. (Eds.), vol. 9652. Cham: Springer, pp. 30–41, 2016.

[26] Y. Cui, S. Ahmad and J. Hawkins, "The HTM spatial pooler—A neocortical algorithm for online sparse distributed coding," *Frontiers in Computational Neuroscience*, vol. 11, pp. 134, 2017.

[27] Y. Cui, S. Ahmad and J. Hawkins, "Continuous online sequence learning with an unsupervised neural network model," *Neural Computation*, vol. 28, no. 11, pp. 2474–2504, 2016.

[28] S. Ahmad, A. Lavin, S. Purdy and Z. Agha, "Unsupervised real-time anomaly detection for streaming data," *Neurocomputing*, vol. 262, pp. 134–147, 2017.

[29] Lavin and S. Ahmad, "Evaluating real-time anomaly detection algorithms—The Numenta anomaly benchmark," in *Proc. of IEEE 14th Int. Conf. on Machine Learning and Applications*, Miami, FL, USA, pp. 38–44, 2015.

[30] G. Li, Y. Shen, P. Zhao, X. Lu, J. Liu *et al.,* "Detecting cyberattacks in industrial control systems using online learning algorithms," *Neurocomputing*, vol. 364, no. 82, pp. 338–348, 2019.

[31] S. Mohamad and A. Bouchachia, "Deep online hierarchical dynamic unsupervised learning for pattern mining from utility usage data," *Neurocomputing*, vol. 390, no. 3, pp. 359–373, 2020.

[32] M. F. Pasha, R. Budiarto, S. Ramadass and M. Syukur, "A sequential hierarchical superset implementation of neocortex memory system and its case study of automated network forensic analysis," in *Proc. of Int. Conf. on Artificial Intelligence*, Las Vegas, USA, pp. 490–495, 2018.

[33] MACCDC 2012 dataset, 2021. [Online]. Available: https://maccdc.org/2012-agenda/.

[34] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint 1409.1556, 2018.

[35] V. Losing, B. Hammer and H. Wersing, "Incremental on-line learning: A review and comparison of state of the art algorithms," *Neurocomputing*, vol. 275, no. 2, pp. 1261–1274, 2018.

[36] K. Shaukat, S. Luo, V. Varadharajan, I. A. Hameed, S. Chen *et al.,* "Performance comparison and current challenges of using machine learning techniques in cybersecurity," *Energies MDPI*, vol. 13, no. 10, pp. 2509, 2020.

[37] K. Shaukat, S. Luo, V. Varadharajan, I. A. Hameed and M. Xu, "A Survey on machine learning techniques for cyber security in the last decade," *IEEE Access*, vol. 8, pp. 222310–222354, 2020.