



COREM² project: a beginning to end approach for cyber intrusion detection

Anastasios Panagiotis Psathas¹ · Lazaros Iliadis¹ · Antonios Papaleonidas¹ · Dimitris Bountas¹

Received: 2 December 2021 / Accepted: 14 February 2022 / Published online: 14 March 2022
© The Author(s), under exclusive licence to Springer-Verlag London Ltd., part of Springer Nature 2022

Abstract

The growing need to use online services has made it necessary to ensure protection against all kinds of cyber-threats. This research effort aims to tackle network security problems as follows: It introduces the hybrid intrusion detection system COREM² that successfully detects nine cyber-attacks. Its architecture comprises of a two-dimensional *convolutional neural network* (2-D CNN), a *recurrent neural network* with long short-term memory layers and a *multilayer perceptron*. The COREM² was successfully tested against the timely *Kitsune Network Attack Dataset*, achieving an overall accuracy of 98.64% and 98.92% in the training and testing phases, respectively. Since this is a multiclass classification effort, the “*one-versus-all strategy*” was employed to validate the introduced model, which has proved its ability to generalize. COREM² outperforms other state-of-the-art approaches achieving overall accuracy above 98%, rare for field cyber-security intrusion. We strongly suggest that it can be safely used as a prototype for further research on network security enhancement. Furthermore, this research introduces a holistic approach for cyber intrusion detection, using the COREM² in order to classify network traffic as benign or malicious. It captures network flow packets in the form of PCAP files (packet capture), and it stores them in.csv files and it evaluates them in order to perform classification in ten classes as provided by the Kitsune Dataset. If the malicious traffic exceeds a certain limit, the model notifies the user to take all necessary actions. The proposed method has an average processing power of 10,000 packets per 8 s. It potentially can be used in any device that has Internet access.

Keywords CNN · RNN · LSTM · MLP · Hybrid · IDS · Cyber-security · Cyber-attacks · PCAP

1 Introduction

The recent rapid technological developments have led to a wide employment of computer networks and online services, on a global scale. The popularity of Internet continues to grow rapidly urging security to evolve just as quickly to protect the electronic microcosm, while keeping it functional. This has resulted to a simultaneous increase of cyber-attacks on the interconnected systems [1]. Chronic study and experience in the field of security, unfortunately, are not a given that ensures the desired result. Intermittent security threats and vulnerabilities appear every day. This calls for a continuous effort of prevention and detection of risks. Nowadays, the development and employment of machine learning (ML) systems capable of detecting network intrusion are imperative.

✉ Anastasios Panagiotis Psathas
anpsatha@civil.duth.gr

Lazaros Iliadis
liliadis@civil.duth.gr

Antonios Papaleonidas
papaleon@civil.duth.gr

Dimitris Bountas
dibounta@civil.duth.gr

¹ Department of Civil Engineering-Lab of Mathematics and Informatics (ISCE), Democritus University of Thrace, 67100 Xanthi, Greece

Critical national infrastructures (CNIs) such as ports, water and gas distributors, hospitals, energy providers are among the main targets of cyber-attacks. Supervisory control and data acquisitions (SCADA) or industrial control systems (ICS) in general are the core systems that CNIs rely on in order to manage their production. Protection of ICSs and CNIs has become an essential issue to be considered in an organizational, national and European level. For instance, in order to cope with the increasing risk of CNIs, Europe has recently issued a number of directives and regulations that try to create a coherent framework for securing networks, information and electronic communications. Apart from regulations, directives and policies, specific security measures are also needed to cover all legal, organizational, capacity building and technical aspects of cyber-security [2]. Intrusion detection systems (IDSs) [3] represent a vital technology for all types of networks. IDSs are part of the second defense line of a system. They can be deployed along with other security measures, such as access control, authentication mechanisms and encryption techniques in order to better secure the systems against potential attacks. They learn to preemptively stamp out, normal and malicious network traffic flow, by considering patterns relative to specific potential cyber- threats [4].

In 2020, the average cost of a data breach was 3.86 million \$ globally and 8.64 million in the USA [5]. These costs include the expenses of discovering and responding to the breach, the cost of downtime and lost revenue and the long-term reputational damage to a business and its brand. Cybercriminals target customers' personally identifiable information (PII)—names, addresses, national identification numbers (e.g., Social Security numbers in the USA, fiscal codes in Italy) and credit card information. These stolen records are sold in underground digital marketplaces. Compromised PII often leads to loss of customer trust [6], regulatory fines and even legal actions. Security system complexity, created by disparate technologies and a lack of in-house expertise, can amplify these costs. Organizations with a comprehensive cyber-security strategy, governed by best practices, using advanced analytics, artificial intelligence (AI) and machine learning, can fight cyber-threats more effectively, and they can reduce the lifecycle and impact of potential breaches [7].

This paper constitutes a major extension of a previous research effort of our team [8], in which we have proposed the prototype of a novel hybrid IDS. The enhanced deep learning architecture described herein offers a significant contribution to the literature, by introducing an improved modeling result. More specifically, a more robust hyper-parameters' tuning process has been followed. On the other hand, testing was performed based on a tenfold cross-validation process [9] (instead of using the holdout method)

[2]. Regarding the system's architecture, a bunch of dropout layers have been added after the flattened layer of the 2-D CNN and after the second layer of the LSTM-RNN, in order for the algorithm to become bulletproof to overfitting. Further improvement includes the introduction of a "beginning to end" approach for cyber intrusion detection on a computer (or another device that is connected to the Internet). The novel approach described herein captures the packets of the net flow in the form of packet capture files, it stores them, and it classifies them either in one of the 9 cyber-attacks provided by the *Kitsune* Dataset or to the normal class.

The rest of the paper is organized as follows. Section 2 presents an extended reference to the existing literature review. Section 3 describes the dataset and its features. Section 4 provides the architecture of the proposed model. Section 5 presents the experimental results and the evaluation of the model. Section 6 analyzes the way that the capturing, storing and evaluation of the packets of the net flow is applied. Finally, Sect. 7 concludes the research and highlights the key points of the research.

2 Literature review

A vast amount of machine learning approaches have been introduced for cyber intrusion (CI) detection, during the last twenty years. Recently, several *deep learning* (DL) models have been introduced in the literature, aiming to detect malware, to classify network intrusions and phishing (spam attacks) and to inspect website defacements. Following the rapid spread of Bitcoin [10], the technology of blockchain [11] came to the fore and provided new tools for proposing cyber intrusion detection approaches. In this section, a detailed overview of timely approaches toward tackling cyber vulnerabilities is presented.

In 2011, Damopoulos et al. developed four different ML classifiers (*Bayesian networks*, *radial basis function* (RBF), *k-nearest neighbors* (k-NN) and *random forest* (RF)) achieving an accuracy of 98.9% for anomaly detection on mobile devices [12]. In 2012, Liet et al. introduced a combination of the *k-means* clustering, with the *ant colony* and support vector machines (SVM), detecting a Denial of Service (DoS), a Remote to Local (R2L), a User to Root (U2R) and a *Probe Attack* with overall accuracy as high as 98.62% [13]. Elekar, used decision trees (J48) with RF, J48 with random trees (RT) and RT with RF on the KDDcup99 and NSLKDD datasets, to recognize the latter cyber-attacks with accuracy of 92.62% [14]. Ganeshkumar and Pandesswari [15] developed respective models toward the detection of the same cyber-attacks. They introduced and successfully applied a hybrid fuzzy system integrated with neural networks, on the KDD Cup 1999 database.

Specifically, the accuracy for the DoS, Probe, R2L, U2R attacks was equal to 78.61%, 95.52%, 85.30% and 78.61%, respectively. Soe et al. [16] applied *artificial neural networks* (ANNs), the J48 *decision tree* algorithm and the *naïve Bayes* [17] to successfully detect the attacks of Mirai and Bashlite Botnets, two of the most dangerous Internet of Thing (IoT) malware and reached a high accuracy of 99%. Zhang et al., 2005 proposed a hierarchical method, where each packet first passes through an anomaly detection model. Then if an anomaly is raised, the packet is evaluated by a set of ANN classifiers where each classifier is trained to detect a specific kind of attack [18]. Last but not least, in 2017 [19], Dash proposed hybrid IDS, in which the gravitational search (GS) and a collaboration of the GS and particle swarm optimization (GSPSO) techniques were used to train a neural network. Subsequently, the GS-ANN and GSPSO-ANN models were used for the intrusion detection process. To evaluate the performance of their method, the author's approaches were compared with other optimization algorithms, such as the genetic algorithm (GA), PSO and an ANN based on gradient descent (GD-ANN). The author claimed that this approach is more suitable for unbalanced datasets. Using the NSLKDD dataset, the presented method achieved 94.90% and 98.13% accuracy using the GS and GSPSO, respectively. In 2020, Demertzis et al. introduced Gryphon, a Semi-Supervised Unary Anomaly Detection System for big industrial data which is employing an evolving spiking neural network (eSNN) one-class classifier (eSNN-OCC). This machine learning algorithm corresponds to a model capable of detecting very fast and efficiently, divergent behaviors and abnormalities associated with cyber-attacks [20]. Similar studies achieved suchlike results in [21–23].

Kolosnjaji et al. [24] used CNNs and RNNs, for the classification of malware “*system call*” sequences. The accuracy of the model was as high as 89.4%, the precision was equal to 85.6% and the recall was 89.4% [24]. In 2016, Pascanu et al. developed a hybrid malware detection method using RNNs combined with a MLP and *logistic regression*, achieving a *true positive rate* (TPR) of 98.3% and a *false positive rate* (FPR) of 0.1% [25]. In 2017, Mizano et al. identified procedures of malicious software using the HTTP headers of network traffic, with a precision equal to 97.1%, whereas the FPR was only 1%. These results were achieved using a *neural network* with two hidden layers [26]. In 2019, Demertzis et al. suggested an active security strategy that adopts a vigorous method including ingenuity, data analysis, processing and decision-making support to face various cyber hazards. This research introduced a novel intelligence driven cognitive computing (security operation center) SOC that is based exclusively on progressive fully automatic procedures. The proposed λ -architecture network flow forensics framework

(λ -NF3) is an efficient cyber-security defense framework against adversarial attacks. It implements the Lambda machine learning architecture that can analyze a mixture of batch and streaming data, using two accurate novel computational intelligence algorithms [27]. In 2018, Cordosky et al. performed classification using a deep neural network (DNN) with nine layers, with batch normalization and dropout between layers, achieving 97% accuracy on classifying malware families, using features derived from static and dynamic analysis [28]. In 2016, Gibert Llauradó, used a 2-D CNN and a 1-D CNN achieving accuracy of 99.5% [29]. Loukas et al., in 2017, proposed a cyber-physical IDS. The system used both RNN and Deep MLP (DMLP) with an average accuracy of 86.9% [30]. In 2019, Thamilarasu et al. developed a *deep belief network* (DBN) to fabricate the feed-forward ANN for the IoT (Internet of Things). The proposed model was tested against 5 cyber-attacks, namely Sinkhole, Wormhole, Blackhole, Opportunistic Service and Distributed Denial of Service (DDoS). The results have shown a higher precision of 96% and a recall rate of 98.7% for detecting DDoS attacks [31]. Finally, in 2018, Shone et al. deployed a *deep* auto-encoder for cyber intrusion detection [32]. In this study, the authors used the Knowledge Discovery and Dissemination (KDD'1999) dataset [33] and the NSLKDD one (a data set suggested to solve some of the inherent problems of the KDD'99) [34] and they reached an average accuracy as high as 97.85%. In more recent research, in 2021, Psathas et al. presented a hybrid intrusion detecting system (IDS) comprising of a two-dimensional *convolutional neural network* (2-D CNN), a RNN and a MLP for the detection of nine cyber-attacks versus normal flow. The timely *Kitsune Network Attack Dataset* was used in this research. The proposed model achieved an overall accuracy of 92.66%, 90.64% and 90.56% in the train, validation and testing phases, respectively [8]. Similar studies achieved suchlike results in [35–37].

In 2008, Nakamoto, presented the blockchain into a peer-to-peer electronic cash system. Transactions were verified and linked each other in an open distributed ledger making almost impossible for someone to tamper the information of any block [38]. In 2019, Serrano proposed a blockchain random neural network (BRNN), which has been applied to an IoT AAA server that covers the digital seven layers of the OSI model and the physical user credentials such as passport or biometrics [39]. The result was a holistic physical and digital cyber-security application in the IoT where access to the network in an area required prior user physical verification between decentralized parties. User data were encrypted, and information was decentralized where attackers could be identified if a criminal attack was delivered. In 2020, Giannoutakis et al. introduced a blockchain to support the cyber-security

mechanisms of smart homes installations, focusing on the immutability of users and devices that constitute such environments [40]. The proposed methodology provides also the appropriate smart contracts support for ensuring the integrity of the smart home gateway and IoT devices, as well as the dynamic and immutable management of blocked malicious IPs. In 2020, Demertzis et al. suggested a blockchain security architecture that aims to ensure network communication between traded industrial Internet of Things devices, following the Industry 4.0 standard and based on Deep Learning Smart Contracts. The proposed smart contracts are implementing (via computer programming) a bilateral traffic control agreement to detect anomalies based on a trained deep auto-encoder neural network [41]. Similar studies achieved suchlike results in [42–44]

3 Dataset

Finding comprehensive and valid datasets to be used in testing and evaluation is of vital importance [45]. Unfortunately, the area of cyber intrusion detection suffers from the lack of labeled datasets, with full access to the source files (most of the cases *.pcap files). The development of IDS using ML requires more or less balanced datasets, containing a sufficient number of benign and malicious data vectors. This is a true challenge, as the benign cases are always outnumbering the malicious. Furthermore, the labeling process, in most cases, is performed manually, by the experts. Thus, it is difficult and time-consuming to label all the packets in a network, given the fact that the number of packets per second (PPS) sent is extremely vast. One of the most widely used datasets for intrusion detection is the KDD 1999 [33]. It contains more than four million network traffic records, corresponding to 22 different attacks (DoS, U2R, R2L and Probe Attacks). As it has already been mentioned, the NSLKDD [34] is an improved version of the KDD 1999 and it is also widely used. The CTU-13 contains only raw packet data [46]. The UNSW-NB15 dataset [47] was created by four tools, namely IXIA PerfectStorm, Tcpcdump, Argus and Bro-IDS. These tools are used to create some types of attacks, including DoS, Exploits, Generic, Reconnaissance, Shellcode and Worms. The Bot-IoT [48] is a more recent one, related to the cyber-attacks of Mirai and Bashlite Botnets. There are several other public sets like the CSE-CIC-IDS2018 [49], the Tor-NonTor [50] and the Android Malware [51]. For this research, we have employed the recently published, Kitsune Network Attack Dataset [52].

3.1 Dataset description

The Kitsune Network Attack Dataset is publicly available, and it was developed by Yisroel Mirsky, Tomer Doitshman, Yuval Elovici and Asaf Shabtai of the Negev Ben-Gurion University, Department of Information Systems Engineering [52, 53]. It contains a vast number of network packets, 21,017,588 in total. The malicious traffic is related to nine different cyber-attacks (CYA) and comprises of 4,851,280 packets. The remaining 16,166,306 records correspond to normal flow. The reasons for choosing this data set are the following: First, the fact that it contains nine cases of CYA related to four classes. Another important attribute of this dataset is its large number of infected packets. It is also important that it is a relatively timely, up to date dataset. Last but not least, the developers of the dataset provide, apart from *.csv files that contain the feature extracted packets and the corresponding labels, the original *.pcap files, for further processing for anyone interested in investing time in this field. Table 1 shows the name and the type of each attack, its description, the total number of packets and the number of infected packets.

Each record of the dataset is a 1×115 -dimensional vector, i.e., each package is a vector consisting of 115 features. In general, each vector contains temporal statistics describing the packet's channel, the packet's sender and the traffic between the packet's sender and the receiver. Specifically, the statistics summarize all features of traffic:

- Originating from this packet's source MAC and IP address (denoted SrcMAC-IP).
- Originating from this packet's source IP (denoted SrcIP).
- Sent between this packet's source and destination IPs (denoted Channel).
- Sent between this packet's source and destination TCP/UDP Socket (denoted Socket).

A total of 23 features can be extracted from a single temporal window. The same set of features is extracted from a total of five temporal windows of approximately: 100 ms, 500 ms, 1.5 s, 10 s and 1 min, thus using 115 features. Not every packet applies to every channel type (e.g., there is no socket if the packet does not contain a TCP (transfer control protocol) or an UDP (user datagram protocol)). In these cases, these features are zeroed. Table 2 presents the feature of the packet being measured, the corresponding statistics and their description, from which elements of the package were calculated and the total

Table 1 Name, type, description, number of packets and number of infected packets for the nine (9) cyber-attacks considered in this research

Attack name	Attack type	Description: the attacker...	# Packets	# Infected packets
OS scan	Reconnaissance	...Scans the network for hosts and their operating systems, to reveal possible vulnerabilities	1,697,850	65,700
Fuzzing		... Searches for vulnerabilities in the camera’s web servers by sending random commands to their cgis	2,244,138	432,783
Video injection	Man in the middle	... Injects a recorded video clip into a live video stream	2,472,400	102,499
ARP MitM		... Intercepts all LAN traffic via an ARP poisoning attack	2,504,266	1,145,272
Active wiretap		... Intercepts all LAN traffic via active wiretap (network bridge) covertly installed on an exposed cable	2,278,688	923,216
SSDP flood	Denial of service	... Overloads the DVR by causing cameras to spam the server with UPnP advertisements	4,077,265	1,439,604
SYN DoS		... Disables a camera’s video stream by overloading its web server	2,771,275	7,038
SSL Renegotiation		... Disables a camera’s video stream by sending many SSL renegotiation packets to the camera	2,207,570	92,652
Mirai	Botnet malware	...Infects IoT with the mirai malware by exploiting default credentials and then scans for new vulnerable victims network	764,136	642,516

Table 2 23 features extracted from each packet for a single time window

Features	Statistics	Description of the statistics	Aggregated by	# Features
Packet’s size out	μ_{Si}, σ_{Si}	Bandwidth of the outbound traffic	SrcMAC-IP, SrcIP, Channel, Socket	8
Packet’s size in-out	$\ S_i, S_j\ , R_{Si,Sj}, Cov_{Si,Sj}, P_{Si,Sj}$	Bandwidth of the outbound and inbound traffic together	Channel, Socket	8
Packet’s count	w_i	Packet rate of the outbound traffic	SrcMAC-IP, SrcIP, Channel, Socket	4
Packet’s jitter	$w_i, \mu_{Si}, \sigma_{Si}$	Inter-packet delays of the outbound traffic	Channel	3

Table 3 Summary of the computed incremental statistics

Type	Statistic	Notation	Calculation
1D	Weight	w	w
	Mean	μ_{Si}	LS/w
	Std.	σ_{Si}	$\sqrt{ SS/w - (LS/w)^2 }$
2D	Magnitude	$\ S_i, S_j\ $	$\sqrt{\mu_{Si}^2 + \mu_{Sj}^2}$
	Radius	$R_{Si,Sj}$	$\sqrt{(\sigma_{Si}^2)^2 + (\sigma_{Sj}^2)^2}$
	Approx. covariance	$Cov_{Si,Sj}$	$\frac{SR_{ij}}{w_i + w_j}$
	Correlation coefficient	$P_{Si,Sj}$	$\frac{Cov_{Si,Sj}}{\sigma_{Si} + \sigma_{Sj}}$

number of features was obtained. Table 3 presents the method used for the calculation of each statistic and Table 4 labels the 10 classes of network’s flow (Nine Malicious and one Benign).

Here, $S = \{x_1, x_2, \dots, x_N\}$ is a sequence of observed packet sizes. The mean, variance and standard deviation of S can be updated incrementally by maintaining the tuple.

IS: = (N, LS, SS), N, LS and SS are the *number, linear sum* and *squared sum* of instances, respectively. Concretely, the update procedure for inserting x_i into IS is.

$IS \leftarrow (N + 1, LS + x_i, SS + x_i^2)$, and the statistics at any given time are $\mu_S = \frac{LS}{N}, \sigma_S^2 = \left| \frac{SS}{N} - \left(\frac{LS}{N}\right)^2 \right|$ and $\sigma_S = \sqrt{\sigma_S^2}$.

SR_{ij} is the sum of the residual products between streams i and j (used for computing 2D statistics). For further information about the feature extraction process, refer to [52].

3.2 Dataset preprocessing

For each attack, the dataset provides the following:

- (a) An *attack.csv* file, comprising of the features. Each vector is a network packet.

Table 4 10 classes of network's flow (Nine Malicious and one Benign)

Attack	Active wiretap	ARP MitM	Fuzzing	OS scan	SSDP flood	SSL Renegotiation	SYN DoS	Video injection	Mirai	Benign
Attack Label	1	2	3	4	5	6	7	8	9	0

SSDP Flood: Simple Service Discovery Protocol Flood attack: A reflection-based DDoS attack that exploits Universal Plug and Play networking protocols to send an amplified amount of traffic to a targeted victim, overwhelming the target's infrastructure and taking their web resource offline

SSL Renegotiation: Secure Socket Layer Renegotiation attack

SYN DoS: Synchronize Denial of Service attack

ARP MitM: Address Resolution Protocol, Man in the Middle attack

Active Wiretap: Attempts to alter data being communicated and affect data flow

Mirai: A malware that turns networked devices running Linux into remotely controlled "bots" that can be used as part of a botnet in large-scale network attacks

- (b) The corresponding labels' file *labels.csv* [benign, malicious].
- (c) The original network capture in truncated *.pcap format.

Data handling has been achieved by writing code from scratch in Python. The script is presented in the form of natural language in Algorithm 1.

CNN is a class of ANN, most commonly applied to analyze visual imagery [54]. The CNNs are comprised of neurons that self-optimize through learning. Each neuron receives an input and performs an operation (such as a *scalar product* followed by a *nonlinear* function). From the input vectors to the final output of the class score, the entire of the network expresses a single perceptive score function

Algorithm 1. The Read_Cyber_Attacks.py Python Script

Script: *Read_Cyber_Attacks.py*

Inputs: The 9 *attack.csv* files and the 9 *labels.csv* files

Step 1: Read the 9 *labels.csv* files. Label the attacks and normal flow according to Table 4.

Step 2: Read the 9 *attack.csv* files

Step 3: Append all tables in a single *Table* of 21,017,588 rows and 116 columns (115 network flow features, 1 label (representing the respective class), 16,166,306 benign records and 4,851,280 malicious packets)

Step 4: Shuffle the rows of the *Table* (to eliminate any pattern on the original data)

Step 5: Apply Undersampling to benign instances so that their number of records is comparable or equal to the vectors corresponding to malicious traffic. (The Syn_Dos attack is only 0,044% of the normal flow).

Step 6: Split the *Final_Table* 9,702,560 x 116 (4,851,280 benign and malicious packets) into 2 tables: (1) **Train Data** (80% of *Final_Table*) and (2) **Test Data** (20% of *Final_Table*). Both tables have flow records related to the 9 Cyber Attacks and to Benign flow.*

Outputs: (1) **Train Data** (7,762,024 x 116) and (2) **Test Data** (1,940,536 x 116)

* Authors tried additional combinations of **Train Data** and **Test Data** (70%-30%, 75%-25%, 85%-15%). Through the trial and error process, the combination of 80%-20% was selected, due to the fact that it offers the best results. Due to the vast amount of results and the extent of the manuscript, the results of the other segmentations will not be mentioned.

4 The Hybrid Ensemble COREM Model

It has already been mentioned that the hybrid ensemble modeling approach introduced in this paper consists of the combination of three ANN with the following architectures: CNN, RNN- LSTM and MLP.

(the weight). CNNs are comprised of three types of layers. These are *convolutional* layers, *pooling* layers and *fully connected* layers. When these layers are stacked, a CNN architecture has been formed [55]. After passing through a convolutional layer, the image becomes abstracted to a feature map, also called an activation map. Convolutional layers convolve the input and pass its result to the next

layer. Pooling layers reduce the dimensions of data by combining the outputs of neuron clusters at one layer into a single neuron in the next layer. Local pooling combines small clusters, tiling sizes such as 2×2 are commonly used. Global pooling acts on all the neurons of the feature map [56]. There are two common types of pooling in popular use: max and average. Max pooling uses the maximum value of each local cluster of neurons in the feature map, while average pooling takes the average value [57]. Fully connected layers connect every neuron in one layer to every neuron in another layer. It is the same as a traditional MLP. The flattened matrix goes through a fully connected layer to classification. In the fully connected layer, each node in the output layer connects directly to a node in the previous layer. This layer performs the task of classification based on the features extracted through the previous layers and their different filters [57]. Figure 1 presents an example from the CNN application for digit recognition. The input is an image of a digit with dimensions 28×28 . The combination of the (5×5 core) convolutional layer with the (2×2) max pooling layer is applied twice. Then we have the flattened matrix that goes through the fully connected layer to sort which number is the digit.

The architecture of the RNNs consists of an N input layer, one or more hidden layers and an output layer. The output of the RNNs in each phase, depends on the output computed in the previous. RNNs are characterized by a chain-like structure of repeating modules. These modules are used as memory that stores important information from previous processing steps. Unlike feedforward neural networks, RNNs include a feedback loop that allows the neural network to accept a sequence of inputs. This means that the output from step $t-1$ is fed back to the network to influence the outcome of step t . This is repeated in each subsequent step. The same task is recurrently performed for every element of the sequence. In other words, RNNs benefit from their memory that stores previously calculated

information. They are widely used in *natural language modeling* [58].

Figure 2 presents a simple RNN with one input unit, one output unit and one recurrent hidden unit expanded into a full network, where X_t is the input at time step t and h_t is the output at time step t . During the training process, RNNs use the backpropagation algorithm, a prevalent algorithm applied in calculating gradients and adjusting weight vectors. It adjusts the weights following the modification of the feedback process.

It is difficult for RNNs to remember information for a very long time period, because the backpropagated gradients either grow or shrink at each time step and they are making the training weights either explode or vanish notably. However, the *long short-term memory* (LSTM) has addressed this issue. A typical LSTM unit is composed of three gates, an *input gate*, an *output gate* and a *forget gate*, which regulates information into and out of the memory cell [59]. The input gate decides the ratio of input and it has an effect on the value of the cell's state. The forget gate controls the amount of information that can remain in the memory cell. The output gate determines the amount of information in the memory cell that can be used to compute the output activation of the LSTM unit [60]. Figure 3 illustrates the architecture of an LSTM node. X_t is the input, h_t is the output of the LSTM node, h_{t-1} is the output of the previous LSTM node, C_t and C_{t-1} are the cell states at time t and $t-1$, respectively, σ is the sigmoid function (decide what to forget), \tanh is the tangent hyperbolic function and b is the bias. The “x” is where the scaling of information is applied, and the “+” is where the adding information process is performed [61].

MLP are well-known algorithms with very strong and well-defined mathematical foundations [62–65]. Thus, the detailed description of their mathematical equations and their theoretical approach algorithms will be omitted. There is a vast literature review about their structure. Secondly, the detailed delineation is out of the scope of this research.

Fig. 1 A CNN sequence to classify an image to bridge or skyscraper

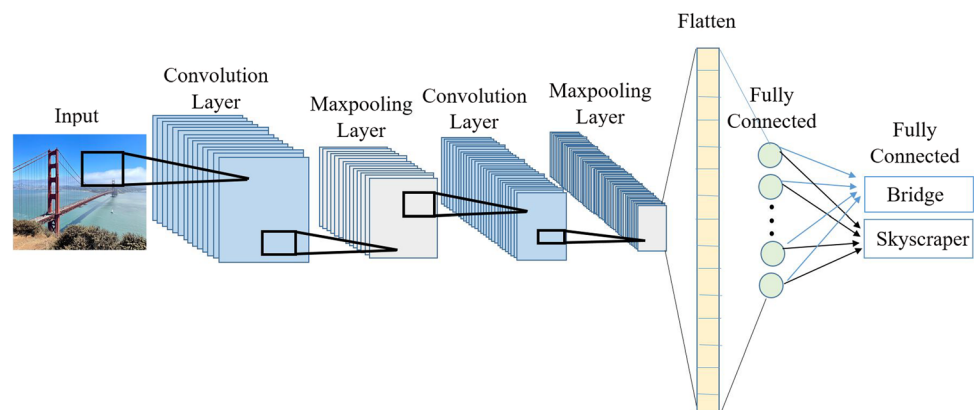


Fig. 2 Sequential processing in a recurrent neural network (RNN)

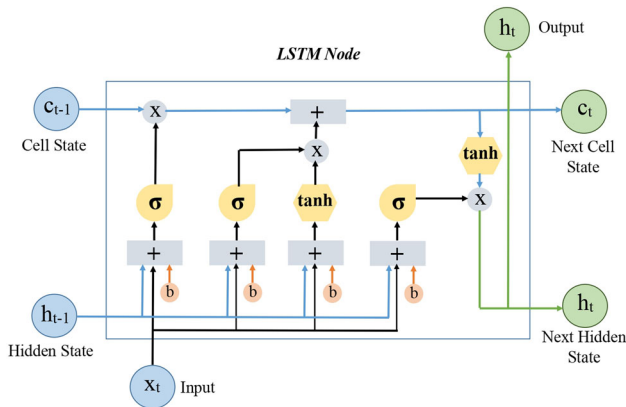
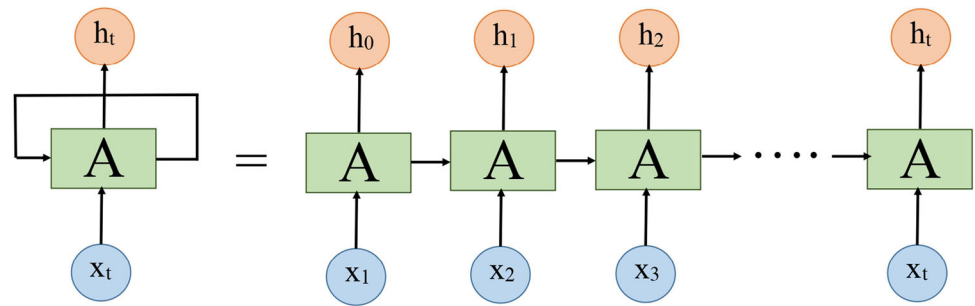


Fig. 3 Structure of an LSTM node

4.1 Architecture of the hybrid COREM² model

The proposed hybrid approach uses a 2-D CNN [66], a 2-LSTM layer RNN and a MLP with two hidden layers. The model accepts as input a record of the following dimensions $1 \times 115 \times 1$, which is the record with the 115 features and 1 channel. The output of the classification is an integer in the closed interval $[0, 9]$. The number 0 corresponds to *benign traffic*, whereas each of the integer numbers of the attack set $A = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ is related to each one of the 9 attacks under consideration. Its architecture (layers) alongside the parameters that being set is displayed in Table 5. The input of each layer is the output of the previous one.

The number of filters tested in the CNN were 2^n , where $n = 1, 2, \dots, 5$. A stride denotes how many steps we are moving in convolution each time. By default its value is equal to 1. For this research, the values of strides that were tested are 1, 2 and 3. The sizes of the kernel for the testing strides were 2, 3, 4 and 5. For the LSTM, we have used 25, 50, 75, 100, 125, 150, 175 and 200 nodes and for the MLP 15, 20, 25, 30, 40 and 50 nodes. Last but not least, for the dropout layers, the values that were tested were 0.2, 0.5 and 0.8. The decision for the layers of the model, as well as for the optimal values of the hyperparameters, was made through a trial-and-error process. The architecture of the hybrid approach COREM² is presented in Fig. 4.

The architecture of the developed CNN model is clearly shown in Fig. 1. The goal of the *convolutional layer* (CONL) is to apply filtering and to extract potential features for each record. The *pooling* layer reduces the size of the series while preserving the important characteristics identified by the CONL. The model comprises of four 2D-CONLs and two max pooling layers, in order to deepen to the features of the records.

The *flattened layer* prepares the data vectors to be used as input to the LSTM. Right before the first LSTM layer, the data is filtered through a dropout layer [67]. The role of the dropout layer is to avoid overfitting of the model. Like the name of the layer indicates, a percentage of the nodes is dropped. Thus, only a part of the information passes to the LSTM layers. In this stage, the model is asked to complete the classification without having considered all available information, but to try to decode and locate the patterns in the vectors available. The LSTMs layers have been chosen to be trained on the output of the CNN, avoiding the long-term dependency problem, as they remember information for long periods of time. LSTM has feedback connections which could find easier patterns between the features of the observations. Two LSTM layers have been employed in order to cope with the vast numbers of records that are used in the training process.

After the LSTM layers, another dropout layer is applied for the aforementioned reasons. The dense layers have been used at the end of the network, in order to map the extracted features. Last but not least, the softmax layer has been applied, in order to determine which class each observation corresponds to. The main goal of this hybrid algorithm is the exhaustive search for the patterns that exist in the characteristics of each record.

All experiments have been performed in Python using a computer with an Intel Core i9-9900 CPU (3.10 GHz) processor, DDR4 memory (32 GBytes) and GPU NVIDIA GeForce RTX 2070 Super (8GBytes).

Keras [68] and Tensorflow [69, 70] libraries have been employed to build the model's architecture. Based on the literature, the *sparse categorical cross-entropy* [71], the *Adam optimizer* [72] and the *ReLU* functions [73] were applied in all layers, as the *loss function*, the *optimizer* and

Table 5 Model’s architecture (layers and parameters set)

NN	Layer	Parameters set
CNN	2-D Covolution	(Filters, kernel size, strides) = (8, 5, 1)
	2-D Covolution	(Filters, kernel size, strides) = (8, 5, 1)
	2-D Max pooling	Pool size = (1,2)
	2-D Covolution	(Filters, kernel size, strides) = (16, 4, 1)
	2-D Covolution	(Filters, kernel size, strides) = (16, 4, 1)
	2-D Max pooling	Pool size = (1,2)
	Flatten	–
RNN	Dropout	0.5
	LSTM	(Nodes, Return Sequences) = (200, True)
	LSTM	(Nodes, Return Sequences) = (100, False)
MLPN	Dropout	0.5
	Dense	Nodes = 50
	Dense	Nodes = 20
	Dense	Nodes = 10

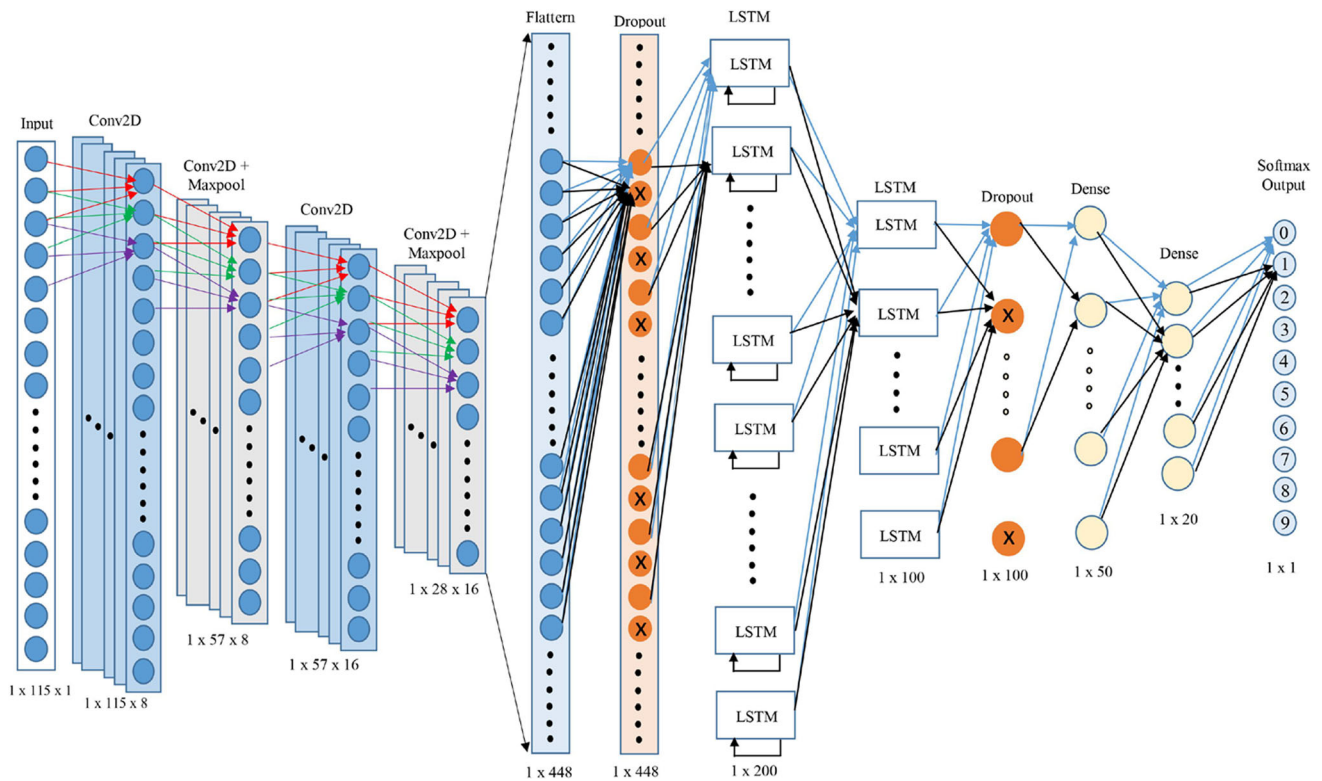


Fig. 4 Hybrid COREM² model

the *activation function*, respectively. Only in the last dense layer, the *softmax activation function* has been used, due to the fact that this is a classification problem.

4.2 Evaluation of the proposed model

The accuracy is the overall evaluation index of the developed machine learning models [74]. However, additional

performance indices have been used to estimate the efficiency of the algorithms. This is a multiclass classification research, so the “one versus all” strategy [75] was employed. Table 6 presents the considered validation indices.

Here, TP, TN, FP and FN refer to true positives, true negatives, false positives and false negatives, respectively. PREC is the measure of the correctly identified positive

Table 6 Calculated indices for the evaluation of the multiclass classification approach

Index	Abbreviation	Calculation
Sensitivity (SNS) (also known as True Positive Rate—TPR or Recall REC)	SNS, REC, TPR	$SNS = TP/(TP + FN)$
Specificity, (SPC) (also known as True Negative Rate—TNR)	SPC, TNR	$SPC = TN/(TN + FP)$
Accuracy (ACC)	ACC	$ACC = (TP + TN)/(TP + FP + FN + TN)$
F1 Score	F1	$F1 = 2*(Precision*Sensitivity)/(Precision + Sensitivity)$
Precision PREC (also known as Positive Predictive Value)	PREC	$PREC = TP/(TP + FP)$

cases from all the predicted positive cases. Thus, it is useful when the cost of false positives is high. Moreover, SNS is the measure of the correctly identified positive cases from all the actual positive cases. It is important when the cost of false negatives is high. SPC is the true negative rate or the proportion of negatives that are correctly identified. The *F1* score can be interpreted as the harmonic mean (weighted average) of the precision and recall. As it is known from the literature, accuracy can be seriously considered when the class distribution is balanced while *F1* score is a better metric when there are imbalanced classes as in the above case [76].

Furthermore, the validation of the Train Data was performed using the tenfold cross-validation method [77], in order to confirm the optimal values of the hyper parameters and that our model avoids overfitting (combination with dropout layer).

5 Experimental results

The developed model was trained for 50 epochs. As shown in Fig. 5, the accuracy during the training process is gradually increasing, whereas the loss is gradually decreasing. The observed spikes are due to the existence of the dropout layer in the proposed network. The vast majority of the records were correctly classified. The overall accuracy in Training is equal to 98.64% and the overall loss is as high as 0.0354. However, the performance in training cannot be used to prove the actual generalization ability of a ML model.

To prove the generalization ability and the credibility of the model, we have obtained and analyzed the values of the performance indices in the testing phase. The confusion matrix and the corresponding indices for the testing data are presented in Tables 7 and 8, respectively. The overall accuracy in testing is equal to 98.92%. This is a clear indicator of the model's generalization capacity when considering first time seen data. The number of the misclassified instances in testing is limited. For the intrusion detection problem, the algorithm seems to generalize with

high level of success for all classes. A comprehensive presentation of the results is given in Tables 7 and 8.

According to the above evaluation indices, the proposed model classifies correctly the 98.95% of the benign traffic. This means that the algorithm misclassifies 16,070 malicious cases as benign out of the 970,762 records that have been classified as benign. On the other hand, only 4240 malicious instances (out of millions) have been misclassified as benign, while they are malicious. The percentage of malicious classified as benign is negligible, strictly speaking mathematically, only 0.46%. On the contrary, in theory, even one infected package can cause damage. However, as there is no perfect model in the literature capable of offering absolute accuracy (for any domain of interest) the proposed hybrid model is very robust, and in fact, it is more than satisfactory in meeting the needs of the task.

It is confirmed that COREM² can stand out with extremely high success rate the benign and malicious traffic. As the subclasses of the malicious traffic are concerned, the model seems to classify the 9 cyber-attacks correctly and not to confuse them with each other something that further enhances its robustness.

Further attention must be given to class #7 (SYN DoS). Although the accuracy and the specificity indicators are very high, it is obvious that the other three indicators SNS, *F1* score and precision are not as high as they should be. Looking at the confusion matrix, one can notice that from the 1416 respective records, the 345 are sorted incorrectly. The performance indices for the aforementioned class are also high; however, they are not as high as the indices corresponding to the other classes. This is due to the fact that class #7 can be characterized as a minority one compared to the others. This means that this problem could be resolved with oversampling (e.g., SMOTE [78]) or with the availability of more respective available data vectors. This is a minor problem as it is related to the data and not to the robust introduced algorithm.

If we attempt an overall assessment, we clearly see that the performance indices have excellent values (even in the case of the minority class they are still high) and they prove

Fig. 5 COREM²'s Accuracy and Loss on Training Data for 50 Epochs

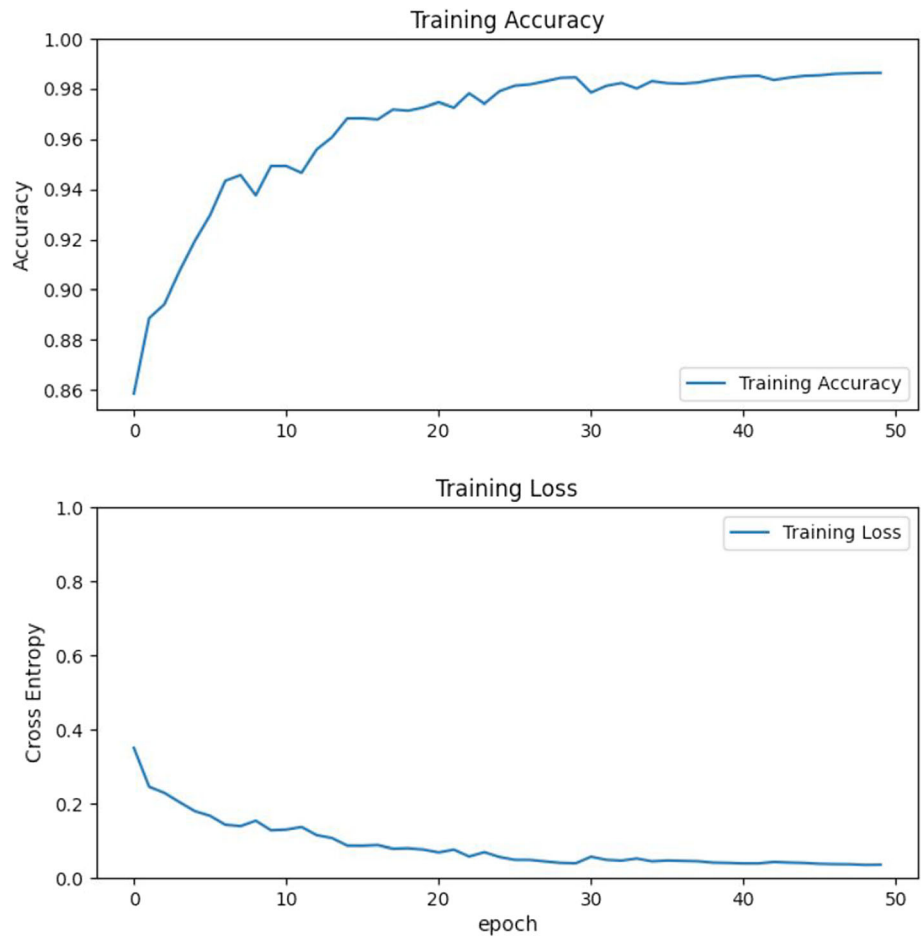


Table 7 Confusion matrix for the introduced model in testing

	Predicted class										
	Label	0	1	2	3	4	5	6	7	8	9
Actual Class	0	954,689	564	2419	8419	3	3	66	233	1333	3030
	1	1235	183,123	326	0	0	0	1	1	0	2
	2	15	290	228,340	0	0	0	1	0	0	2
	3	228	0	0	85,893	2	0	0	0	0	1
	4	22	0	0	1	13,110	0	0	0	0	0
	5	0	0	0	0	1	287,954	0	0	0	0
	6	100	1	0	8	0	0	18,592	1	1	8
	7	343	0	0	0	0	0	1	1071	0	1
	8	211	0	0	0	0	0	0	0	20,220	0
	9	2086	0	0	0	0	0	4	0	0	126,556

that the model performs a very reliable classification for all classes. Almost all indices have values are close to 1, and there are only 3 indices with performance in the interval [0.75, 0.82]. We conclude that we are introducing a robust algorithm capable of generalizing with very high level of

success for the 9 out of 10 classes and with reliable performance for the minority class.

To the best of our knowledge, no research effort has been made on the Kitsune dataset (except the previous work of the authors [8]). Thus, no comparison with other

Table 8 Evaluation indices for the introduced model in the testing process

Index	0	1	2	3	4	5	6	7	8	9
SNS	0.9834	0.9915	0.9987	0.9973	0.9982	1	0.9936	0.7564	0.9897	0.9838
SPC	0.9956	0.9995	0.9984	0.9954	1	1	1	0.9999	0.9993	0.9983
ACC	0.9895	0.9987	0.9984	0.9955	1	1	0.999	0.997	0.9992	0.9973
F1	0.9895	0.9934	0.9934	0.952	0.9989	1	0.9949	0.7869	0.9632	0.9801
PREC	0.9956	0.9954	0.9881	0.9106	0.9995	1	0.9961	0.8201	0.9381	0.9765

approaches can be done. Moreover, the developed hybrid approach discussed herein performs very efficiently, in cyber-threats detection. Moreover, in the cases of KDDcup9 and NSLKDD datasets, which are the two most frequently used in the literature, COREM² outperforms existing approaches. More specifically, Elekar, 2015, achieved an accuracy as high as 92.62% [14]. The hybrid fuzzy neural network developed by Ganeshkumar and Pandesswari, 2016, has been tested for the DoS, Probe, R2L, U2R attacks with an accuracy equal to 78.61%, 95.52%, 85.30% and 78.61%, respectively. Dash, 2016 proposed a hybrid IDS, suitable for unbalanced cases. The IDS was successfully tested on the NSLKDD dataset. It achieved precision and recall indices equal to 85.6% and 89.4%, respectively, outperforming the hybrid CNN/RNN model introduced by Kolosnjaji et al. that was applied on the detection of malware “*system call*” sequences. Mizano et al. identified procedures of malicious software using the HTTP headers of network traffic, with a precision equal to 97.1%, whereas the false positive ratio was only 1%. These results were achieved using a *neural network* with two hidden layers. Thamilarasu et al. developed a *deep belief network* (DBN) to fabricate a feed-forward ANN for IoT (Internet of Things) cases. The precision was equal to 96% and the recall rate was as high as 98.7% for the detection of DDoS attacks. Shone et al. deployed a *deep* auto-encoder for cyber intrusion detection, reaching an average accuracy as high as 97.85%. Overall, the proposed hybrid IDS, outperforms all of the above approaches.

6 Beginning to end approach for cyber intrusion detection

In this section, the authors propose a holistic approach to cyber intrusion detection. This approach starts by capturing the packets of a network traffic, continues with the storage and processing of information and ends with the classification of each packet of the net flow, in nine malicious and one benign class by employing the COREM² hybrid neural network. For the entire implementation of this research effort, code was written from scratch in the Python programming language. Figure 6 (also refer to Appendix

Section) presents this holistic approach for cyber intrusion detection. Its architecture will be described in detail in the following section.

6.1 Capturing network traffic

Python’s Pyshark library [79] has been used to capture network’s traffic packages. Pyshark is a wrapper for the Wireshark [80] CLI interface, Tshark, so all of the Wireshark decoders are available. Pyshark allows python packet parsing using Wireshark dissectors and parsing from a capture file or a live capture. This library gives access to attributes like packet number, relative and delta times, IP addresses, MAC addresses, packets length, protocols, flags and other info. It allows the user to sniff the netflow of network traffic and to save packages with their corresponding features in a **.pcap* format (PCAP file).

In more detail, the Pyshark library gives the user the capability of packet capturing:

- (a) Per time: One can set a time limit (e.g., 60 s or 100 s) and record all packets emerging in this time interval, regardless their number. The result will be a **.pcap* file containing all packages available during the temporal period set by the user.
- (b) Per number of packages: one can set the limit of packages (e.g., 1000 packages or 20,000 packages), and when their recording is completed, they are saved in a **.pcap* file regardless the required time horizon.

While writing code for both cases, it was observed that the capturing needs of the Pyshark libraries do not meet the hybrid structure of this research. The problem was resolved by, modifying the respective *Pyshark* source code. These changes were sufficient to enable Pyshark to be linked to the next stages of the methodology.

For the needs of this paper, the authors employed the method of capturing packages, i.e., per number of packages and specifically, per 1000 packages in order to develop the **.pcap* file. Specifically, each developed file was named *PCAP_i.pcap*, where $i = 1, 2, 3, \dots$ depending on its respective group (See Fig. 6). The model immediately starts capturing groups, comprising of 1,000 records which

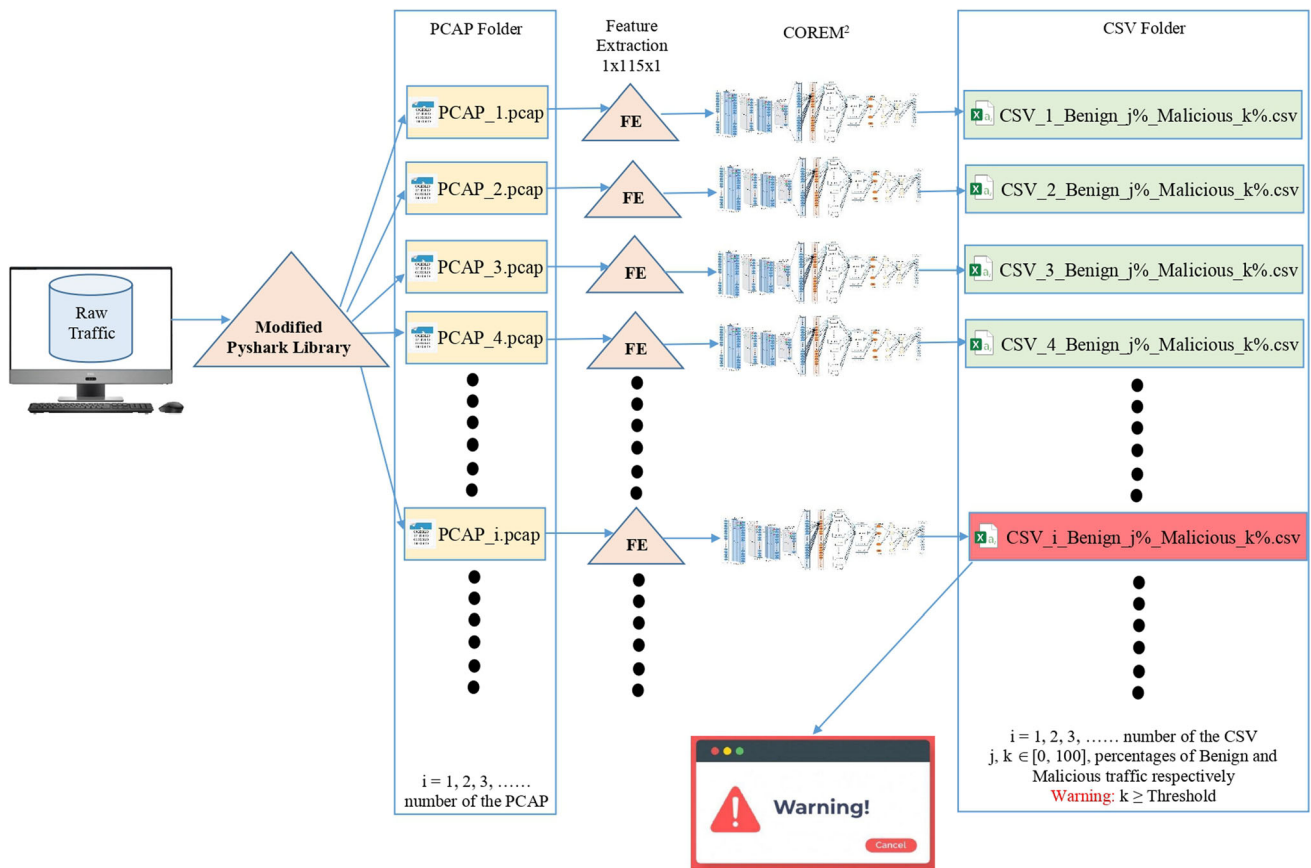


Fig. 6 A holistic approach for cyber intrusion detection, using the hybrid NN COREM²

are obviously stored in separate *.pcap files. Then it waits for the second stage which is the feature extraction process, which determines the features to be used as input to the hybrid neural network.

In the first stage of capturing, the experts and specialists are the ones who decide either the time horizon of the packets’ capturing or the number of captured packets based on the needs of each network and on the type of each cyber-attack. There are different characteristics and different sampling approaches that can lead to better results.

6.2 Feature extraction

In the second stage, the feature extraction process for each PCAP_i.pcap file is initiated. The proposed hybrid COREM² is always trained based on the 115 features (which have been described in detail in Sect. 3) after the performance of feature extraction for each package in the *.pcap file. During this process, each package is reshaped to get the exact dimension (1 × 115 × 1) required by COREM² in order to be processed.

This process is not linear (see Fig. 6). This means that the application does not wait for the feature extraction of the first PCAP_1.pcap file to finish before launching the

feature extraction for the second PCAP_2.pcap file. With the creation of a new PCAP_i.pcap, the model starts the feature extraction directly, without any delay, in order to be used later as input to COREM². This means that many PCAP_i.pcap feature extractions can be performed at the same time and that the model will always start processing the last received packets, just as soon as they are captured.

6.3 Traffic classification with COREM²

After feature extraction is performed for each PCAP_i.pcap, it is fed into the COREM² hybrid neural network. However, this process is not runs sequentially. The model does not wait for the PCAP_1.pcap classification process to finish before proceeding to run on the PCAP_2.pcap. Thus, when the feature extraction of the PCAP_i.pcap ends, it is immediately fed to an instance of COREM². As a result, many instances of COREM² can run simultaneously.

Each COREM² class performs the following actions:

- (a) It processes the packets derived from the feature extracted PCAP_i.pcap in order to classify them into benign packets or packets containing one of the nine cyber-attacks, as analyzed in Sect. 3.

- (b) It creates a **.csv* file named *CSV_i_Benign_j%_Malicious_k%.csv*, where $i = 1, 2, 3, \dots$ which corresponds to i of *PCAP_i.pcap* and states which **.pcap* comes from the csv, j, k are the percentages of the benign and malicious packets of

Obviously deleting files can also be done manually by the user himself.

This holistic model for cyber intrusion detection is described in Algorithm 2 in the form of natural language:

For this model, a case study was performed on a com-

Algorithm 2. The Capture_and_Classify.py Python Script

Script: *Capture_and_Classify.py*

Inputs: Raw traffic on a computer

Step 1: Create *PCAP folder* and *CSV folder* on Desktop

Step 2: Capture Network Traffic. For every 1000 packets save a *PCAP_i.pcap*, where $i = 1, 2, 3, \dots$ in *PCAP folder*.

Step 3: When a *PCAP_i.pcap* is being created, apply feature extraction in each packet of the *PCAP_i.pcap* (115 features), and reshape each packet in dimensions $1 \times 115 \times 1$.

Step 4: Feed the 1000 reshaped packets into an instance of COREM²

Step 5: COREM² applies the classification of the packets into Benign and Malicious Traffic.

Outputs: *CSV_i_Benign_j%_Malicious_k%.csv*, where $i = 1, 2, 3, \dots$ which corresponds to i of *PCAP_i.pcap*, $j, k \in [0, 100]$ are the percentages of the benign and malicious packets of the corresponding *PCAP_i.pcap*, respectively. If $k \geq \text{Threshold } \%$, show a warning.

the corresponding *PCAP_i.pcap*, respectively. It should be clarified that k, j belong to the closed interval $[0, 100]$ (see Fig. 6). In each **.csv*, the percentages of benign packages, malicious packages, as well as the exact percentages of malicious packages belonging to each of the 9 cyber-attacks are written in detail. In addition, within the **.csv* files, there are 10 lines (1 for benign packets and 1 for each of the cyber-attacks) followed by the number of packets belonging to each of these 10 classes, as shown in Fig. 7. This can help in a later study of the **.pcap* file packages in which the cyber-attack was detected, for example, the IP addresses from which they came, or the Mac Addresses, and whatever else is needed.

- (c) , If malicious traffic is above a certain threshold in a *pcap* file (e.g., $k \geq 5\%$), a pop up window with a warning for the user will be shown, in order for the user to proceed to the necessary actions to address the problem.

The proposed approach allows the user to save all **.pcap* and **.csv* files in the *PCAP folder* and *CSV folder*, respectively (see Fig. 7), which is automatically created on the desktop by the respective code. (Obviously the path and the name of the folder created can be changed by the user.) The advantage is that the **.pcap* and **.csv* files are available for future study. However, it is allowed to delete files after a certain period of time to save storage space.

puter with the features mentioned in Sect. 4.1, in a home network. After the application of the holistic model, it was observed that the processing efficiency extends to 10,000 packets per 8 s. Obviously, the methodology is not limited to its application on a computer, but it can be customized to work on any device that has Internet access.

7 Conclusions and future work

This extensive research has a twofold contribution to the existing literature. Many conclusions can be drawn and several points are highlighted that can be studied further.

Initially, the authors propose the novel hybrid IDS COREM² which extends the existing work of the authors entitled “A Hybrid Deep Learning Ensemble for Cyber Intrusion Detection” [51]. The architecture of this hybrid approach comprises of a 2D-CNN, RNN (LSTM) and MLP. The decision for the architecture of the hybrid neural network and the optimal values of the parameters was reached after exhaustive trial-and-error process and after performing a variety of experiments. The combination of tenfold cross-validation and dropout layers was used to evaluate the performance of the introduced deep learning model. All experiments have been performed in Python using a computer with an Intel Core i9-9900 CPU (3.10 GHz) processor, DDR4 memory (32 GBytes) and GPU NVIDIA GeForce RTX 2070 Super (8GBytes). The

	A	B	C	D	E	F
1	Percentages of Benign and Malicious Traffic,					
2	Benign ,98					
3	Malicious,2					
4						
5	Percentages of each cyber-attack,					
6	OS Scan,					
7	Fuzzing,					
8	Video Injection,					
9	ARP MitM,					
10	Active Wiretap,					
11	SSDP Flood,0.6					
12	SYN DoS,0.8					
13	SSL Renegotiation,0.6					
14	Mirai,					
15						
16	Class of each packet					
17	Benign,1,2,3,4,5,6,7,8,9,16,17,18,19,20,21,22,23,24,25,26,27,28,29,					
18	OS Scan,					
19	Fuzzing,					
20	Video Injection,					
21	ARP MitM,					
22	Active Wiretap,					
23	SSDP Flood,10,11,12,13,14,15					
24	SYN DoS,980,981,982,983,984,985,986,987					
25	SSL Renegotiation,100,105,117,118,119,132					
26	Mirai,					
27						
28						

Fig. 7 Example of a CSV_i_Benign_j%_Malicious_k%.csv

model was trained and evaluated using the *Kitsune Network Attack Dataset* [31]. It includes records from 9 cyber-attacks and also normal network flow. The model generalizes with high rate of success. Given the fact that we are dealing with a multiclass classification problem the “one-versus-all” strategy was used. Apart from the confusion matrix on testing data, which also indicates the very good performance of the network, all indices confirm the robustness of the introduced architecture. It has been noted that the performance of the minority class # 7 (SYN_DoS attack) is a little bit lower but still remarkably high.

One major advantage of the developed hybrid model is that it can specialize for specific features without losing its ability to generalize. Each one of the CNN, LSTM and

MLP specializes for different cases in the literature. More specifically, CNN is used for image classification, LSTM is used for time series modeling and MLP can classify records that are not linearly separable. A hybrid model inherits the advantages of each methodology and combines them to correctly classify cyber-attacks. The biggest disadvantage of the introduced model is that it is “too deep” and it takes a long time to be able to train in the first place. However, once the training is done, the hybrid model works quickly to process the new measurements. Given the fact that in this research effort, authors use the combination of well-known algorithms that are already used in other fields like diagnostic apps, financial apps, educational, futuristic Internet of Things apps, there is no restriction on

the application domain of the hybrid IDS. Following the optimization of the hyperparameters and the correct choice of the number of layers, the hybrid IDS could also be used to solve regression problems.

And the above leads to the second contribution of this research is the fact that it introduces a “beginning to end” integrated approach for cyber intrusion detection, using the COREM² hybrid neural network. The holistic model has the ability to save all network traffic packages. The proposed method has an average processing power of 10,000 packets per 8 s. It potentially can be used in any device that has Internet access. This rate can be improved further in order to achieve faster and thus more efficient classification. Our research effort paves the way for many future approaches.

Although the introduced model has proved to be very robust, there is always room for improvement. One of the future research direction will be the use of the *synthetic minority oversampling technique* (SMOTE) [78] in order for class #7 SYB_DoS to stop being a minority one and to allow the improvement of its respective classification

indices. Furthermore, different time windows for the 115 extracted features could be used in order to test other methodologies, on the Kitsune Dataset. Finally, after the success of this research effort, the authors could continue testing other combinations of layers, with other parameters trying to achieve an even more efficient architecture. There is always room for improvement. After all, no model is perfect, and a model is good when it is practically useful. A significant advantage that *Kitsune Network Attack Dataset* provides is that the original network captures are included to the dataset. Thus, in a future research a different way of feature extraction could be used, aiming to offer a different number and type of features.

Appendix

See Fig. 8.

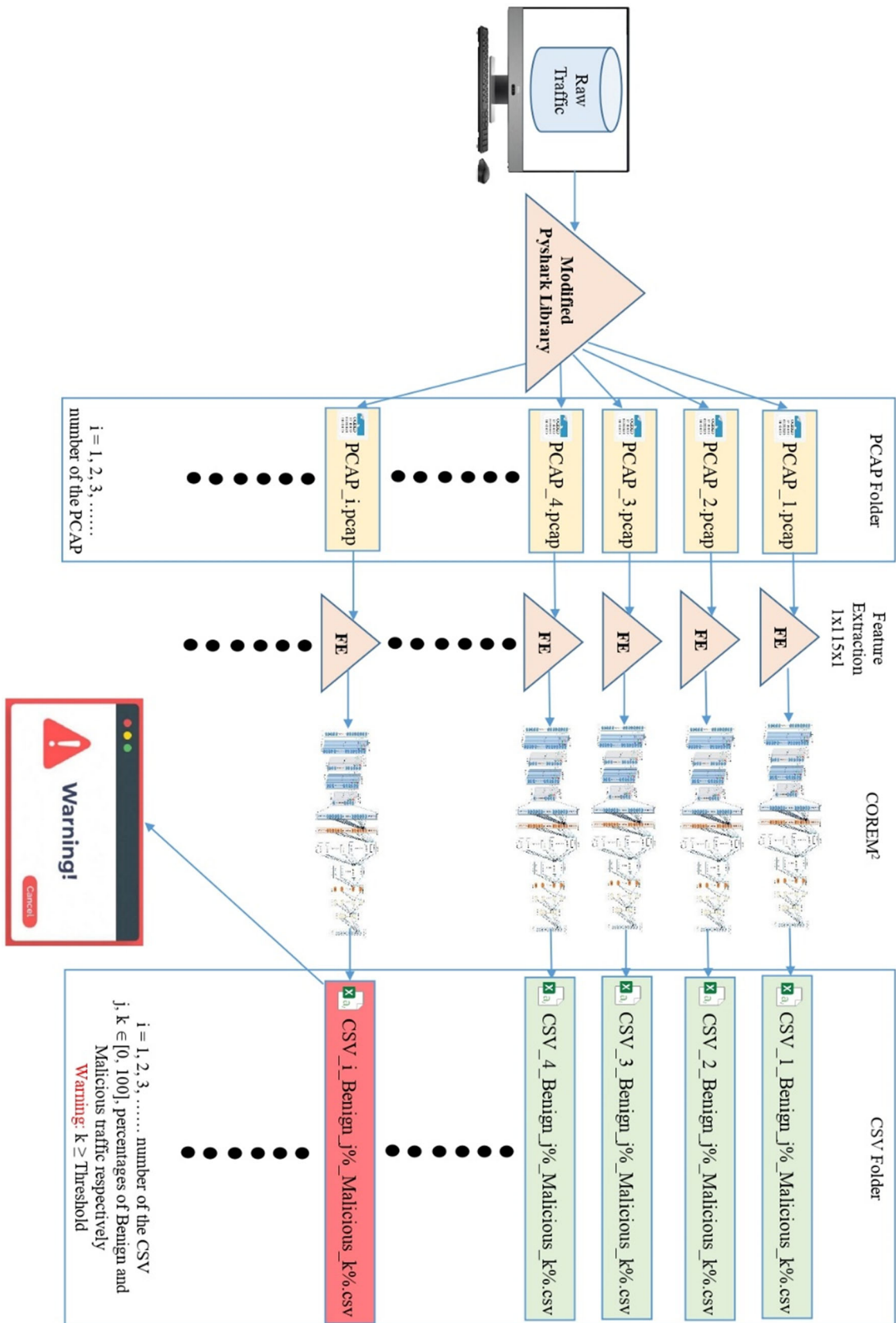


Fig. 8 A holistic approach for cyber intrusion detection, using the hybrid NN COREM²

Declarations

Conflict of interest The authors declare no conflict of interest. There is no financial and personal relationships with other people or organizations that can inappropriately influence their work. There is no professional or other personal interest of any nature or kind in any product, service or company that could be constructed as influencing the position presented in, or the review of, the manuscript.

References

- Kuypers MA, Maillart T, Paté-Cornell E (2016) An empirical analysis of cyber security incidents at a large organization. Department of Management Science and Engineering, Stanford University, School of Information, UC Berkeley, 30
- Yadav S, Shukla S (2016) Analysis of k-fold cross-validation over hold-out validation on colossal datasets for quality classification. In 2016 IEEE 6th International conference on advanced computing (IACC). IEEE. pp 78–83
- Ahmim A, Derdour M, Ferrag MA (2018) An intrusion detection system based on combining probability predictions of a tree of classifiers. *Int J Commun Syst* 31(9):e3547
- Ahmim A, Maglaras L, Ferrag MA, Derdour M, Janicke H (2019) A novel hierarchical intrusion detection system based on decision tree and rules-based models. In 2019 15th international conference on distributed computing in sensor systems (DCOSS). IEEE. pp 228–233
- Statista, <https://www.statista.com/statistics/273575/average-organizational-cost-incurred-by-a-data-breach/>. Accessed 28 Nov 2021
- Holzinger K, Mak K, Kieseberg P, Holzinger A (2018) Can we trust machine learning results? artificial intelligence in safety-critical decision support. *Ercim News* 112:42–43
- IBM, <https://www.ibm.com/topics/cybersecurity>. Accessed 30 Nov 2021
- Psathas AP, Iliadis L, Papaleonidas A, Bountas D (2021) A hybrid deep learning ensemble for cyber intrusion detection. In international conference on engineering applications of neural networks. Springer, Cham. pp 27–41
- Stone M (1974) Cross-validated choice and assessment of statistical predictions. *J Roy Stat Soc Ser B (Methodol)* 36(2):111–133
- Böhme R, Christin N, Edelman B, Moore T (2015) Bitcoin: economics, technology, and governance. *J Econ Perspect* 29(2):213–238
- Sherman AT, Javani F, Zhang H, Golaszewski E (2019) On the origins and variations of blockchain technologies. *IEEE Secur Priv* 17(1):72–77
- Damopoulos D, Menesidou SA, Kambourakis G, Papadaki M, Clarke N, Gritzalis S (2012) Evaluation of anomaly-based IDS for mobile devices using machine learning classifiers. *Secur Commun Netw* 5(1):3–14
- Li Y, Xia J, Zhang S, Yan J, Ai X, Dai K (2012) An efficient intrusion detection system based on support vector machines and gradually feature removal method. *Expert Syst Appl* 39(1):424–430
- Elekar KS (2015) Combination of data mining techniques for intrusion detection system. In 2015 international conference on computer, communication and control (IC4). IEEE. pp 1–5
- Ganeshkumar P, Pandeewari N (2016) Adaptive neuro-fuzzy-based anomaly detection system in cloud. *Int J Fuzzy Syst* 18(3):367–378
- Meidan Y, Bohadana M, Mathov Y, Mirsky Y, Shabtai A, Breitenbacher D, Elovici Y (2018) N-baiot—network-based detection of iot botnet attacks using deep autoencoders. *IEEE Pervasive Comput* 17(3):12–22
- Soe YN, Feng Y, Santosa PI, Hartanto R, Sakurai K (2020) Machine learning-based IoT-botnet attack detection with sequential architecture. *Sensors* 20(16):4372
- Zhang C, Jiang J, Kamel M (2005) Intrusion detection using hierarchical neural networks. *Pattern Recogn Lett* 26(6):779–791
- Dash T (2017) A study on intrusion detection using neural networks trained with evolutionary algorithms. *Soft Comput* 21:2687–2700
- Demertzis K, Iliadis L, Bougoudis I (2020) Gryphon: a semi-supervised anomaly detection system based on one-class evolving spiking neural network. *Neural Comput Appl* 32(9):4303–4314
- Shon T, Moon J (2007) A hybrid machine learning approach to network anomaly detection. *Inf Sci* 177(18):3799–3821
- Buczak AL, Guven E (2015) A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Commun Surv Tutor* 18(2):1153–1176
- Xie M, Hu J, Han S, Chen HH (2012) Scalable hypergrid k-NN-based online anomaly detection in wireless sensor networks. *IEEE Trans Parallel Distrib Syst* 24(8):1661–1670
- Kolosnjaji B, Zarras A, Webster G, Eckert C (2016) Deep learning for classification of malware system call sequences. In Australasian joint conference on artificial intelligence. Springer, Cham. pp 137–149
- Pascanu R, Stokes JW, Sanossian H, Marinescu M, Thomas A (2015) Malware classification with recurrent networks. In 2015 IEEE international conference on acoustics, speech and signal processing (ICASSP). IEEE. pp 1916–1920
- Mizuno S, Hatada M, Mori T, Goto S (2017) Botdetector: a robust and scalable approach toward detecting malware-infected devices. In 2017 IEEE international conference on communications (ICC). IEEE. pp 1–7
- Demertzis K, Tziritas N, Kikiras P, Sanchez SL, Iliadis L (2019) The next generation cognitive security operations center: adaptive analytic lambda architecture for efficient defense against adversarial attacks. *Big Data Cognit Comput* 3(1):6
- Cordonsky I, Rosenberg I, Sicard G, David EO (2018) DeepOrigin: end-to-end deep learning for detection of new malware families. In 2018 international joint conference on neural networks (IJCNN). IEEE. pp 1–7
- Gibert Llauradó D (2016). Convolutional neural networks for malware classification (Master's thesis, Universitat Politècnica de Catalunya)
- Loukas G, Vuong T, Heartfield R, Sakellari G, Yoon Y, Gan D (2017) Cloud-based cyber-physical intrusion detection for vehicles using deep learning. *IEEE Access* 6:3491–3508
- Thamilarasu G, Chawla S (2019) Towards deep-learning-driven intrusion detection for the internet of things. *Sensors* 19(9):1977
- Shone N, Ngoc TN, Phai VD, Shi Q (2018) A deep learning approach to network intrusion detection. *IEEE Trans Emerg Topics Comput Intell* 2(1):41–50
- Kdd Cup 1999. <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>. Accessed 8 Mar 2021
- Nsl kdd. <https://www.unb.ca/cic/datasets/nsl.html>. Accessed 8 Mar 2021
- Nisa M, Shah JH, Kanwal S, Raza M, Khan MA, Damaševičius R, Blažauskas T (2020) Hybrid malware classification method using segmentation-based fractal texture analysis and deep convolution neural network features. *Appl Sci* 10(14):4966
- He Y, Mendis GJ, Wei J (2017) Real-time detection of false data injection attacks in smart grid: a deep learning-based intelligent mechanism. *IEEE Trans Smart Grid* 8(5):2505–2516
- Miller ST, Busby-Earle C (2017) Multi-perspective machine learning a classifier ensemble method for intrusion detection.

- In proceedings of the 2017 international conference on machine learning and soft computing, pp 7–12
38. Nakamoto S (2008) Bitcoin: a peer-to-peer electronic cash system. *Decentralized Business Review*, 21260
 39. Serrano W (2019) The blockchain random neural network in cybersecurity and the Internet of Things. In IFIP international conference on artificial intelligence applications and innovations. Springer, Cham. pp 50–63
 40. Giannoutakis KM, Spathoulas G, Filelis-Papadopoulos CK, Collen A, Anagnostopoulos M, Votis K, Nijdam NA (2020) A blockchain solution for enhancing cybersecurity defence of IoT. In 2020 IEEE international conference on blockchain (Blockchain). IEEE. pp 490–495
 41. Demertzis K, Iliadis L, Tziritas N, Kikiras P (2020) Anomaly detection via blockchain deep learning smart contracts in industry 4.0. *Neural Comput Appl* 32(23):17361–17378
 42. Mora OB, Rivera R, Larios VM, Beltrán-Ramírez JR, Maciel R, Ochoa A (2018) A Use Case in Cybersecurity based in Blockchain to deal with the security and privacy of citizens and Smart Cities Cyberinfrastructures. In 2018 IEEE international smart cities conference (ISC2). IEEE. pp 1–4
 43. Mylrea M, Gouriseti SNG (2018) Blockchain for supply chain cybersecurity, optimization and compliance. In 2018 Resilience Week (RWS). IEEE. pp 70–76
 44. Wang B, Dabbaghjamanesh M, Kavousi-Fard A, Mehraeen S (2019) Cybersecurity enhancement of power trading within the networked microgrids based on blockchain and directed acyclic graph approach. *IEEE Trans Ind Appl* 55(6):7300–7309
 45. Berman DS, Buczak AL, Chavis JS, Corbett CL (2019) A survey of deep learning methods for cyber security. *Information* 10(4):122
 46. Ctu-13 Dataset. <https://mcfp.weebly.com/the-ctu-13-dataset-a-labeleddataset-with-botnet-normal-and-background-traffic.html>. Accessed 8 Mar 2021
 47. Unsw-nb15 Dataset. <https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/>. Accessed 30 Nov 2021
 48. Bot-IoT Dataset. https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/bot_iot.php. Accessed 8 Mar 2021
 49. CSE-CIC-IDS2018 Dataset. <https://www.unb.ca/cic/datasets/ids-2018.html>. Accessed 30 May 2019
 50. Tor-Nontor Dataset. <https://www.unb.ca/cic/datasets/tor.html>. Accessed 8 Mar 2021
 51. Android Malware Dataset. <https://www.unb.ca/cic/datasets/andmal2017.html>. Accessed 30 Nov 2021
 52. Mirsky Y, Doitshman T, Elovici Y, Shabtai A (2018) Kitsune: an ensemble of autoencoders for online network intrusion detection. arXiv preprint [arXiv:1802.09089](https://arxiv.org/abs/1802.09089)
 53. Kitsune Network Attack Dataset. <https://archive.ics.uci.edu/ml/datasets/Kitsune+Network+Attack+Dataset>. Accessed 8 Mar 2021
 54. Valueva MV, Nagornov NN, Lyakhov PA, Valuev GV, Chervyakov NI (2020) Application of the residue number system to reduce hardware costs of the convolutional neural network implementation. *Math Comput Simul* 177:232–243
 55. O’Shea K, Ryan N (2015) “An introduction to convolutional neural networks.” arXiv preprint [arXiv:1511.08458](https://arxiv.org/abs/1511.08458)
 56. Ciresan DC, Meier U, Masci J, Gambardella LM, Schmidhuber J (2011) Flexible, high performance convolutional neural networks for image classification. In Twenty-second international joint conference on artificial intelligence
 57. Ciregan D, Meier U, Schmidhuber J (2012) Multi-column deep neural networks for image classification. In 2012 IEEE conference on computer vision and pattern recognition. IEEE. pp 3642–3649
 58. Martin E, Cundy C (2017) Parallelizing linear recurrent neural nets over sequence length. arXiv preprint [arXiv:1709.04057](https://arxiv.org/abs/1709.04057)
 59. Aydın S (2019) Deep learning classification of neuro-emotional phase domain complexity levels induced by affective video film clips. *IEEE J Biomed Health Inform* 24(6):1695–1702
 60. Mahdavi S, Ghorbani AA (2019) Application of deep learning to cybersecurity: a survey. *Neurocomputing* 347:149–176
 61. Le XH, Ho HV, Lee G, Jung S (2019) Application of long short-term memory (LSTM) neural network for flood forecasting. *Water* 11(7):1387
 62. Saleh AI, Talaat FM, Labib LM (2019) A hybrid intrusion detection system (HIDS) based on prioritized k-nearest neighbors and optimized SVM classifiers. *Artif Intell Rev* 51(3):403–443
 63. Yeung DS, Li JC, Ng WW, Chan PP (2015) MLPNN training via a multiobjective optimization of training error and stochastic sensitivity. *IEEE Trans Neural Netw Learn Syst* 27(5):978–992
 64. Malik A, Kumar A, Rai P, Kuriqi A (2021) Prediction of multi-scalar standardized precipitation index by using artificial intelligence and regression models. *Climate* 2021(9):28
 65. Dawson CW, Wilby RL (2001) Hydrological modelling using artificial neural networks. *Prog Phys Geogr* 25(1):80–108
 66. Chambon S, Galtier MN, Arnal PJ, Wainrib G, Gramfort A (2018) A deep learning architecture for temporal sleep stage classification using multivariate and multimodal time series. *IEEE Trans Neural Syst Rehabil Eng* 26(4):758–769
 67. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. *J Mach Learn Res* 15(1):1929–1958
 68. Ketkar N (2017) Introduction to keras. In Deep learning with Python. Apress, Berkeley, CA. pp 97–111
 69. Dillon JV, Langmore I, Tran D, Brevdo E, Vasudevan S, Moore D, Saurous, RA (2017) Tensorflow distributions. arXiv preprint [arXiv:1711.10604](https://arxiv.org/abs/1711.10604)
 70. Tensorflow. <https://www.tensorflow.org/>. Accessed 8 Mar 2021
 71. Mirchev A, Ahmadi SA (2018) Classification of sparsely labeled spatio-temporal data through semi-supervised adversarial learning. arXiv preprint [arXiv:1801.08712](https://arxiv.org/abs/1801.08712)
 72. Zhang Z (2018) Improved adam optimizer for deep neural networks. In 2018 IEEE/ACM 26th international symposium on quality of service (IWQoS). IEEE. pp 1–2
 73. Agarap AF (2018) Deep learning using rectified linear units (relu). arXiv preprint [arXiv:1803.08375](https://arxiv.org/abs/1803.08375)
 74. Psathas AP, Papaleonidas A, Iliadis L (2021) A Machine Learning Approach for Recognition of Elders’ Activities Using Passive Sensors. In IFIP International Conference on Artificial Intelligence Applications and Innovations. Springer, Cham. pp 157–170
 75. Psathas AP, Papaleonidas A, Papathanassiou G, Iliadis L, Valkaniotis S (2021) Hybrid computational intelligence modeling of coseismic landslides’ severity. in international conference on computational collective intelligence. Springer, Cham. pp 427–442
 76. Psathas AP, Papaleonidas A, Iliadis L (2020) Machine learning modeling of human activity using PPG signals. In: international conference on computational collective intelligence. Springer, Cham. pp 543–557
 77. Psathas AP, Papaleonidas A, Papathanassiou G, Valkaniotis S, Iliadis L (2020) Classification of coseismic landslides using fuzzy

- and machine learning techniques. In: international conference on engineering applications of neural networks. Springer, Cham. pp 15–31
78. Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP (2002) SMOTE: synthetic minority over-sampling technique. *J Artif Intell Res* 16:321–357
79. Pyshark. <https://pypi.org/project/pyshark/>. Accessed 30 Nov 2021
80. Wireshark. <https://www.wireshark.org/>. Accessed 30 Nov 2021

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.