







HEDL-IDS: A Hybrid Ensemble Deep Learning Approach for Cyber Intrusion Detection

Anastasios Panagiotis Psathas^(✉) , Lazaros Iliadis , Antonios Papaleonidas ,
and Dimitris Bountas 

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

{anspatha, liliadis, papaleon, dibounta}@civil.duth.gr

Abstract. The continuously increasing number of activities processed via the internet, often leaves the user vulnerable to cyber-attacks. The goal of the scientific community is to deploy innovative approaches and methodologies, capable to offer protection from potential cyber threats. This research effort aims to contribute to networks' security by introducing the *Hybrid Ensemble Deep Learning (HEDL)* Intrusion Detection System (IDS) that successfully detects nine serious cyber-attacks. Its architecture comprises of three *Deep Neural Networks (DNN)*, three *Convolutional Neural Networks (CNN)* and 3 *Recurrent Neural Networks (RNN)* using *Long-Short Term Memory (LSTM)* layers, running in parallel. The HEDL-IDS was successfully tested against the *UNSW-NB15* dataset, achieving an overall accuracy of 98.35% and 96.25% in the training and testing phases respectively. The performance of the proposed model was evaluated by calculating *Accuracy, Sensitivity, Specificity, Precision and F-1 Score*. The values of all above indices were higher than 0.92, indicating the accurate performance of the developed model. The HEDL-IDS was compared with 20 robust Machine Learning Classification algorithms, sealing its reliability.

Keywords: Hybrid · Ensemble · CNN · DNN · RNN · LSTM · Cyber attacks · Cyber intrusion detection

1 Introduction

Technology keeps evolving rapidly. Nowadays, the vast number of applications used by individuals or groups, for either personal or commercial use, increases. The wide introduction of the *Internet-of-Thing (IoT)* was determinant for the growth of these applications in everyday life [1]. However, the increasing use of computer networks and interconnected systems, paved the way for exploiting their weaknesses. Cyber-attacks cause severe damage and severe financial losses in large-scale networks [2]. Despite the tremendous development in network security, the existing solutions are unable to completely defend computer networks against the malicious threats [3]. The traditional security techniques such as hardware and software firewalls, user authentication and data encryption are not capable enough to fully safeguard networks' security, due to the

© IFIP International Federation for Information Processing 2022

Published by Springer Nature Switzerland AG 2022

I. Maglogiannis et al. (Eds.): AIAI 2022, IFIP AICT 646, pp. 116–131, 2022.

https://doi.org/10.1007/978-3-031-08333-4_10

fast development of intrusion techniques [4]. Thus, the deployment of other approaches and methodologies is imperative.

Intrusion Detection Systems (IDS), a rapidly growing field of study, were suggested in order to facilitate system's security. Using patterns of benign traffic or normal behavior or specific rules that describe a specific attack, IDSs can distinguish between normal and malicious actions [5]. There are two main types of cyber analytics in support of IDSs: misuse-based (sometimes also called signature-based) and anomaly-based [6]. *Signature-based* techniques detect anomalies by matching predefined attack's signatures [7]. Two of the main advantages of these methods are that they are simple to implement and they have low false positive rates. However, it is not feasible for them to detect new cyber threats. *Anomaly-based* (AB) detection techniques rely on the assumption that the intruder's behavior is different from the typical one [8]. They model the network's normal patterns, and they identify anomalies as deviations from them. *Anomaly-based Intrusion Detection Systems* (ABIDSs) can detect zero-day attacks, however they may often result in high *False Alarm Rates* (FARs) due to the fact that several previously unseen (yet legitimate) system behaviors may be classified as anomalies.

This paper introduces a novel *Hybrid Ensemble Deep Learning Approach* for Cyber Intrusion Detection (HEDL-IDS). The architecture of this Hybrid Model comprises of 3 *Deep Neural Networks* (DNN), 3 *Convolutional Neural Network* (CNN) and 3 *Recurrent Neural Network* (RNN) with Long-Short Term Memory (LSTM) layers. The majority vote of these 9 models has been applied for each observation, improving the accuracy and robustness of the results. Additionally, the aforementioned approach searches for the best deep learning structure, as all nine models have different architectures and they are using different parameters. To the best of our knowledge, it is the first time that such a Hybrid Ensemble Deep Learning architecture is employed in the literature, in order to perform Cyber Intrusion Detection (CID). This methodology was tested on a subset of the UNSW-NB15 intrusion detection dataset [9] which comprises of raw network packets, related to several well-known attacks e.g. Denial of Service (DoS), Worms, Fuzzers and Backdoors.

The subset of the UNSW-NB15 that was used in this research, comprises of 2 classes namely: the malicious flow and the benign one.

The rest of the paper is organized as follows: Sect. 2 performs a literature review of IDS based on Machine Learning (ML) and Deep Learning (DL), Sect. 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. Finally, Sect. 6 concludes the research.

2 Literature Review

Several ML approaches have been introduced for CID, during the last twenty years. Recently, several 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.

In 2012, Li et al. [10], presented an approach, capable to classify the predefined attack categories such as DoS (Denial of Service), Probe or Scan, U2R (User to Root), R2L (Root to Local), as well as normal traffic. This was achieved by utilizing the most popular KDD'99 cup dataset, by using the hyperplane-based SVM classifier with an RBF kernel. The obtained accuracy was as high as 98.6249%. In 2012, Koc et al. [11], developed a *Hidden Naïve Bayes* (HNB) model, achieving an accuracy equal to 93.72% and an error rate of 0.0628, improving significantly the accuracy of detecting denial-of-services (DoS) attacks. In 2017, Shapoorifard and Shamsinejad, [12] used the NSL-KDD dataset to develop a model, combining the k-Means clustering and the k-Nearest Neighbors (k-NN) algorithms, achieving an accuracy equal to 98%. In 2018, Malik and Khan [13], introduced a hybrid model that combines the *Binary Particle Swarm Optimization* with the Decision Tree Pruning, for network intrusion detection. In 2020, Sarker et al., introduced the “*IntruDTree*” machine-learning security model, which is characterized by low computational complexity and high accuracy, due to the performed feature dimensionality reduction [14].

In 2016, Kim et al. [15], developed a *Long-Short Term Memory* (LSTM) multi class classifier on the KDD Cup 1999 dataset, by considering an input vector of 41 features. In 2018, Zhang et al. [16], employed a convolutional neural network, to model the B2C (Business to Consumer) online transaction dataset of a commercial bank. The model achieved a Precision as high as 91% and a Recall equal to 94%. Convolutional neural networks were also used by Basumallik et al. [17] for packet-data anomaly detection, in phasor measurement units-based state estimator (PHMUB). The IEEE-30 bus and IEEE-118 bus systems, were used as the PHMUB. This research uses a probability of 0.5 with 512 neurons on a fully connected layer. The accuracy reached the value of 98.67%. Thamilarasu and Chawla [18], introduced a *Deep Belief* network to fabricate a feed-forward DNN for an *Internet of Things* (IoT) case. The proposed model was tested against five attacks, namely: The Sinkhole, the Wormhole, the Blackhole, the Opportunistic service and the DDoS. The results show a Precision of 96% and a Recall of 98.7% for the case of the DDoS attacks. Khan et al. [19], proposed an intrusion detection system based on the two-stage deep learning model, named TSDL. The KDD99 and the UNSW-NB15 network intrusion public datasets were considered for the TSDL training and testing, with an accuracy equal to 99.996% and 89.134% respectively. In 2020, Demertzis et al. suggested a *Blockchain Security Architecture* that aims to ensure network communication between traded Industrial IoT devices, following the Industry 4.0 standard, 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 Autoencoder Neural Network [20]. In 2021, Psathas et al. [21], presented a hybrid Intrusion Detecting System (IDS) comprising of a 2-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.

3 Dataset Description and Pre-processing

This research, has considered the *UNSW-NB15* network intrusion public dataset [9] which contains raw network packets, related to nine different attacks. It was developed in 2015 at the University of New South Wales, of the Australian Defense Force Academy Canberra, Australia [22]. The *IXIA PerfectStorm* tool [23] was utilized to create a hybrid normal and abnormal network traffic. The *IXIA* tool contains all information about new attacks that are updated continuously from a CVE site [24]. CVE is a list of publicly disclosed cybersecurity vulnerabilities that is free to search, use, and incorporate into products and services, per the terms of use. The simulation period was 16 h of the 22nd of January 2015 and 15 h of the 17th of February 2015. The total volume of captured data is 100 GBs.

The original dataset contains a vast number of network packets, 2,540,044 in total. The malicious traffic is related to nine different *Cyber Attacks* (*Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode and Worms*) and comprises of 321,283 packets. The remaining 2,218,761 records correspond to Normal flow. However, due to the fact that there is inhomogeneity in the instances (Cyber Attacks are only 12.5% of the original dataset) a subset of the *UNSW-NB15* dataset was used which was provided by the developers. The subset contains a total of 257,673 records, 164,673 of which corresponds to malicious traffic (nine cyber-attacks) and 93,000 to normal flow (see Table 1). This is a less unbalanced dataset, containing all the traffic of the eight *Cyber Attacks* a part of the *Generic Attack* and a part of the *Normal* flow. Nevertheless, there is still a major issue like *Worms* and *Shellcode* attacks, which correspond to minority classes. Therefore, it was obviously more feasible to perform a binary classification effort. Thus, the records corresponding to normal flow were assigned the value 0, whereas the ones related to the nine cyber-attacks were assigned the value 1 (see Table 1). The 42 features of the subset UNSW-NB15 are listed in the following Table 2.

The features *Proto, State* and *Service* are stored as strings (sequences of characters). Thus, they were transformed by the authors from nominal to numeric. The rest 39 features have numerical (either integer or float) values. The transformations are presented in the following Tables 3 and 4.

Due to the fact that the *Proto* feature has 133 different elements, the table with the correspondence to labels is omitted. For further information about the feature extraction process and the features, refer to [22].

Data handling has been achieved by writing code from scratch in Matlab. After labeling was completed, the dataset had the shape of a 257,673 x 43 *Table* (42 columns contain features' values and 1 contains the respective label). The data *Table* was divided in *Training* (75%) 193,256 rows X 43 columns and *Testing* (25%) 64,417 rows X 43 columns.

Table 1. Type, description and number of subset packets, the number of dataset packets and label of the nine (9) cyber attacks and normal flow considered in this research

Type	Description: the attacker...	# Subset's packets	# Dataset packets	Label
Normal	Natural transaction data	93,000	2,218,761	0
Fuzziers	Attempting to cause a program or network suspended by feeding it the randomly generated data	24,246	24,246	1
Analysis	It contains different attacks of port scan, spam and html files penetrations	2,677	2,677	1
Backdoors	A technique in which a system's security mechanism is bypassed stealthily to access a computer or its data	2,327	2,329	1
DoS	A malicious attempt to make a server or a network's resource unavailable to users, usually by temporarily interrupting or suspending the services of a host connected to the Internet	16,353	16,353	1
Exploits	The attacker knows of a security problem within an operating system or a piece of software and leverages that knowledge by exploiting the vulnerability	44,525	44,525	1
Generic	A technique works against all block ciphers (with a given block and key size), without consideration about the structure of the block-cipher	85,871	215,481	1
Reconnaissance	Contains all Strikes that can simulate attacks that gather information	13,987	13,987	1
Shellcode	A small piece of code used as the payload in the exploitation of software vulnerability	1,511	1,511	1
Worms	Attacker replicates itself in order to spread to other computers. Often, it uses a computer network to spread itself, relying on security failures on the target computer to access it	174	174	1
Total		257,673	2,540,044	

Table 2. Features’ abbreviation and respective description in the UNSW-NB15 subset

Feature	Description
proto	Transaction protocol
state	Indicates the state and its dependent protocol
dur	Record of total duration
sbytes	Source to destination transaction bytes
dbytes	Destination to source transaction bytes
sttl	Source to destination time to live value
dttl	Destination to source time to live value
sloss	Source packets retransmitted or dropped
dloss	Destination packets retransmitted or dropped
service	http, ftp, smtp, ssh, dns, ftp-data,irc and (-) if not much used service
Sload	Source bits per second
Dload	Destination bits per second
Spkts	Source to destination packet count
Dpkts	Destination to source packet count
swin	Source TCP window advertisement value
dwin	Destination TCP window advertisement value
stcpb	Source TCP base sequence number
dtcpb	Destination TCP base sequence number
smeansz	Mean of the row packet size transmitted by the src
dmeansz	Mean of the row packet size transmitted by the dst
trans_depth	The pipelined depth into the connection of http request/response transaction
res_bdy_len	Actual uncompressed content size of data transferred from the server http service
Sjit	Source jitter (mSec)
Djit	Destination jitter (mSec)
Sintpkt	Source interpacket arrival time (mSec)
Dintpkt	Destination interpacket arrival time (mSec)
tcprtt	TCP connection setup round-trip time, the sum of ‘synack’ and ‘ackdat’
synack	TCP connection setup time, the time between the SYN and SYN_ACK packets
ackdat	TCP connection setup time, the time between the SYN_ACK and ACK packets

(continued)

Table 2. (continued)

Feature	Description
is_sm_ips_ports	If the source and destination IP addresses are equal and port numbers are also equal then this variable takes value 1 else 0
ct_state_ttl	No. for each state according to the specific range of values for source/destination time to live
ct_flw_http_mthd	No. of flows that has methods such as Get and Post in http service
rate	No. of packets per Second

Table 3. Service with the corresponding label

Service	-	dhcp	dns	ftp	ftp-data	http	irc	pop3	radius	smtp	snmp	ssh	ssl
Service label	1	2	3	4	5	6	7	8	9	10	11	12	13

Table 4. State with the corresponding label

State	CON	ECO	FIN	INT	PAR	REQ	RST	URN	no	ACC	CLO
State label	1	2	3	4	5	6	7	8	9	10	11

4 The HEDL-IDS 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: MLP (Fig. 1), CNN (Fig. 2) and RNN (Fig. 3). Thus, the detailed description of their mathematical foundations will be very brief. There is a vast literature review about their structure [25–27].

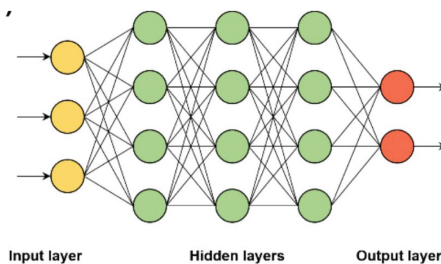


Fig. 1. 5-layer DNN model architecture (1 input layer, 1 output layer and 3 hidden layers)

A CNN is defined as a neural network that extracts features at a higher resolution, and then it converts them into more complex features at a coarser resolution, as presented in Fig. 2. Therefore, CNN is based on three types of layers namely: *Convolutional*, *Pooling* and *Fully-Connected*. When these layers are stacked, a CNN architecture has been formed [28]. The feature value at location (x, y) in the k^{th} feature map of M^{th} layer can be calculated as follows:

$$feature_{x,y,k}^M = W_k^{M^T} X_{x,y}^M + b_k^M \tag{1}$$

where $X_{x,y}^M$ is the input patch centered at location (x, y) , W_k^M is the weight vector of the k^{th} filter, and b_k^M is the bias term of the M^{th} layer. The activation value $activate_{x,y,k}^M$ and the pooling value $pool_{x,y,k}^M$ of convolution feature $feature_{x,y,k}^M$ can be calculated as follow:

$$activate_{x,y,k}^M = activation(feature_{x,y,k}^M) \tag{2}$$

$$pool_{x,y,k}^M = pooling(feature_{a,c,k}^M), \quad \forall (a, c) \in N_{x,y} \tag{3}$$

where $N_{x,y}$ is a local neighborhood at location (x, y) . The nonlinear activation functions are ReLU, Sigmoid, and Tangent Hyperbolic (Tanh). The Pooling operations are Average and Max Pooling.

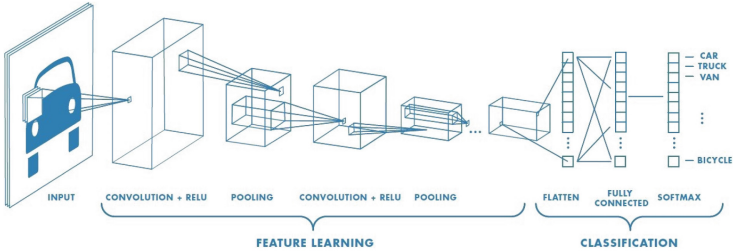


Fig. 2. CNN model architecture with 2 Convolution Layer, 2 Maxpooling Layer, one Flatten and 1 Fully Connected Layer.

The output of the RNNs in each phase, depends on the output computed in the previous state. 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 [29]. It is difficult for RNN to remember information for a very long time period, because the backpropagated gradients either grow or they shrink at each time step and they are making the training weights either explode or vanish notably. However, the LSTM has addressed this issue.

A typical LSTM unit is composed of three gates namely: an *input*, an *output*, and a *forget gate*. These gates, regulate information into and out of the memory cell. The *Input gate* decides the input ratio 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 [30]. 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 tanh function (gives weights), b is the Bias. The notation "x" is used where the scaling of information is applied, and the symbol "+" is used where adding information process is performed [31].

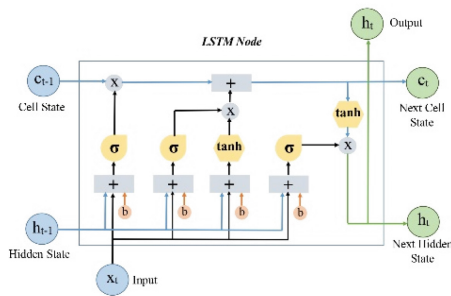


Fig. 3. Structure of an LSTM node

4.1 Architecture of the HEDL-IDS Model

The proposed hybrid approach uses three 2-D CNNs, three RNNs with LSTM layers and three DNNs. The CNN model accepts as input a record of the dimensions $1 \times 42 \times 1$, (1 observation, 42 features, and 1 channel). Moreover, the DNN and the RNN models accept as input a record of dimensions 1×42 (1 observation, 42 features). The output of the classification for each model is 0 (corresponding to *benign traffic*) or 1 (corresponding to *malicious traffic*). The number of filters applied in the CNNs were 2^n , where $n = 1, 2, \dots, 8$. A stride denotes the number of Convolution steps used each time. By default its value is equal to 1. The sizes of the kernel for the testing strides, were 2, 3, 4, and 5. For the LSTMs and the DNNs we have used 50, 100, 150, 200 and 250 nodes. The last layer for all 9 models is a *Dense Layer* with 2 nodes. Furthermore, the penultimate layer of the CNN models (after the *Flatten Layer*) is a *Dense Layer* with 20 nodes. To avoid overfitting, dropout layers were added after each Flatten, LSTM and Dense Layer. The values 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 parameters, was made through a trial and error process. The architecture of the nine models, alongside the used parameters is displayed in the following Table 5. The Dropout Layer's rate is 0.2. The input of each layer is the output of the previous one. Finally, Fig. 4, depicts the architecture of the hybrid approach.

Table 5. Layers and parameters set for the 9 models of HEDL-IDS

NN	Layer	Parameters set
1 st CNN	2-D Covolution	(filters, kernel size, strides) = (32, 3, 1)
	2-D Maxpooling	Pool size = (1, 2)
	Flatten	-
2 nd CNN	2-D Covolution	(filters, kernel size, strides) = (16, 3, 1)
	2-D Maxpooling	Pool size = (1, 2)
	2-D Covolution	(filters, kernel size, strides) = (64, 3, 1)
	2-D Maxpooling	Pool size = (1, 2)
	Flatten	-
3 rd CNN	2-D Covolution	(filters, kernel size, strides) = (8, 3, 1)
	2-D Maxpooling	Pool size = (1, 2)
	2-D Covolution	(filters, kernel size, strides) = (32, 3, 1)
	2-D Maxpooling	Pool size = (1, 2)
	2-D Covolution	(filters, kernel size, strides) = (128, 3, 1)
	2-D Maxpooling	Pool size = (1, 2)
	Flatten	-
1 st RNN	LSTM	Nodes = 100
2 nd RNN	LSTM	Nodes = 150
	LSTM	Nodes = 50
3 rd RNN	LSTM	Nodes = 250
	LSTM	Nodes = 150
	LSTM	Nodes = 50
1 st DNN	Dense	Nodes = 100
2 nd DNN	Dense	Nodes = 150
	Dense	Nodes = 50
3 rd DNN	Dense	Nodes = 250
	Dense	Nodes = 150
	Dense	Nodes = 50

*After each Flatten Layer always a Dense Layer with 20 nodes

**Last Layer of all Models is a Dense Layer with 2 nodes

***Dropout Layer after each Flatten, LSTM and Dense Layer

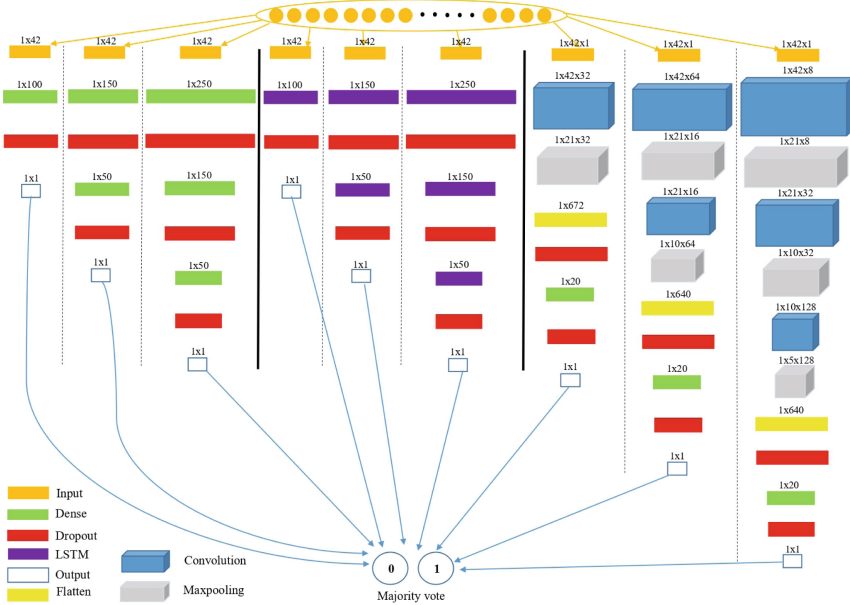


Fig. 4. Architecture of the HEDL-IDS. From left to right 1st DNN, 2nd DNN, 3rd DNN, 1st RNN, 2nd RNN, 3rd RNN, 1st CNN, 2nd CNN and 3rd CNN

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 Keras [32] and Tensorflow [33] libraries have been employed to build the model’s architecture. Based on the literature, in all layers, the *Categorical Crossentropy*, the *Adam Optimizer* and the *ReLU* functions were employed as the *Loss Function*, the *Optimizer* and the *Activation Function* respectively. The *Softmax Activation Function* has been used in the last dense layer of each model.

After training all models, the final prediction was calculated using majority vote. The output space uses majority vote to determine the final y_i . Therefore, y_i is given as follows:

$$y_i = [y_{i1} \dots y_{ij} \dots y_{in}]^T \tag{4}$$

where n is the model’s number, y_{ij} is the prediction of the model’s output for the data point i in model j and y_{ij} is defined as follows:

$$y_{i,j} = \arg \max_k [\text{soft max}(y_{i,j}^*)] \tag{5}$$

where $k \in \{0, 1\}$ for all binary classification.

As it is easily understood from Table 5 and Fig. 4, the complexity of the models increases as their number increases. For example, the 1st RNN has one LSTM layer, the 2nd RNN has two LSTM layers and so on. Thus, if the algorithm consists of four RNNs, the 4th would have 4 LSTM layers. Therefore, for the RNNs, every time a new

model is developed, an LSTM layer is added. Respectively, a dense layer is added for the case of the DNNs, and a Convolution and a Maxpool Layers are added for the CNNs. Furthermore, the number of models is also editable. One could choose to deploy 1 DNN, 4 RNN and 2 CNN. In this research, using the specific dataset, number three is the optimal for each algorithm.

5 Evaluation and Experimental Results

The Accuracy is the overall evaluation index of the developed Machine Learning models. However, four additional performance indices have been used to estimate the efficiency of the algorithms. The following Table 6 presents the validation indices used herein.

Table 6. Calculated indices for the evaluation of the binary classification approach

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

where 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 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.

The training of the model was performed for 50 epochs for all nine models. The HEDL-IDS performs very well in training, by correctly classifying the majority of the records. The overall accuracy in Training was equal to 98.35%. To be assured that the model can generalize, the authors examined whether the model is efficient in data that has not been used in training (Testing Data). 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 as high as 96.25%. Once again, the excellent performance of the algorithm is noticeable. Even for the first time seen data, the number of the misclassification instances is limited. For the intrusion detection problem, the algorithm seems to generalize with high level of success.

Table 7. Confusion matrix for the testing data

	Predicted class		
	Label	0	1
Actual class	0	22205	1110
	1	1303	39799

Table 8. Evaluation indices for the testing data

Index	SNS	SPC	ACC	F1	PREC
Value	0.9446	0.973	0.962	0.948	0.952

An overall assessment, clearly shows that the performance indices have excellent values and they prove that the model performs a very reliable classification, distinguishing with high accuracy the normal flow from the cyberthreats. Overall, both confusion matrix and indices, indicate a very good performance of the model on the specific data. The algorithm seems to generalize with high level of success, as all indices have values over 0.9 for the intrusion detection problem.

To complete the detailed presentation of the results, HEDL-IDS will be compared with classic (but powerful) Machine Learning classification algorithms. A wide spectrum of ML algorithms were employed on the available dataset, namely: *Fine Tree, Medium Tree, Coarse Tree, Linear SVM, Quadratic SVM, Cubic SVM, Fine Gaussian SVM, Medium Gaussian SVM, Coarse Gaussian SVM, Fine KNN, Medium KNN, Coarse KNN, Cosine KNN, Cubic KNN, weighted KNN, Boosted Trees, Bagged Trees, Subspace Discriminant, RUSBoosted Trees an Subspace KNN*. The results for each algorithm, as well as, the comparison with the proposed model are presented in Table 9.

It is clearly shown in Table 9, that all SVM and KNN algorithms cannot handle the specific dataset efficiently, as there is a problem with the False Positives. The aforementioned algorithms seem to be struggling to identify the Normal class. However, they seem to identify with high accuracy the malicious traffic. The Naïve Bayes approach, seems to have the exact opposite problem. It identifies with high accuracy the Normal traffic but it cannot pinpoint the malicious one. Tree algorithms seem to perform noticeably better. The best performance of all ML algorithms is achieved by the *Ensemble Methods: Boosted Trees, Subspace KNN, Subspace Discriminant and RUSBoosted Trees*. This is an indicator that the dataset can be best handled by the ensemble techniques. Overall, it has been shown that the HEDL-IDS is the optimal methodology.

Table 9. Evaluation indices for HEDL-IDS and the machine learning algorithms

Model	SNS	SPC	ACC	F1	PREC
Fine Tree	0.885	0.958	0.930	0.906	0.928
Medium Tree	0.854	0.953	0.915	0.886	0.921
Coarse Tree	0.803	0.937	0.883	0.847	0.896
Linear SVM	0.961	0.704	0.729	0.408	0.259
Quadratic SVM	0.940	0.706	0.730	0.419	0.270
Cubic SVM	0.229	0.538	0.404	0.250	0.276
Fine Gaussian SVM	0.868	0.711	0.730	0.444	0.298
Medium Gaussian SVM	0.912	0.707	0.730	0.426	0.278
Coarse Gaussian SVM	0.949	0.703	0.727	0.406	0.258
Fine KNN	0.927	0.710	0.734	0.436	0.285
Medium KNN	0.918	0.710	0.734	0.438	0.288
Coarse KNN	0.941	0.707	0.732	0.424	0.274
Cosine KNN	0.918	0.710	0.734	0.438	0.288
Weighted KNN	0.934	0.710	0.735	0.439	0.287
Boosted Trees	<u>0.912</u>	<u>0.939</u>	<u>0.929</u>	<u>0.901</u>	<u>0.890</u>
Subspace Discriminant	<u>0.928</u>	<u>0.855</u>	<u>0.875</u>	<u>0.804</u>	<u>0.709</u>
RUSBoosted Trees	<u>0.842</u>	<u>0.963</u>	<u>0.915</u>	<u>0.888</u>	<u>0.939</u>
Cubic KNN	0.917	0.710	0.733	0.437	0.287
Kernel Naïve Bayes	0.418	0.993	0.497	0.589	0.997
Subspace KNN	<u>0.804</u>	<u>0.916</u>	<u>0.873</u>	<u>0.829</u>	<u>0.856</u>
HEDL-IDS	0.945	0.973	0.962	0.948	0.952

6 Conclusions and Future Work

This paper introduces a *Hybrid Ensemble Deep Learning Intrusion Detection System* (HEDL-IDS). The architecture of this hybrid approach comprises of three CNNs, three RNNs and three DNNs running in parallel. For each record, the output space is the majority vote from the nine models. The model was trained and evaluated using the *UNSW-NB15* dataset [9]. It includes records from nine Cyber Attacks and also Normal Netflow. The parameters and the layers of the hybrid model were determined through a trial and error process. The overall accuracy for the training and testing data was as high as 98.35% and 96.525% respectively. The values of the performance indices were above 0.9 for both benign and malicious traffic, for the majority of the cases.

Although the results were very good, there is always room for improvement. There are already plans for future expansion of this research. The first scenario that the authors have to consider is the development of the HEDL-IDS or of a different model to deal with the multi-class classification problem. Another potential scenario is the use of the

mathematically based, *Synthetic Minority Over-sampling Technique* (SMOTE) approach [34] for the *Worms* or *Shellcode* attacks. A third scenario is to perform the HEDL-IDS or other approach on the original *UNSW-NB15* dataset. Moreover, the introduced model could be tested on other data sets and finally other Deep Learning techniques, in order to figure out if there is a better approach to classify network traffic. Finally, the authors could test other combinations of layers, with other parameters and maybe a more efficient architecture could emerge. After all, no model is perfect, a model is good when it is practically useful.

References

1. Alqahtani, H., Sarker, I.H., Kalim, A., Hossain, S.M.M., Ikhtlaq, S., Hossain, S.: Cyber intrusion detection using machine learning classification techniques. In: Chaubey, N., Parikh, S., Amin, K. (eds.) COMS2 2020. CCIS, vol. 1235, pp. 121–131. Springer, Singapore (2020). https://doi.org/10.1007/978-981-15-6648-6_10
2. Sarker, I.H., Kayes, A.S.M., Badsha, S., Alqahtani, H., Watters, P., Ng, A.: Cybersecurity data science: an overview from machine learning perspective. *J. Big Data* **7**(1), 1–29 (2020). <https://doi.org/10.1186/s40537-020-00318-5>
3. Mohammadi, S., Mirvaziri, H., Ghazizadeh-Ahsae, M., Karimipour, H.: Cyber intrusion detection by combined feature selection algorithm. *J. Inf. Secur. Appl.* **44**, 80–88 (2019)
4. Tavallae, M., Stakhanova, N., Ghorbani, A.A.: Toward credible evaluation of anomaly-based intrusion-detection methods. *IEEE Trans. Syst. Man Cybern. Part C (Appl. Rev.)* **40**(5), 516–524 (2010)
5. Ahmim, A., Maglaras, L., Ferrag, M.A., Derdour, M., Janicke, H.: 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), pp. 228–233. IEEE, May 2019
6. Buczak, A.L., Guven, E.: A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Commun. Surv. Tutor.* **18**(2), 1153–1176 (2015)
7. Kabir, E., Hu, J., Wang, H., Zhuo, G.: A novel statistical technique for intrusion detection systems. *Futur. Gener. Comput. Syst.* **79**, 303–318 (2018)
8. Hwang, K., Cai, M., Chen, Y., Qin, M.: Hybrid intrusion detection with weighted signature generation over anomalous internet episodes. *IEEE Trans. Dependable Secure Comput.* **4**(1), 41–55 (2007)
9. The UNSW-NB15 Dataset. <https://research.unsw.edu.au/projects/unsw-nb15-dataset>
10. Li, Y., Xia, J., Zhang, S., Yan, J., Ai, X., Dai, K.: An efficient intrusion detection system based on support vector machines and gradually feature removal method. *Expert Syst. Appl.* **39**(1), 424–430 (2012)
11. Koc, L., Mazzuchi, T.A., Sarkani, S.: A network intrusion detection system based on a Hidden Naïve Bayes multiclass classifier. *Expert Syst. Appl.* **39**(18), 13492–13500 (2012)
12. Shapoorifard, H., Shamsinejad, P.: Intrusion detection using a novel hybrid method incorporating an improved KNN. *Int. J. Comput. Appl.* **173**(1), 5–9 (2017)
13. Malik, A.J., Khan, F.A.: A hybrid technique using binary particle swarm optimization and decision tree pruning for network intrusion detection. *Cluster Comput.* **21**(1), 667–680 (2018)
14. Sarker, I.H., Abushark, Y.B., Alsolami, F., Khan, A.I.: Intrudtree: a machine learning based cyber security intrusion detection model. *Symmetry* **12**(5), 754 (2020)
15. Kim, J., Kim, J., Thu, H.L.T., Kim, H.: Long short term memory recurrent neural network classifier for intrusion detection. In: 2016 International Conference on Platform Technology and Service (PlatCon), pp. 1–5. IEEE, February 2016

16. Zhang, Z., Zhou, X., Zhang, X., Wang, L., Wang, P.: A model based on convolutional neural network for online transaction fraud detection. *Secur. Commun. Netw.* **2018** (2018)
17. Basumallik, S., Ma, R., Eftekharijad, S.: Packet-data anomaly detection in PMU-based state estimator using convolutional neural network. *Int. J. Electr. Power Energy Syst.* **107**, 690–702 (2019)
18. Thamilarasu, G., Chawla, S.: Towards deep-learning-driven intrusion detection for the internet of things. *Sensors* **19**(9), 1977 (2019)
19. Khan, F.A., Gumaei, A., Derhab, A., Hussain, A.: A novel two-stage deep learning model for efficient network intrusion detection. *IEEE Access* **7**, 30373–30385 (2019)
20. Demertzis, K., Iliadis, L., Tziritas, N., Kikiras, P.: Anomaly detection via blockchained deep learning smart contracts in industry 4.0. *Neural Comput. Appl.* **32**(23), 17361–17378 (2020). <https://doi.org/10.1007/s00521-020-05189-8>
21. Psathas, A.P., Iliadis, L., Papaleonidas, A., Bountas, D.: A hybrid deep learning ensemble for cyber intrusion detection. In: Iliadis, L., Macintyre, J., Jayne, C., Pimenidis, E. (eds.) *EANN 2021. PINNS*, vol. 3, pp. 27–41. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-80568-5_3
22. Moustafa, N., Slay, J.: UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In: *2015 Military Communications and Information Systems Conference (MilCIS)*, pp. 1–6. IEEE, November 2015
23. The IXIA PerfectStorm tool. <http://www.ixiacom.com/products/perfectstorm>
24. CVE. <https://cve.mitre.org/>
25. Yeung, D.S., Li, J.C., Ng, W.W., Chan, P.P.: MLPNN training via a multiobjective optimization of training error and stochastic sensitivity. *IEEE Trans. Neural Netw. Learn. Syst.* **27**(5), 978–992 (2015)
26. Baek, J., Choi, Y.: Deep neural network for predicting ore production by truck-haulage systems in open-pit mines. *Appl. Sci.* **10**(5), 1657 (2020)
27. Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., Alsaadi, F.E.: A survey of deep neural network architectures and their applications. *Neurocomputing* **234**, 11–26 (2017)
28. O’Shea, K., Nash, R.: An introduction to convolutional neural networks. arXiv preprint [arXiv: 1511.08458](https://arxiv.org/abs/1511.08458) (2015)
29. Martin, E., Cundy, C.: Parallelizing linear recurrent neural nets over sequence length. arXiv preprint [arXiv:1709.04057](https://arxiv.org/abs/1709.04057) (2017)
30. MahdaviFar, S., Ghorbani, A.A.: Application of deep learning to cybersecurity: a survey. *Neurocomputing* **347**, 149–176 (2019)
31. Le, X.H., Ho, H.V., Lee, G., Jung, S.: Application of long short-term memory (LSTM) neural network for flood forecasting. *Water* **11**(7), 1387 (2019)
32. Ketkar, N.: Introduction to keras. In: *Deep Learning with Python*, pp. 97–111. Apress, Berkeley (2017)
33. Dillon, J.V., et al.: Tensorflow distributions. arXiv preprint [arXiv:1711.10604](https://arxiv.org/abs/1711.10604) (2017)
34. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: SMOTE: synthetic minority over-sampling technique. *J. Artif. Intell. Res.* **16**, 321–357 (2002)