Computers & Security 114 (2022) 102588

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

A comprehensive deep learning benchmark for IoT IDS

Rasheed Ahmad[a,*], Izzat Alsmadi[b], Wasim Alhamdani[a], Lo'ai Tawalbeh[b]

[a] University of the Cumberlands, 6178 College Station Drive, Williamsburg, KY 40769, USA
[b] University of Texas A&M San Antonio, One University Way, San Antonio, TX 78224, USA

A R T I C L E I N F O

Article history:
Received 19 April 2021
Revised 9 November 2021
Accepted 21 December 2021
Available online 28 December 2021

Keywords:
Intrusion detection system (IDS)
Machine learning
Deep learning
Large-scale attacks
Internet of Things (IoT)
Benchmark network dataset

A B S T R A C T

The significance of an intrusion detection system (IDS) in networks security cannot be overstated in detecting and responding to malicious attacks. Failure to detect large-scale attacks like DDoS not only makes the networks vulnerable, but a failure of critical lifesaving medical and industrial equipment can also put human lives at risk. Lack of availability of comprehensive and quality network datasets and the narrow scope to build an IDS based on a single machine learning classifier adds further limitations. Such issues can risk producing inaccurate or biased results in the solutions proposed by various researchers. Toward this end, this paper analyzed several datasets (old, recent, non-IoT, and IoT specific) using several individual and hybrid deep learning classifiers. Our goal is to establish a benchmark that can compare several classification models on several datasets to limit (1) dataset quality issues and (2) possible bias in produced results. We reported our empirical results by revealing exciting findings on some of the classifiers, which took hours to converge but could not successfully detect attacks. In contrast, others quickly converged and were able to produce the best results in terms of accuracy and other performance metrics. We believe that this paper's findings will help build a comprehensive IDS by recognizing that classification or prediction models should be trained beyond a limited scope of one dataset or application.

1. Introduction

Internet of Things (IoT) has been making a significant impact on people's lives. Governments and industries such as healthcare and transportation have generated immense value by vastly adopting these small devices to make critical, better, effective, and timely decisions. Healthcare is one of the critical and fastest-growing industrial sectors which produces a large amount of data through continuous patient monitoring, infrastructure management, surveillance cameras, record keeping, insurance records, and meeting compliance requirements. The digital transformation of connected devices in the healthcare industry is expected for the market to grow to $534.3 billion by 2025 globally (De Michele and Furini, 2019). It is critical to analyze massive data generated by IoT devices effectively to solve complex problems, make critical decisions, address new challenges, and prevent large-scale cyber-attacks. An estimated $300 to $450 billion can be saved in the healthcare industry by applying proper tools and technologies to process massive data, generate insights, and enhance security (Kayyali et al., 2013). Similarly, IoTs have revolutionized the transportation industry through autonomous vehicles, preventing accidents, controlling traffic patterns and congestion, and optimizing the supply chain process (Humayun et al., 2020). The market value of autonomous vehicles in 2019 is worth $54.23 billion and is expected to grow ten times to $556.67 billion by 2026 (Narla and Stowell, 2019).

Today's technology advances are driven by the fact that a massive amount of heterogeneous data is generated by billions of devices connected through the internet. Managing such large data without proper tools and techniques is a big challenge, and data may become useless if handled incorrectly. Besides all the benefits and projected growth in the IoT market, serious security concerns need attention to protect networks from vulnerabilities and various known and unknown cyber-attacks. IoT devices and networks are vulnerable to many security threats. Without a comprehensive solution to protect against known and zero-day attacks, critical issues such as data breaches, false diagnostics, fatal accidents due to equipment failure, and distributed denial of service (DDoS) attacks can happen. A DDoS attack would either shut down the critical services or make them unresponsive (Hady et al., 2020). Governments and organizations have been struggling to implement a comprehensive, secure, and efficient technology-based solution to deliver essential products and services; thus, the critical national infrastructure and networks remain at risk from potential cyber-attacks.

* Corresponding author.
E-mail address: rahmad4758@ucumberlands.edu (R. Ahmad).

https://doi.org/10.1016/j.cose.2021.102588
0167-4048/© 2021 Elsevier Ltd. All rights reserved.

The massive data generated by IoT devices attracted researchers to propose data-driven intrusion detection systems (IDS). The researchers mostly gather various publicly available network datasets or generate their own dataset in a small, controlled, and simulated environment to base their proposed IDS solution. Once the dataset is available, various traditional machine learning (ML) and deep learning (DL) algorithms are trained to build their models to predict traffic types such as benign or malicious. One of the difficulties in creating an effective intrusion detection system for IoT environments is the availability of a comprehensive network dataset that mirrors the current traffic patterns and includes various attacks and their variants. Over the past many years, institutions, organizations, and researchers have generated several datasets. Unfortunately, not all of them have been made available for public use due to privacy and security concerns. Due to the shortage of a reliable dataset, models suffer from inconsistent and inaccurate performance results. The dataset shortage forces researchers to (a) collect the new datasets of their own in a small simulated environment by launching a handful of known attack types in a controlled manner, (b) propose a solution by limiting their investigations on a single publicly available dataset, or (c) propose a single machine learning classifier for an IDS.

The literature review presented in Section 2 shows that researchers generally pick a single model and a dataset to propose an IDS solution and report performance metrics. One of the problems with a single datasets approach is that it produces biased results by focusing only on limited network traffic patterns and minimal attack classes. The same is the case when researchers base their analysis only on a single machine learning classifier. It lacks diversity; a classifier may perform well on a single dataset but switching the dataset to a new network dataset yields a biased result, and the performance drops considerably. Other problems in single dataset or classifier approaches include: (a) researchers spend time to tune their proposed classifier for a specific dataset which is either an old dataset that lacks the modern-day traffic and attack patterns. (b) The proposed IoT IDS model is trained on a non-IoT dataset such as NSL-KDD and KDD CUP 99; these models are prone to failure in a live IoT environment with different traffic patterns attack information. (c) Build the models using traditional machine learning techniques instead of deep learning algorithms. A common challenge in the traditional machine learning approach is the complex feature selection process which requires in-depth network traffic understanding and is labor and time-intensive (Ahmad and Alsmadi, 2021). Feature selection and model training are independent processes and cannot be performed jointly for optimized classifier performance (Liang and Znati, 2019). Our approach in this study is to analyze several datasets (old, recent, non-IoT, and IoT specific) using several individual and hybrid deep learning models. We captured several pieces of valuable information such as training parameters, training time, model settings, and performance metrics to identify several datasets and deep learning algorithm combinations. This approach helps to narrow down what is good and weak in datasets and classifiers. We hope that our analysis will help future authors gain insight and guidance on what algorithms perform well on a specific dataset and what does not. It will also help future researchers to build a comprehensive IDS based on deep learning models for IoTs.

The rest of the paper is organized as follows. Section 2 presents several publicly available network intrusion benchmark datasets, the background of several deep learning models commonly adopted by the researchers, and different intrusion detection techniques in network traffic. In Section 3, we present our adopted methodology and provide detail of several deep learning algorithms used. Section 4 presents a comprehensive analysis of our experiments and their results. Section 5 discusses the results, trends, and future assessments. Finally, Section 6 concludes the paper and provides our approach going forward.

2. Literature review

The growth of IoT devices is unprecedented. The projected number of IoT devices in 2025 is 38.6 billion, reaching 50 billion by 2030 (Karie et al., 2020). IoT devices' ability to gather and communicate data makes them very powerful but vulnerable to cyber threats. Securing network traffic in such a large number of IoT devices has been a challenge for manufacturers and researchers. Unfortunately, there is no single comprehensive solution or model available to protect these devices worldwide. End-to-end security architecture with new solutions is required to prevent zero-day attacks and protect the complete IoT infrastructure (Aydos et al., 2019). IoT networks are vulnerable to several known and unknown attack types. According to Cisco's annual internet report, there will be 15.4 million global DDoS attacks in 2023, double the numbers 7.9 million in 2018 ("Cisco annual internet report (2018–2023) white paper," 2020). Large-scale attacks like DDoS not only bring loss to businesses, but a failure of critical life-saving medical and industrial equipment can also put millions of lives at risk. Both Denial-of-Service (DoS) and Distributed Denial-of-Service (DDoS) attacks are lethal with similar motives of flooding and bringing critical services and infrastructure down. However, DDoS attacks cause broader damage and are difficult to prevent because cyber-attackers can use command and control (C&C) servers to control thousands and possibly millions of compromised IoT devices (a.k.a. bots) to simultaneously launch an attack and flood the server resources, making it unresponsive. Some DDoS attacks like "Slowloris" attacks slowly deplete server resources and are hard to detect (Shorey et al., 2018), whereas others like "Jamming" attacks can quickly drain resources and have serious consequences (Fadele et al., 2019).

Dataset has a vital role in building and testing an IDS. Deep learning models need a large volume of data for better classification and improved performance. The lack of availability of a comprehensive and recent network dataset forces institutions and researchers to either rely on existing available datasets or generate their own datasets in a simulated environment. These approaches have several limitations, including (a) lack of attack classes captured in a single dataset, (b) datasets size not being large enough to cover a wide range of traffic patterns, (c) models trained on a single dataset produce bias results, limitations and restrictions in model training (Feng et al., 2019), and (d) not capturing new attack variants getting generated constantly on a daily basis. For example, In 2016, one of the significant large-scale attacks, "Mirai," happened due to default and hardcoded credentials in IoT devices (Kelly et al., 2020). Mirai launched a 665 Gbps DDoS attack infecting over 2.5 million IoT devices (Jaidka et al., 2020). Mirai's variants are still a serious threat. As of July 2019, there were sixty-three Mirai-like variants discovered using different payloads and extending the attack surface (DeBeck et al., 2019). The following section presents the details of various publicly available datasets commonly and recently adopted by the researchers to build their IDS for IoTs.

2.1. Benchmark datasets

The datasets presented below are chosen based on their novelty, attack diversity, and common utilization for IDS by the researchers. Some of these datasets are old and lack recent traffic patterns and attack types, whereas others were generated recently to overcome the old dataset's limitations. Section 4 presents our experiments and findings on these datasets by applying various deep learning classifiers.

Table 1
BoT-IoT dataset.

Class	Total	Class%age	Training Set	%age	Test Set	%age
DoS	1,650,260	44.984%	1,155,182	31.489%	495,078	13.495%
DDoS	1,926,624	52.518%	1,348,637	36.762%	577,987	15.755%
Reconnaissance	91,082	2.483%	63,757	1.738%	27,325	0.745%
Theft	79	0.002%	55	0.002%	24	0.001%
Normal	477	0.013%	334	0.009%	143	0.004%
Total	3,668,522		2,567,965	70%	1,100,557	30%

Table 2
N_BaIoT dataset.

Device Type	Benign	Gafgyt Attack	Mirai Attack	Total Attacks
Danmini Doorbell	49,548	316,650	652,100	968,750
Ecobee Thermostat	13,113	310,630	512,133	822,763
Ennio Doorbell	39,100	316,400	0	316,400
Philips B120N10 Baby Monitor	175,240	312,723	610,714	923,437
Provision PT 737E Security Camera	62,154	330,096	436,010	766,106
Provision PT 838 Security Camera	98,514	309,040	429,337	738,377
Samsung SNH 1011 N Webcam	52,150	323,072	0	323,072
SimpleHome XCS7 1002 WHT Security Camera	46,585	303,223	513,248	816,471
SimpleHome XCS7 1003 WHT Security Camera	19,528	316,440	514,860	831,300
Total	555,932	2,838,274	3,668,402	6,506,676

2.1.1. *BoT-IoT*

This dataset was designed and collected in 2018 by the University of New South Wales (UNSW) Center for Cyber Security (ACCS) Canberra, Australia (in the Cyber Range Labs) (Koroniotis et al., 2019). The dataset consists of both simulated and real IoT network traffic. It captures stateful traffic information in multiple file formats (CSV, argus, p-cap) and provides corresponding labels. Dataset is highly imbalanced with very few benign and a large volume of malicious records. The full dataset consists of 73,360,900 rows and has a size of over 69 GB. UNSW provides a scaled-down 5% dataset to make data handling easy. Table 1 shows the breakdown of the number of rows in each class in the 5% dataset used in our study.

2.1.2. *N_BaIoT*

This dataset was published and provided in 2018 by the UCI Center of Machine Learning and Intelligence Systems at the University of California, Irvine (Meidan et al., 2018). The dataset consists of real IoT network traffic collected from 9 commercial IoT devices. They infected the dataset with two famous and harmful IoT botnets Mirai and BASHLITE (a.k.a. Gafgyt). Dataset provides 115 statistical traffic features in a CSV format. It captures ten attack classes: Gafgyt combo, Gafgyt junk, Gafgyt scan, Gafgyt TCP, Gafgyt UDP, Mirai ack, Mirai scan, Mirai syn, Mirai UDP, and mirai_udpplain. Dataset is an imbalance with very few benign and a large volume of malicious records (Alsamiri and Alsubhi, 2019). Table 2 provides the detail of this dataset.

Due to the resource limitations, we used only the "Danmini Doorbell" in our experiments. The breakdown of classes for this device is shown in Table 3.

2.1.3. *CICIDS2017*

This dataset was developed in 2017 by the Canadian Institute for Cybersecurity at the University of New Brunswick (UNB), Canada (Sharafaldin et al., 2018). Dataset is a non-IoT dataset (Chaabouni et al., 2019) but contains several large-scale attacks common to every network environment and are part of current attack types such as DDoS, DoS, brute force attack, botnet, Heartbleed, and web attacks. It also contains several complex features that are unavailable in previous datasets, such as NSL-KDD. The dataset captures five days of network traffic. Monday's traffic only consists of normal traffic. The reset of the days consists of attack traffic such as Brute Force, Botnet, DDoS, DoS, Heartbleed, Web Attack, Infiltration, and the standard protocols such as HTTP HTTPS, SSH, FTP, and email protocols (Binbusayyis and Vaiyapuri, 2019). They generated the dataset using CICFlowMeter to capture bidirectional traffic flow (source to destination and destination to source). It provides the stateful traffic information in CSV and p-cap file formats with corresponding labels. The dataset consists of 83 features and captures 14 attacks. The dataset is highly imbalanced (Mera and Branch, 2014) and is prone to generate biased results towards the majority classes with poor generalization (Wang et al., 2018). Table 4 provides the detail of this dataset.

2.1.4. *UNSW-NB15*

This dataset was designed and collected in 2015 by the University of New South Wales (UNSW) Center for Cyber Security (ACCS) Canberra, Australia (in the Cyber Range Labs) (Moustafa and Slay, 2015, p. 15). This dataset is not IoT specific and is generated by collecting the real benign network traffic and synthetically generated attacks. It captures stateful traffic information in multiple file formats (CSV, argus, BRO, p-cap) and provides corresponding labels. The dataset captures nine attack categories and is available to download in full in four CSV files, namely UNSW-NB15_1.csv, UNSW-NB15_2.csv, UNSW-NB15_3.csv, and UNSW-NB15_4.csv; or in smaller filesizes namely, UNSW_NB15_training-set.csv and UNSW_NB15_testing-set.csv. The overall classification accuracy has a mitigating effect on this dataset; it is due to the greater number of classes (10 in NB15 vs. 5 in KDD) and a higher Null Error Rate (55.06% in NB15 vs. 26.1% in KDD) (Divekar et al., 2018). Table 5 shows the breakdown of classes in each of the dataset files. We trained our classifiers separately on the full and partial datasets to compare the performance measures in our experiments.

2.1.5. *NSL_KDD*

This dataset was developed in 2009 by the Canadian Institute for Cybersecurity at the University of New Brunswick (UNB), Canada. Dataset is a non-IoT dataset created to overcome some of the previous dataset (KDD CUP 99) limitations, such as duplicate and unbalanced classification (Soe et al., 2019; Chaabouni et al., 2019; Tavallaee et al., 2009). Eliminating these limitations enhances this dataset to produce an unbiased classifier and reduce false-positive results. It is an old dataset and lacks current at-

Table 3
N_BaIoT dataset (Danmini Doorbell).

Class	Total	Class%age	Training Set	%age	Test Set	%age
Gafgyt combo	59,718	5.864%	41,803	4.105%	17,915	1.759%
Gafgyt junk	29,068	2.855%	20,348	1.998%	8720	0.856%
Gafgyt scan	29,849	2.931%	20,894	2.052%	8955	0.879%
Gafgyt tcp	92,141	9.049%	64,499	6.334%	27,642	2.715%
Gafgyt udp	105,874	10.397%	74,112	7.278%	31,762	3.119%
Mirai ack	102,195	10.036%	71,536	7.025%	30,659	3.011%
Mirai scan	107,685	10.575%	75,379	7.402%	32,306	3.173%
Mirai syn	122,573	12.037%	85,801	8.426%	36,772	3.611%
Mirai udp	237,665	23.339%	166,365	16.338%	71,300	7.002%
Mirai udpplain	81,982	8.051%	57,387	5.636%	24,595	2.415%
Normal	49,548	4.866%	34,684	3.406%	14,864	1.460%
Total	1,018,298		712,808	70%	305,490	30%

Table 4
CICIDS2017 Dataset.

Class	Total	Class%age	Training Set	%age	Test Set	%age
DoS Hulk	230,124	8.138%	161,087	5.696%	69,037	2.441%
PortScan	158,804	5.616%	111,163	3.931%	47,641	1.685%
DDoS	128,025	4.527%	89,618	3.169%	38,408	1.358%
DoS GoldenEye	10,293	0.364%	7205	0.255%	3088	0.109%
FTP-Patator	7935	0.281%	5555	0.196%	2381	0.084%
SSH-Patator	5897	0.209%	4128	0.146%	1769	0.063%
DoS slowloris	5796	0.205%	4057	0.143%	1739	0.061%
DoS Slowhttptest	5499	0.194%	3849	0.136%	1650	0.058%
Bot	1956	0.069%	1369	0.048%	587	0.021%
Web Attack Brute Force	1507	0.053%	1055	0.037%	452	0.016%
Web Attack XSS	652	0.023%	456	0.016%	196	0.007%
Infiltration	36	0.001%	25	0.001%	11	0.000%
Web Attack Sql Injection	21	0.001%	15	0.001%	6	0.000%
Heartbleed	11	0.000%	8	0.000%	3	0.000%
BENIGN	2,271,320	80.319%	1,589,924	56.223%	681,396	24.096%
Total	2,827,876		1,979,513	70%	848,363	30%

Table 5
UNSW-NB15 dataset.

	Partial Dataset					Full Dataset				
Class	Total	Training Set	%age	Test Set	%age	Total	Training Set	%age	Test Set	%age
Generic	58,871	18,871	7.324%	40,000	15.524%	215,481	150,837	5.938%	64,644	2.545%
Exploits	44,525	11,132	4.320%	33,393	12.959%	44,525	31,167	1.227%	13,358	0.526%
Fuzzers	24,246	6062	2.353%	18,184	7.057%	24,246	16,972	0.668%	7274	0.286%
DoS	16,353	4089	1.587%	12,264	4.760%	16,353	11,447	0.451%	4906	0.193%
Reconnaissance	13,987	3496	1.357%	10,491	4.071%	13,987	9791	0.385%	4196	0.165%
Analysis	2677	677	0.263%	2000	0.776%	2677	1874	0.074%	803	0.032%
Backdoor	2329	583	0.226%	1746	0.678%	2329	1630	0.064%	699	0.028%
Shellcode	1511	378	0.147%	1133	0.440%	1511	1058	0.042%	453	0.018%
Worms	174	44	0.017%	130	0.050%	174	122	0.005%	52	0.002%
Normal	93,000	37,000	14.359%	56,000	21.733%	2,218,764	1,553,134	61.146%	665,630	26.205%
Total	257,673	82,332	32%	175,341	68%	2,540,047	1,778,032	70%	762,015	30%

tack types. Dataset does not provide basic packet-level information. Dataset provides separate training and test data files with a training dataset consisting of 22 attack types and a test dataset consisting of 37 attack types. These attack types encompass four main classes, namely Denial of Service (DoS) attack, Probe attack, Remote to Local (R2L) attack, and User to Root (U2R) attack (Ingre and Yadav, 2015). Table 6 provides the detail of this dataset.

2.1.6. KDD CUP 99

Several IDS studies have been performed using KDD CUP 99 dataset; however, many researchers question the dataset's credibility due to its skewed response distribution, non-stationarity, and lack of recent attacks (Divekar et al., 2018). The dataset was published in 1999 by MIT University's Lincoln Laboratories and simulating a US Air Force LAN environment. It consists of two weeks of normal and five weeks of attack traffic. It consists of 42 features, and the attack types broadly classify into four attack classes, namely Denial of Service (DoS) attack, Probe attack, Remote to Local (R2L) attack, and User to Root (U2R) attack. The number of attack types is different in the training and test dataset. The training dataset consists of 24 attack types, whereas the test dataset consists of 14 new attack types, which does not exist in the training dataset. Dataset is highly unbalanced with a large number of duplicate records (Soe et al., 2019; Chaabouni et al., 2019; Tavallaee et al., 2009). Excessive duplication in records leads to skewed label distribution, and the classifier generates biased results towards most occurring records and cannot thoroughly learn the least occurring records (McHugh, 2000). This dataset is not specific to the IoT environment. Table 7 provides the detail of this dataset.

2.2. IDS approaches

Anomaly detection is a vast evolving field where researchers have proposed many approaches to detect network attacks. However, there is no single IDS approach that can meet the require-

Table 6
NSL-KDD dataset.

Label	Attack type	Total	Training data	%age	Testing data	%age
DoS	Snmpgetattack, back, land, Neptune, smurf, teardrop, pod, apache2, Udpstorm, process table, mailbomb	53,563	45,927	30.924%	7,636	5.141%
Probe	IP-sweep, Nmap, Portsweep, satan, saint, m-scan	14,077	11,656	7.848%	2,421	1.630%
R2L	Snmpguess, worm, HTTP tunnel, named, x-lock, x-snoop, send mail, ftp_write, guess_passwd, IMAP, multihop, PHF, spy, Warezclient, Warezmaster	3,704	995	0.670%	2,709	1.824%
U2R	Sqlattack, buffer_overflow, Loadmodule, Perl, rootkit, Xterm, ps	119	52	0.035%	67	0.045%
Normal	Benign traffic	77,054	67,343	45.344%	9,711	6.539%
Total		148,517	125,973	85%		15%

Table 7
KDD CUP 99 dataset.

Class	Sub-classes	Total	Training Set	%age	Test Set	%age
Dos	back, land, Neptune, pod, smurf, teardrop.	391,458	274,020	55.467%	117,438	23.772%
Probe	ftp_write, guess_passwd, imap, multihop, phf, spy, warezclient, warezmaster.	4,107	2,875	0.582%	1,232	0.249%
R2L	ipsweep, nmap, portsweep, satan	1,126	788	0.160%	338	0.068%
U2R	buffer_overflow, loadmodule, perl, rootkit	52	36	0.007%	16	0.003%
Normal		97,278	68,095	13.784%	29,183	5.907%
Total		494,021	345,814	70%	148,207	30%

ments of detecting all attacks (Elejla et al., 2018). The two common approaches are signature-based (a.k.a. misused-base), which depends on the existing threat repository, and anomaly-based (a.k.a. behavior-based), which analyzes network traffic to detect malicious activities. This section presents some of the common techniques that researchers adopted within anomaly-based approaches as these approaches are commonly adopted and outperform signature-based approaches in detecting unknown attacks.

2.2.1. *Offline approaches*

These approaches are mostly data-driven and are good to generate early warnings. Data-driven strategies typically consist of multiple phases such as (a) capturing incoming and outgoing network traffic packets or using an existing benchmark network dataset, (b) filtering redundancies and noise, (c) selecting features most relevant to attack detection goals (d) constructing flow-based features (e.g., source IP, destination IP, source Port, destination Port, and protocol) from the captured packets (e) dividing the dataset into training and test set (f) building ML-based models to predict normal and malicious traffic (g) model deployment, monitoring and analysis (h) defense response by a software or a security expert. Some of the approaches within offline approaches are the following:

2.2.1.1 Threshold-based. A widely used and broader approach within anomaly detection is a threshold-based approach. In this approach, the model is generally trained on normal traffic instances, and a threshold-value (a.k.a. anomaly score) is calculated from the predicted scores. During testing, a new instance exceeding the predefined threshold value is classified as an attack. In Aygun and Yavuz (2017), the authors trained autoencoder on normal data only to establish a threshold value. In Charyyev and Gunes (2020), the author's calculated threshold values using average hash values. A new instance with a hash value greater than the threshold value is classified as an attack during testing. In another unsupervised approach, the authors used adversarial autoencoders to generate fake attack data. They calculated the ii-loss threshold value to identify new instances as an attack if their distance is far from the cluster mean (Hassen and Chan, 2020). Different researchers adopted different approaches to set threshold values, e.g., in Said Elsayed et al. (2020), the researchers computed the $\ell 2$-norm threshold value using the difference between original and output features as shown in Eq. (1). In Hassen and Chan (2020), the authors used autoencoder to compute reconstruction loss, as shown in Eq. (2). In Meidan et al. (2018), the authors used two datasets to calculate mean square error (MSE) as the threshold value. The authors used DS_{trn} (training set) and DS_{opt} (optimized parameters dataset) to compute the threshold value (tr^*) as shown in Eq. (3).

$$\ell 2 - norm\ error = \left|\left|X_t - \hat{X}_t\right|\right|^2 \tag{1}$$

$$Reconstruction\_loss = \frac{1}{2N}\sum_{i=1}^{N}\left|\left|\hat{\vec{x}}_i - \vec{x}_i\right|\right|_2^2 \tag{2}$$

$$tr^* = \overline{MSE}_{DS_{opt}} + s\left(MSE_{DS_{opt}}\right) \tag{3}$$

Although a threshold-based approach provides a good solution for distinguishing normal and malicious traffic, it cannot detect different attack types precisely. Some researchers reported limitations in the threshold-based approach. For example, in Said Elsayed et al. (2020), the authors stated that a threshold-based approach is insufficient. An attacker can use sophisticated tools to easily generate malicious traffic similar to normal traffic, making it difficult for

IDS to detect malicious traffic accurately. Another author argued that identifying threshold values based on the softmax function is problematic and may produce biased results for certain classes (Dhamija et al., 2018). Lastly, keeping a constant threshold value has significant limitations in changing network traffic and conditions (Haider et al., 2020).

2.2.1.2. Threat-driven. Due to different malware classes, some researchers adopt a threat-based approach to separately deal with individual attack classes. The rationale for this approach is that detecting network worms is different from detecting drive-by-downloads. Similarly, detecting DDoS is different from detecting Heartbleed or SQL injection attacks. DDoS and DOS are volumetric attacks (with high bandwidth) typically launched against a specific target. Researchers typically analyze traffic flow information instead of packets to detect DDoS attacks (Elejla et al., 2018).

In Elejla et al. (2018), the authors proposed a DDoS detection approach using ICMPv6 flow information. The authors argued that ICMPv6 is an OSI network layer protocol that works without ports in the message delivery. Due to only evaluating ICMPv6, the authors eliminated the protocol information from the flow information. Thus, the authors initially only used source and destination IP addresses to form a flow to detect DDoS attacks and later added a few more features based on empirical analysis and their experience in the field. The authors reported that the IDS model accuracy on packet data is much lower compared to flow-based traffic; hence a flow-based approach is a right approach to detect DDoS attacks. In Roopak et al. (2020a), the authors proposed an IDS to detect DDoS attacks based on traffic flow-based dataset CICIDS2017 and feature engineering method NSGA-II. In Shurman et al. (2020), the authors proposed two approaches to detect DoS and DDOS attacks in IoT. In the first approach, the author simulated network data and combined signature-based IDS and anomaly-based IDS to obtain the benefit of both approaches for attack detection. In the second approach, the authors built an LSTM model to detect DDoS and DoS attacks. Another author (Haider et al., 2020) proposed a CNN-based ensemble approach to detect DDoS attacks in SDN by using a flow-based CICIDS2017 dataset.

In Malik et al. (2020), the authors proposed an approach to detect three Reconnaissance attacks (Bot, Port Scan, and XSS). Reconnaissance attacks are entirely different from DDoS and DoS attacks. It is an initial phase of launching any other malicious activities. The adversaries in this attack generally gather the knowledge of the target systems through social engineering, port scans, phishing, and packet sniffing (C.-Y. Chen et al., 2017). The authors built a hybrid deep learning model using LSTM and CNN on the CICIDS2017 dataset and reported a high attack detection rate. In Zhou et al. (2020), the authors proposed two approaches to detect worms by analyzing network payload and worms' signatures in a synthetically generated environment. The authors used CNN to detect worms and DNN to generate worm signatures from the worm payloads.

Threat-driven IDS are good for detecting a specific network attack, but they are basic detectors that miss the real-world IDS standards. For example, a tailored IDS solution to detect volumetric DDoS attacks would miss attacks other than DDoS (e.g., worms) (Mergendahl and Li, 2020).

2.2.1.3. Multi-attack based. Another commonly used approach for IDS is multiclass classification. In this approach, researchers first select one or more datasets consisting of numerous attack classes and then train the classifier to detect each attack type optimally. For example, the authors Samy et al. (2020) proposed a distributed IoT attack detection framework at a fog layer. The authors trained six DL models at the cloud on five different benchmark datasets. Each dataset consists of different attacks, feature space and is collected at different times by different organizations. The authors reported model performance against each attack type in each dataset. In Moustafa et al. (2019), the authors analyzed network flow information and proposed an AdaBoost ensemble to detect various IoT attacks in MQTT, DNS, and HTTP protocols. The authors extracted essential features using correlation coefficients from benchmark UNSW-NB15 and NIMS datasets. The authors argued that correntropy is a significant measure to differentiate between normal and malicious traffic. They selected individual ensemble members (decision tree, naïve Bayes, and artificial neural network) to classify the traffic with minor correntropy differences. The proposed ensemble approach can detect nine target classes in the UNSW-NB15 dataset. In another study Ge et al. (2019), the authors develop a feed-forward neural network to classify different IoT attacks by analyzing network packets. The proposed model could detect five target classes in the BoT_IoT dataset. In Ferrag et al. (2020), the authors develop a hybrid IDS by combining multiple tree-based classifiers to detect numerous attacks in IoT. The models are built using benchmark CICIDS2017 and BoT_IoT datasets that have twenty target classes altogether. Their approach creates a hierarchical classifiers approach where two classifiers run in parallel and feed into third classifiers, which produces the final classification of traffic as benign or attack.

2.2.2. Online approaches

Threat-based IDS approaches provide good and tailored solutions towards a particular attack type. However, some researchers argued that threat-based IDS solutions that require flow-based network data (e.g., IDS for DDoS detection) have limitations in real-time deployments. Some of them include (a) they require a lot of computational time to gather packets in a particular flow (especially for longer sessions) before extracting important features, (b) they require a lot of resources, e.g., memory and storage, to accumulate traffic. These limitations increase attack detection time. To mitigate these limitations, the authors suggested parallel processing to inspect packet-level traffic with an accelerated detection process for classification (Hwang et al., 2019). Some other authors argued that online IDS approaches are useful in resource-constrained IoT environments with limited computational power and memory resources (Mirsky et al., 2018).

In Hwang et al. (2019), the authors proposed an LSTM and word embedding-based approach to inspecting semantics in packet-level traffic to detect large-scale attacks. They Used public datasets (ISCX2012, USTC-TFC2016) and custom honeypot datasets (Mirai-RGU, Mirai-CCU) to convert traffic into normal and attack without any preprocessing for performance evaluation. The authors claimed that their approach reduces computational time compared to flow-based IDS. In another study, Hwang et al. (2020), the authors proposed an unsupervised online IDS based on CNN and autoencoder. The authors argued that manual feature extraction is time-consuming for diverse network traffic; instead, they used CNN to auto-learn the features by looking at the first few bytes of raw traffic. The authors trained the autoencoder on benign traffic and used MSELoss as the threshold value to detect attacks. Another research (Mergendahl and Li, 2020) proposed an online IDS for DDoS detection only. Their solution requires little to no human intervention by adopting the LSTM algorithm. The authors argued that offline trained models generate high false-positive results under a domain shift and require extensive manual data labeling by the security professional. The author stated that their solution is only tailored to detecting volumetric DDoS attacks and is not suitable for other attack classes such as worms. Their solution gathers aggregated packet flow features and uses a statistical approach to find flow severity.

2.3. Related work

Traditional perimeter security features like firewalls, antivirus, and software patches are not adequate and are becoming weaker with the growing number of IoT devices and their associated vulnerabilities (Anthi et al., 2019). IoT systems operate and behave differently than traditional computer systems. A significantly large number of applications and types of IoT devices make their normal behavior different from conventional computing devices. The two main methods to detect the network intrusions and alerting the security teams are the Signature-based (a.k.a. misuse-based) IDS, which depends on existing threat knowledge, and the Anomaly-based (a.k.a. behavior-based) IDS, which rely on network traffic behavior. Sometimes they are combined and used as a Hybrid approach or as a Stateful Protocol-based approach that stores the characteristics of benign traffic activities. Although these network intrusion detection systems (NIDS) provide good security features, they also contain some weaknesses (Ahmad and Alsmadi, 2021). For example, signature-based network intrusion detections (NIDS) systems are ineffective in detecting zero-day attacks because they rely on prior attack knowledge from the attack signatures. Anomaly-based security systems are also not entirely accurate in detecting intrusion and may return false-positive (García-Teodoro et al., 2009). The hybrid approach is too complex to implement. Lastly, the Stateful Protocol approach depends on vendor profiles and cannot effectively inspect attacks if they use benign network traffic behavior (Sarker et al., 2020).

Researchers have been extensively studying anomaly-based IDS solutions by building traditional ML and DL models. Because of the computational limitations (Ng et al., 2019) and manual feature engineering constraints in traditional ML techniques, we present only IDS solutions implemented using DL in this study. Ma et al. (2020) proposed a feature fusion mechanism using Convolutional Neural Network (CNN) and developed a Symmetric logarithmic loss function based on categorical cross-entropy, SLL, and CCE. The authors have used the NSL-KDD dataset and achieved an accuracy of 92.99% and a detection rate of 82.62%. Liang and Znati (2019) proposed a Long Short-term Memory Network (LSTM) based IDS solution using the CICIDS2017 dataset. The authors inspected the stateful traffic by focusing only on detecting DOS and DDoS attacks. Their approach is prone to large false-positive classifications in the testing phase because their training only involves a single day's data. Not training the model on other four days of data would limit the model's ability to generalize traffic patterns from the entire dataset. Haider et al. (2020) proposed hybrid classifier solutions using CNN+CNN, RNN+RNN, LSTM+LSTM, and RNN+LSTM. The authors based their models on the CICIDS2017 dataset and provided binary classification to detect DDoS attacks only. Sriram et al. (2020) proposed a Deep Neural Network (DNN) based IDS solution. They trained their model on two IoT datasets, Bot-IoT and N_BaIoT. The authors divided the dataset into three categories, i.e., benign and attack, Bashlite and Mirai, and attack subclasses. t-SNE is used to plot data on low-dimensionality. Ge et al. (2019) build their Feed-forward Artificial Neural Network (ANN) based on the BoT-IoT dataset. The authors excluded features that contained aggregated information and only focused on individual packet information. The authors manually extracted features and labels based on personal judgment. Das et al. (2019) implemented an ensemble approach by training the classifiers parallel to different data subsets. The researchers only performed binary classification to detect DDoS attacks using the NSL-KDD dataset.

3. Proposed methodology

The research work above reflects recent research studies on IDS. Researchers trained their classifiers only on a single dataset and on a single or hybrid classifier, trying to find the optimal IDS solution. Such models are not only prone to return biased results in a live environment but may return high false-positive results if trained on very old datasets such as NSL-KDD or KDD CUP 99 (Das et al., 2019). Our methodology in this study analyzes several benchmark datasets using several individuals and hybrid deep learning models. The detail of datasets used is presented in Section 2.1 above. Firstly, we implemented various individual DL classifiers such as Multi-layer Perceptron (MLP), CNN, and LSTM on each of the datasets. Secondly, we implemented hybrid classifiers by combining Autoencoder with classifiers commonly used in sequential and time-series data. For the hybrid approach, we implemented following algorithms (a) autoencoder + Temporal Convolutional Network (TCN), (b) Autoencoder + LSTM, (c) Autoencoder + Bi-Directional Recurrent Neural Network (BRNN), (d) Autoencoder + Bidirectional LSTM (BLSTM), and (e) CNN + LSTM.

Fig. 1 primarily shows the architecture of our hybrid model "autoencoder + TCN"; however, the same architecture has been used for other hybrid implementations mentioned above. In other implementations, we replaced the "TCN Structure" block in Fig. 1 with LSTM, BRNN, or BLSTM. Model-specific settings are reported in Table 8, 9 and 10 (individual classifiers) and in Table 14, 15, 16, 17 and 18 (hybrid classifiers). Firstly, the Autoencoder takes the input shape of the dataset, reduces its dimensions to a smaller size (a.k.a. latent space), and then reconstructs the original data from the compressed representation. Autoencoders generate reconstruction errors and guarantees very high accuracy with low latency detection (Naveed and Wu, 2020). The output of autoencoders is passed into a TCN structure. The pooled outputs of TCN block layers are flattened into a 1-dimensional array and are passed to a fully connected layer. The last layer is passed with total output classes using the "*softmax*" activation function to make predictions on each label.

TCN block in Fig. 1 is the layered implementation of the dilated causal convolution of our TCN architecture. Fig. 2 reflects the details of this model. TCN block takes three parameters as follows:

$$TCNblock(filters,\ kernel\_size,\ dialation\_rate)$$

Where "*filters*" are similar to units in LSTM, they affect the model size. Larger filter size is preferred. It helps to train the model in parallel and faster, unlike RNN and LSTM, where predictions must wait for the predecessor results (Bai et al., 2018). In TCN, a longer input sequence can be processed as a whole instead of sequentially. "*kernel_size*" parameter determines the size of each filter in each convolution layer. It helps to calculate how much of the input is used to calculate each value in the output. Larger kernel_size helps detect complex features. "*dialation_rate*" parameter represents a fixed step between two adjacent filters. A "$dilation\_size = 1''$ is the same as a regular convolution in a CNN network. A larger dilation rate captures a bigger input range on the top convolution layer, making a TCN more receptive (Bai et al., 2018). Our TCN structure first starts with a CONV1D dilated causal convolution layer, followed by a batch normalization layer to obtain high accuracy values and increase model training. We then implemented a Rectified Linear Unit *"Relu"* to allow quick network convergence. We then added a "*Dropout*" layer to avoid overfitting and add regularization by randomly dropping 30% of weights connected to certain Nodes during the training process. The same layers structure was repeated in the TCN block and is shown in Fig. 2.

3.1. Rational for classifiers selection

Many studies performed on network anomaly detection by utilizing a vast array of built-in and customized DL models. Network data is sequential, so we shortlisted and extracted the models from previous studies known for processing sequential data. The re-

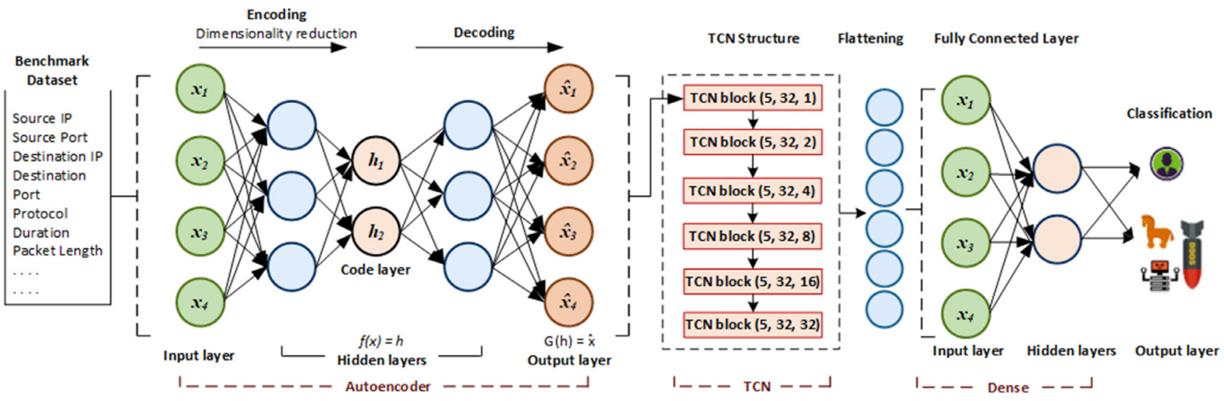

Fig. 1. Autoenoder + TCN architecture.

Table 8
Multi-layer perceptron (MLP).

Dataset	Trainable Parameters	epoch	Batch Size	Training Time	Model Size	Accuracy
Bot-IoT	32,581	20	256	7 min	434 KB	100.00%
CICIDS2017	38,255	20	256	5 min	500 KB	99.90%
NSL-KDD [*a]	43,205	100	256	1 min	193 KB	77.30%
NSL-KDD [*b]	43,205	20	256	1 min	560 KB	98.90%
UNSW_NB15 [*a]	52,890	20	256	1 min	673 KB	38.90%
UNSW_NB15 [*c]	53,786	20	256	5 min	684 KB	97.80%
N-BaIoT	42,411	20	256	2 min	550 KB	90.80%
KDD CUP 99	42,693	20	256	1 min	553 KB	99.90%

Table 9
Convolutional neural network (CNN).

Dataset	Trainable Parameters	epoch	Batch Size	Training Time	Model Size	Accuracy
Bot-IoT	12,901	20	256	11 min	198 KB	100.00%
CICIDS2017	18,495	20	256	9 min	264 KB	100.00%
NSL-KDD [*a]	23,525	100	256	15 min	322 KB	74.50%
NSL-KDD [*b]	23,525	20	256	1 min	322 KB	98.80%
UNSW_NB15 [*a]	33,170	20	256	1 min	436 KB	37.30%
UNSW_NB15 [*c]	34,066	20	256	15 min	447 KB	97.80%
N-BaIoT	22,683	20	256	5 min	313 KB	90.90%
KDD CUP 99	23,013	20	256	2 min	316 KB	99.90%

Table 10
Long short-term memory (LSTM).

Dataset	Trainable Parameters	epoch	Batch Size	Training Time	Model Size	Accuracy
Bot-IoT	3,100,261	20	256	120 min	36 MB	100.00%
CICIDS2017	3,188,495	20	256	185 min	37 MB	99.90%
NSL-KDD [*a]	3,270,245	100	256	113 min	38 MB	75.10%
NSL-KDD [*b]	3,270,245	20	256	15 min	38 MB	98.60%
UNSW_NB15 [*a]	3,423,930	20	256	23 min	39 MB	57.40%
UNSW_NB15 [*c]	3,438,266	20	256	532 min	39 MB	98.00%
N-BaIoT	3,256,011	20	256	121 min	37 MB	90.90%
KDD CUP 99	3,262,053	20	256	47 min	37 MB	99.90%

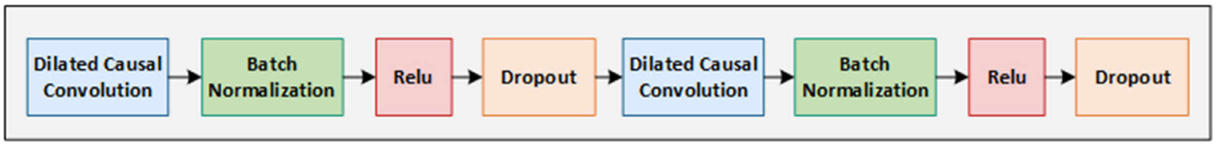

Fig. 2. TCN block architecture.

searchers have returned good attack detection results. The following sub-sections provide a brief background of the different models used in our experimental study.

3.1.1. *Multilayer perceptron (MLP)*

A feed-forward network, a.k.a. multilayer perceptron (MLP) or deep neural network (DNN), consists of a collection of neurons with three fundamental layers, namely (a) input layer, (b) hidden layers, and (c) output layer (Nagisetty and Gupta, 2019). At each perceptron in the hidden layer, the previous layer's input is multiplied by a weight, added to a bias, and passed through an activation function. In the feed-forward networks, there is no backward or loop connection; information only flows forward. The input layer accepts the data, which then passes through one or more hidden layers in sequence until it reaches the output layer. They use backpropagation to update the weights until it achieves a desirable performance iteratively. MLP has frequently been used by researchers for anomaly detection in network traffic (Das et al., 2019; Nagisetty and Gupta, 2019; Lai et al., 2019; Liu et al., 2019; Mergendahl and Li, 2020).

3.1.2. *Convolutional neural network (CNN)*

CNN's are primarily used for image classification and are known to produce high accuracies when performing complex tasks (Ahmad and Alsmadi, 2021). A typical CNN consists of one or more convolutional, pooling, and fully connected layers. 1D-CNN has been used in many IDS studies and produces good results in processing sequential data (Hwang et al., 2020). CNN's are computationally expensive when trying to get higher accuracies (Liang et al., 2020). CNN consists of multiple layers with multiple identical neuron copies. The input feeds into a convolutional layer to extract features that feed into the pooling layer. Reducing unimportant features (dimensionality reduction) helps minimize overfitting by preserving the full information. The next is the flattening layer, which converts the pooled layer into a 1-dimensional array to feed into a successive, fully connected layer. The output of the flattening layer feeds into a fully connected layer for the final classification output. CNN's main advantage over a feed-forward neural network is that each neuron in CNN only connects to a subset of input; this reduces the total number of network parameters and improves training time and process (Mohammadi et al., 2018).

3.1.3. *Autoencoder (AE)*

AE is one of the powerful unsupervised neural network techniques. They consist of three layers: the input layer, the hidden encoding layer (a.k.a. bottleneck), and the decoded output layer. Both input and output layers have the same dimensions. AE reduces the dimensionality and size of the input data after it passes through encoding and transforming into a new representation (a.k.a. latent representation or code). The output layer takes the transformed (or compressed) data as input and converts it back into the same dimensions as the original input data with no or minimum distortion. AE's learn and classify output automatically without the need for a labeled dataset (Ahmad and Alsmadi, 2021). The reason to use Autoencoder is that it can reconstruct errors while detecting anomalies (Moussa and Alazzawi, 2020). It also helps avoid data imbalance and dimensionality reduction, which is usually performed manually in traditional ML algorithms. There are three common types: AE, Stacked AE, Sparse Autoencoder, and variational Autoencoder (VAE) (Ahmad et al., 2020). Researchers have shown great interest in AE when solving anomaly detection, fault diagnostics, dimensionality reduction, compression, and other related problems.

3.1.4. *Recurrent neural network (RNN)*

RNN is a powerful and famous algorithm to find hidden patterns in sequential data (Ahmad and Alsmadi, 2021). One of the advantages of RNNs is that they have a memory cell that can remember data received earlier and capture the temporal dependency among data (Liang and Znati, 2019). It introduces a directional loop to preserve the sequential dependency between the current packet and the previously observed packet. RNN consists of some limitations such as, they suffer from gradient vanishing problem; they also are challenging to train and parallelize (Fu et al., 2019) and cannot remember longer sequences (Chang et al., 2017).

3.1.5. *Long short-term memory (LSTM)*

LSTM is a specialized RNN type that can remember information for an extended period. This property of remembering patterns gives LSTM an edge over traditional RNNs (Ahmad and Alsmadi, 2021). LSTM introduced a memory cell concept to retain important information for a long time. It overcomes the vanishing and exploding gradient problems of RNN through a set of gates to manage information storage and removal (Rezaei and Liu, 2019). The memory cell in LSTM consists of three gates: input gate, forget gate, and output gate (Liang and Znati, 2019). The gates control access and information flow in the memory cell and prevent stored information from being overwritten with irrelevant information. The input gate controls the information flow into and out of the cell. The forgot gate manages the information cell stores; if more relevant information is available, the forgot gate would discard the previously stored information and store new information. The output gate manages when the cell's information can become an output of the cell.

3.1.6. *Bi-directional long short-term memory (BLSTM)*

Despite LSTM's success in sequential data, LSTMs are not suitable for complex tasks which require explicit and external memory (Rezaei and Liu, 2019). LSTM's have a major limitation when training is required in both positive and negative time directions. LSTM cannot operate in both directions (Hayashi et al., 2016). BLSTM is good in processing sequential data in both time directions using a forward LSTM layer and a backword LSTM layer (Cui et al., 2019).

3.1.7. *Temporal convolutional networks (TCN)*

TCN is a variant of CNN that is composed of multiple layers and uses causal convolutions. The combination of CNN architecture with causal padding makes a causal convolution (Derhab et al., 2020). A TCN is known as temporal convolution because it maintains the temporal sequence of data and prevents information loss. TCN is a feed-forward network that predicts the output $\hat{y}$ by only looking at most recent inputs (e.g., k steps) instead of looking at the full history. The k step value can be changed to make the previous input size bigger. TCN, as appose to RNN, allows parallelization because they do not maintain any hidden states and do not depend on previous time step computations. It allows better GPU optimization during training (Gehring et al., 2017).

4. Experiments and results analysis

We divided our analysis into two groups of DL classifiers. Firstly, we implemented individual classifiers on datasets mentioned in Section 2.1 and captured several pieces of valuable information such as training time, parameters, model settings, and performance matrices. Secondly, we captured the same information on several hybrid classifiers. Two separate experiments were performed on NSL-KDD and UNSW_NB15 datasets; In [*a], models were trained and predicted on the given training and testing set. In [*b], we merged the training and testing set, shuffled the dataset, and recreated new training and testing set based on the 70:30 ratio. In [*c], models were trained on the given full dataset and created a training and testing set based on a 70:30 ratio.

4.1. *Individual classifiers*

Table 8, 9, 10 and 11 show the summarized results of individual classifiers, whereas Table 12, 13 and 14show the detailed classification results against each target class.

The summary of individual classifiers in Table 8, 9, 10 and 11 show a big difference in trainable parameters. Fig. 3 compares them against each of the models. CNN takes the least number of trainable parameters, followed by MLP. By looking only at the first dataset, "*Bot_IoT*," CNN uses around 12 K parameters, followed by MLP using around 32 K, followed by TCN using around 79 K, and finally, LSTM using over 3 million parameters.

Table 11
Temporal convolutional networks (TCN).

Dataset	Trainable Parameters	epoch	Batch Size	Training Time	Model Size	Accuracy
Bot-IoT	79,261	20	256	78 min	1 MB	100.00%
CICIDS2017	106,693	20	256	61 min	1 MB	99.90%
NSL-KDD [*a]	135,203	100	256	25 min	2 MB	77.70%
NSL-KDD [*b]	135,203	20	256	4 min	2 MB	98.80%
UNSW_NB15 [*a]	196,782	20	256	4 min	3 MB	73.60%
UNSW_NB15 [*c]	204,086	20	256	86 min	3 MB	97.80%
N-BaIoT	130,055	20	256	27 min	2 MB	90.80%
KDD CUP 99	132,191	20	256	13 min	2 MB	99.90%

Table 12
Performance metrics - individual classifiers.

		MLP			CNN			LSTM			TCN		
Dataset	Attacks	PR	RE	F1	PR	RE	F1	PR	RE	F1	PR	RE	F1
Bot-IoT	DoS	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	DDoS	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	Reconnaissance	0.994	1.000	0.997	1.000	1.000	1.000	0.995	1.000	0.997	0.999	1.000	1.000
	Theft	0.000	0.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	Normal	0.000	0.000	0.000	0.992	0.902	0.945	0.000	0.000	0.000	1.000	0.867	0.929
CICIDS2017	DoS Hulk	0.999	0.999	0.999	0.999	1.000	1.000	0.999	1.000	1.000	0.999	1.000	0.999
	PortScan	1.000	1.000	1.000	1.000	0.999	1.000	1.000	0.999	1.000	0.994	0.999	0.997
	DDoS	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	DoS GoldenEye	0.996	0.990	0.993	0.995	0.990	0.992	0.995	0.976	0.986	0.996	0.992	0.994
	FTP-Patator	0.998	0.992	0.995	0.997	0.998	0.998	0.998	0.999	0.999	0.998	0.994	0.996
	SSH-Patator	0.950	0.996	0.972	0.984	0.991	0.988	0.984	0.995	0.990	0.979	0.993	0.986
	DoS slowloris	0.963	0.994	0.978	0.998	0.993	0.995	0.989	0.997	0.993	0.988	0.994	0.991
	DoS Slowhttptest	0.998	0.972	0.985	0.993	0.995	0.994	0.996	0.987	0.991	0.995	0.989	0.992
	Bot	0.742	0.988	0.847	0.936	0.894	0.915	0.893	0.978	0.933	0.930	0.831	0.878
	Web Attack Brute Force	0.792	0.783	0.788	0.922	0.701	0.796	0.934	0.622	0.746	0.941	0.673	0.785
	Web Attack XSS	0.616	0.597	0.606	0.586	0.852	0.694	0.543	0.893	0.676	0.605	0.944	0.737
	Infiltration	0.000	0.000	0.000	1.000	0.727	0.842	0.727	0.727	0.727	0.071	0.091	0.080
	Web Attack Sql Injection	0.000	0.000	0.000	0.000	0.000	0.000	0.023	0.167	0.041	0.000	0.000	0.000
	Heartbleed	0.000	0.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	BENIGN	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.999	1.000

Table 13
Performance metrics - individual classifiers cont.

		MLP			CNN			LSTM			TCN		
Dataset	Attacks	PR	RE	F1	PR	RE	F1	PR	RE	F1	PR	RE	F1
NSL-KDD [*a]	Dos	0.967	0.790	0.869	0.889	0.771	0.825	0.967	0.754	0.847	0.953	0.815	0.879
	Probe	0.830	0.662	0.737	0.866	0.586	0.699	0.699	0.574	0.630	0.810	0.663	0.730
	R2L	0.731	0.129	0.220	0.924	0.144	0.249	0.962	0.195	0.325	0.858	0.120	0.210
	U2R	0.714	0.149	0.247	0.889	0.239	0.376	0.132	0.403	0.199	0.684	0.194	0.302
	Normal	0.679	0.971	0.799	0.657	0.936	0.772	0.666	0.949	0.783	0.685	0.962	0.800
NSL-KDD [*b]	Dos	0.998	0.995	0.996	0.998	0.995	0.996	0.995	0.996	0.996	0.998	0.995	0.996
	Probe	0.986	0.987	0.986	0.987	0.983	0.985	0.974	0.984	0.979	0.974	0.987	0.980
	R2L	0.840	0.910	0.873	0.952	0.780	0.858	0.872	0.813	0.842	0.926	0.828	0.874
	U2R	0.778	0.194	0.311	0.500	0.139	0.217	0.591	0.361	0.448	0.000	0.000	0.000
	Normal	0.991	0.989	0.990	0.982	0.995	0.988	0.987	0.989	0.988	0.987	0.992	0.989
N-BaIoT	gafgyt_combo	0.979	0.996	0.987	0.994	0.992	0.993	0.998	0.993	0.996	0.988	0.988	0.988
	gafgyt_junk	0.994	0.953	0.973	0.985	0.987	0.986	0.986	0.996	0.991	0.976	0.974	0.975
	gafgyt_scan	0.993	0.999	0.996	0.992	1.000	0.996	0.999	1.000	0.999	0.998	0.999	0.998
	gafgyt_tcp	0.000	0.000	0.000	0.741	0.001	0.001	1.000	0.001	0.002	0.250	0.000	0.000
	gafgyt_udp	0.534	0.999	0.696	0.535	1.000	0.697	0.535	1.000	0.697	0.534	0.999	0.696
	mirai_ack	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	mirai_scan	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	mirai_syn	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	mirai_udp	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	mirai_udpplain	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	Normal	1.000	0.995	0.997	1.000	0.998	0.999	1.000	0.999	0.999	0.999	1.000	0.999

Models with a large number of trainable parameters take a longer time to train the model. As mentioned above, the large number of LSTM trainable parameters indicates that the model would take a longer training time. Table 8, 9, 10 and 11 show training time details of classifiers when trained on each of the datasets. Fig. 4 reflects that there is a significant training time difference in MLP, CNN, TCN, and LSTM. By looking at the UNSW_NB15 dataset only, MLP executes in 5 min, CNN executes in 15 min, TCN in 86 min; however, LSTM takes 532 min of model training time. Finally, the large trainable parameters in LSTM cause the model to have a bigger size than MLP, CNN, and TCN. On average, MLP has a size of 518 KB, CNN has 327 KB, TCN has 2 MB, and LSTM models have 38 MB size.

We performed multiclass classification to gather important performance metrics (PR: precision, RE: recall, F1: F1-score) on each of the datasets to predict various attack classes. Table 12, 13 and 14

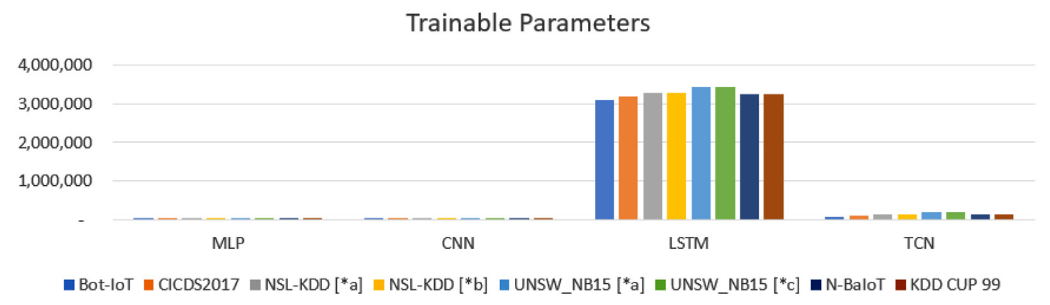

Fig. 3. Trainable parameters - individual classifier.

Table 14
Performance metrics - individual classifiers cont.

		MLP			CNN			LSTM			TCN		
Dataset	**Attacks**	**PR**	**RE**	**F1**	**PR**	**RE**	**F1**	**PR**	**RE**	**F1**	**PR**	**RE**	**F1**
UNSW_NB15 [*a]	Generic	0.111	0.001	0.001	0.000	0.000	0.000	0.971	0.978	0.974	0.997	0.978	0.988
	Exploits	0.586	0.292	0.389	0.588	0.226	0.327	0.492	0.311	0.381	0.600	0.841	0.701
	Fuzzers	0.303	0.192	0.235	0.558	0.224	0.319	0.363	0.235	0.286	0.401	0.050	0.089
	DoS	0.400	0.001	0.003	0.290	0.026	0.048	0.284	0.456	0.350	0.223	0.008	0.015
	Reconnaissance	0.678	0.167	0.269	0.667	0.154	0.250	0.654	0.184	0.287	0.734	0.518	0.607
	Analysis	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	Backdoor	0.000	0.000	0.000	0.000	0.000	0.000	0.500	0.001	0.001	0.000	0.000	0.000
	Shellcode	0.495	0.043	0.080	0.540	0.059	0.107	0.549	0.049	0.091	0.000	0.000	0.000
	Worms	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	Normal	0.368	0.948	0.530	0.342	0.924	0.499	0.495	0.701	0.580	0.699	0.989	0.819
UNSW_NB15 [*c]	Generic	0.995	0.981	0.988	0.999	0.977	0.988	0.997	0.984	0.990	0.995	0.982	0.988
	Exploits	0.571	0.907	0.700	0.573	0.918	0.706	0.599	0.907	0.722	0.580	0.862	0.694
	Fuzzers	0.581	0.499	0.537	0.586	0.370	0.454	0.608	0.561	0.584	0.607	0.352	0.446
	DoS	0.325	0.008	0.015	0.571	0.001	0.002	0.568	0.038	0.071	0.550	0.007	0.013
	Reconnaissance	0.794	0.753	0.773	0.829	0.769	0.798	0.884	0.765	0.820	0.738	0.785	0.761
	Analysis	0.000	0.000	0.000	0.486	0.021	0.041	0.440	0.014	0.027	0.588	0.012	0.024
	Backdoor	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	Shellcode	0.508	0.539	0.523	0.561	0.305	0.395	0.630	0.561	0.593	0.377	0.459	0.414
	Worms	0.000	0.000	0.000	0.000	0.000	0.000	0.750	0.058	0.107	0.000	0.000	0.000
	Normal	0.995	0.996	0.995	0.993	0.998	0.995	0.995	0.997	0.996	0.993	0.998	0.995
KDD CUP 99	Dos	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	Probe	0.990	0.989	0.990	0.990	0.997	0.993	0.993	0.987	0.990	0.992	0.985	0.988
	R2L	0.892	0.950	0.920	0.922	0.950	0.936	0.853	0.947	0.898	0.926	0.920	0.923
	U2R	0.000	0.000	0.000	0.833	0.312	0.455	1.000	0.188	0.316	0.000	0.000	0.000
	Normal	0.999	0.999	0.999	0.999	0.999	0.999	0.999	0.998	0.998	0.998	0.999	0.999

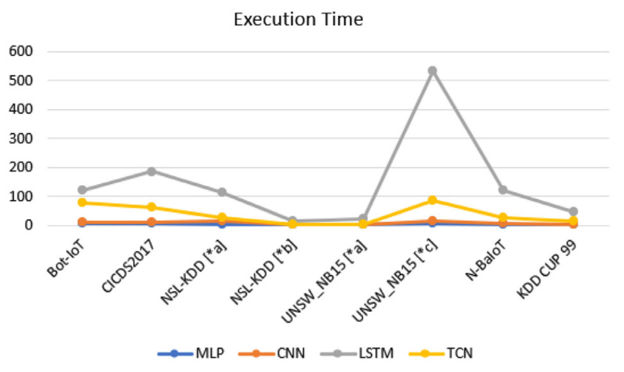

Fig. 4. Execution time - individual classifiers.

show the results obtained against each of the individual classifiers. It is important to look for these performance metrics because some classifiers may show good results on one metric but poor results on a different metric (Liu et al., 2014). The results would also help future researchers to identify biased studies where researchers pick a single model and a single dataset to propose an IDS solution. For clarity and easy identification, results with less than 50% score are shown in red, less than 90% in orange, and above 90% in green background.

Table 12, 13 and 14 reveal important facts in each of the datasets, such as:

- *Bot_IoT: CNN and LSTM can predict "Theft" class instances with 100% precision, but MLP is unable to* predict it. Similarly, CNN can predict "*Normal*" instances with 92% precision, but both MLP and LSTM are unable to predict it. One of the reasons could be because this dataset is highly imbalanced, with only 79 "*Theft*" instances and 477 "*Normal*" instances out of over 3 million records.
- *CICIDS2017:* The same is the case with this dataset prediction. CNN performs well on least instance classes (Infiltration: 36 instances, Heartbleed: 11 instances), but MLP is unable to detect them at all. LSTM, on the other hand, is able to produce only 72% precision on "*infiltration*." None of the classifiers were able to detect 21 instances of *"Web Attack SQL Injection."* None of the classifiers is able to produce over 90% precision on detecting "*bot, Web Attack Brute Force, and Web Attack XSS*" attacks.
- *NSL-KDD [*a]:* For this dataset, all three classifiers were able to detect "*DoS*" attacks with over 89% precision but were unable to produce good detection on the rest of the attack classes. For R2L, LSTM is able to produce 96% precision but a recall of only 19%. From an IDS perspective, a low recall score means a model cannot detect legitimate attacks. This would have serious consequences because IDS would not detect attacks, and malicious traffic would pass through without detection.

Table 15
Autoencoder + TCN.

Dataset	Trainable Parameters	epoch	Batch Size	Training Time	Model Size	Accuracy
Bot-IoT	87,483	10	512	26 min	1 MB	100.00%
CICIDS2017	140,315	10	512	24 min	2 MB	99.99%
NSL-KDD [*a]	173,505	100	256	25 min	2 MB	75.60%
NSL-KDD [*b]	173,505	100	256	20 min	2 MB	98.90%
UNSW_NB15 [*a]	339,626	100	32	53 min	2 MB	75.70%
UNSW_NB15 [*c]	348,858	50	256	208 min	3 MB	97.90%
N-BaIoT	167,073	20	128	39 min	2 MB	90.70%
KDD CUP 99	169,473	10	32	22 min	2 MB	99.90%

Table 16
Autoencoder + LSTM.

Dataset	Trainable Parameters	epoch	Batch Size	Training Time	Model Size	Accuracy
Bot-IoT	3,104,180	5	512	35 min	36 MB	100.00%
CICIDS2017	3,204,681	5	512	40 min	37 MB	99.40%
NSL-KDD [*a]	3,291,591	100	256	90 min	38 MB	79.00%
NSL-KDD [*b]	3,291,591	20	32	68 min	38 MB	98.90%
UNSW_NB15 [*a]	3,454,951	50	32	193 min	40 MB	42.30%
UNSW_NB15 [*c]	3,511,558	15	512	216 min	40 MB	97.90%
N-BaIoT	3,276,174	10	256	42 min	38 MB	80.50%
KDD CUP 99	3,282,603	10	128	30 min	38 MB	99.90%

- *NSL-KDD [*b]:* None of the classifiers were able to produce good performance results on detecting the U2R class. MLP returned a recall value of 19.4%, whereas CNN returned 13.9%, and LSTM returned 36.1%.
- *UNSW_NB15 [*a]:* LSTM is able to detect "*Generic*" class with 97.1% precision. None of the classifiers were able to produce over 67% performance results on detecting any other classes. None of the classifiers were able to detect "*Analysis, Backdoor, & Worms*" instances.
- *N_BaIoT:* All of the classifiers performed well on this dataset with 100% precision in detecting most of the classes. MLP was unable to detect "*gafgyt_tcp*," CNN was able to produce 74.1% precision but only 1% recall. Similarly, LSTM was able to produce 100% precision but only 1% of recall, which is not a good detection capability.
- *UNSW_NB15 [*c]:* All classifiers were able to detect "*Generic*" and "*Normal*" classes with over 99% precision and recall. However, MLP was unable to detect any other class with more than 79.4%, CNN with more than 82.9%, and LSTM with more than 88.4%. None of the classifiers were able to detect "*Backdoor*" instances.
- *KDD CUP 99:* All classifiers relatively performed well on this dataset too. MLP is unable to detect U2R, whereas CNN was able to detect it with 83.3% precision but only a 31.2% recall value. Similarly, LSTM was able to return 100% precision on detecting U2R but with only an 18.8% of recall value.

4.2. Hybrid classifiers

Table 15, 16, 17, 18 and 19 show the summarized results of hybrid classifiers, whereas Table 20, 21 and 22 show the detailed classification results against each target class.

As stated above, trainable parameters in a neural network have an important role in model training. They depend on the number of nodes of the model input, hidden, and output layers. Fig. 5 compares the difference in the number of trainable parameters in each of the models uses. The "*Autoencoder + TCN*" takes on average the least number of trainable parameters, around 199 K, whereas "*Autoencoder + BLSTM*" takes the most over 9.2 million parameters. Compare to "*Autoencoder + TCN*," all other classifiers also have a big difference in parameters, with "*Autoencoder + LSTM*" taking over 3.3 million and "*CNN + LSTM*" taking over 3.2 million trainable parameters.

The larger the trainable parameters, the longer it takes for the model to get trained. Fig. 6 shows the execution time difference in each of the models against the datasets. The "*Autoencoder + TCN*" trains the model very quickly with an average of 52 min, "*Autoencoder + LSTM*" with 90 min, "*Autoencoder + BRNN*" with 229 min, "*Autoencoder + LSTM*" with 164 min, and "*CNN + LSTM*" with an average time of 142 min.

Fig. 7 shows the impact on model size based on different classifiers, trainable parameters, and settings. The "*Autoencoder + TCN*" produces a very small model size of an average 2 MB, "*Autoencoder + LSTM*" with 38 MB, "*Autoencoder + BRNN*" with 31 MB, "*Autoencoder + BLSTM*" with 105 MB, and lastly "*CNN + LSTM*" with an average of 38 MB.

Table 20, 21 and 22 show the multiclass classification results obtained against each of the hybrid classifiers. These results reveal important findings on the good and bad attack detection capabilities of the classifiers. These results would also help future researchers to identify biased studies where researchers pick a single model and a single dataset to propose an IDS solution. For clarity and easy identification, results with less than 50% score are shown in red, less than 90% in orange, and above 90% in green background.

Table 20, 21 and 22 reveal important facts in each of the datasets, such as:

- *Bot_IoT:* "*Autoencoder+TCN*" was the only model that successfully detected each class with over 99% accuracy. None of the other models were able to detect the "*theft*" class. Only "*Autoencoder+BRNN*" was able to detect the "*Normal*" class with 72.7% precision. The rest of the models were not able to detect "*Normal*" class instances.
- *CICIDS2017:* "*Autoencoder+TCN*" and "*Autoencoder+LSTM*" were able to detect most of the classes with over 96% accuracy. None of the models could detect "*Web Attack XSS, Infiltration, Web Attack SQL Injection, & Heartbleed*" except "*CNN+LSTM,*" which also reveals poor precision of less than 50%. Regarding the "*Bot*" attack, both "*Autoencoder+LSTM & Autoencoder+BRNN*" were unable to detect it, whereas other models were able to detect it with over 90% precision. Overall, "*Autoencoder+LSTM & Autoencoder+BRNN*" produced the least detection capabilities for this dataset, and "*Autoencoder+TCN*" produced good results.

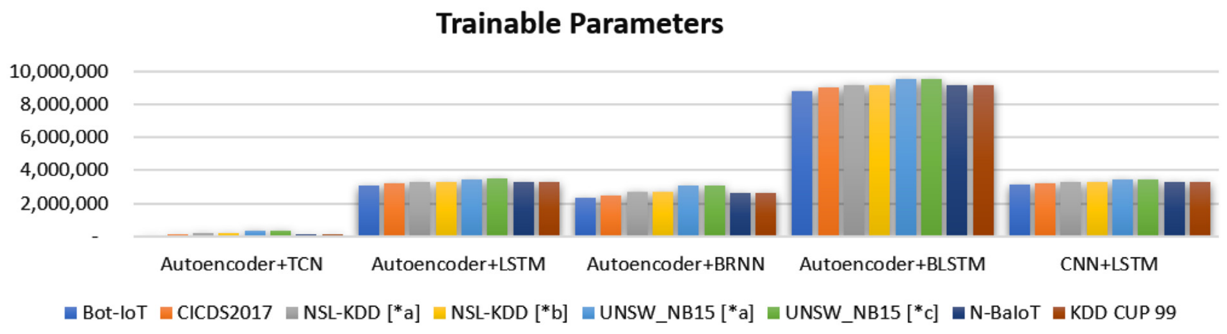

Fig. 5. Trainable parameters - hybrid classifiers.

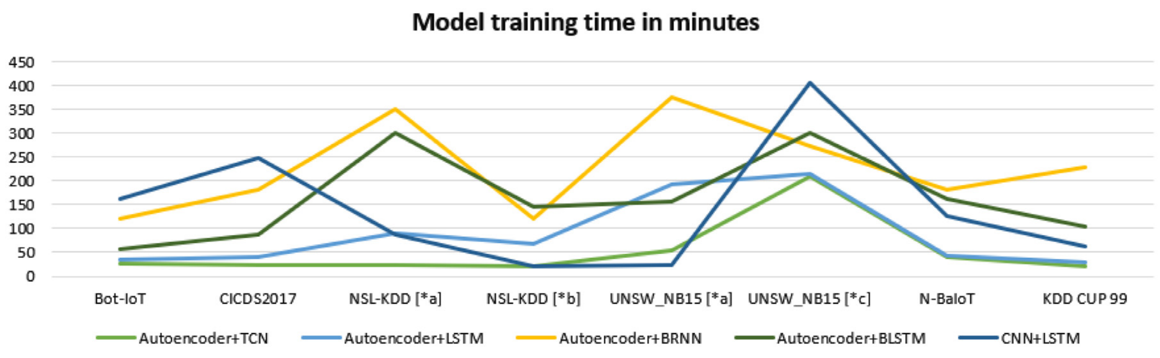

Fig. 6. Execution time - individual classifiers.

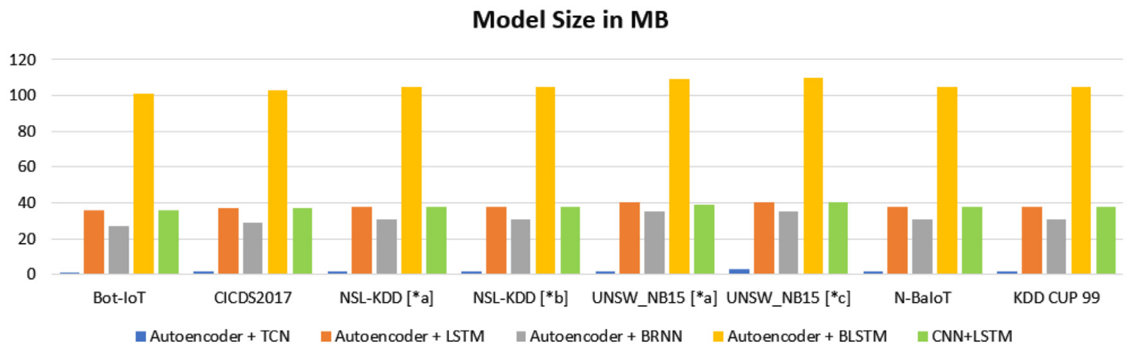

Fig. 7. Model size - hybrid classifiers.

- *NSL-KDD [*a]:* For this dataset, none of the models produced good results. The models were only able to successfully detect "*DoS*" attacks with over 927% precision. The precision score for all other classes is below 70%. "*Autoencoder+LSTM*" was able to detect "*R2L*" with 98.3% precision but with a very poor recall score of only 13%. From an IDS perspective, a low recall score means the model cannot detect legitimate attacks. This would have serious consequences because IDS would not detect attacks, and malicious traffic would pass through without detection.
- *NSL-KDD [*b]:* "*Autoencoder+TCN*" produced the best results in this dataset with over 94.9% precision score. It was also able to detect "*U2R*" with 76% precision which is not optimal but still much better than other models, which could not produce more than a 64% score. "*Autoencoder+BRNN*" is the least performing model that only detected "Normal" instances with 51.9% precision but could not detect any other attack classes.
- *N_BaIoT:* "*Autoencoder+BRNN*" is the least performing model that only detected "mirai_udp" instances with 23.3% precision but could not detect any other attack classes. Only "*CNN+LSTM*" was able to detect "*gafgyt_tcp*" with 74.1% precision but a very poor recall score of only 1%. None of the other models were able to detect the "*gafgyt_tcp*" attack class. Overall, three models, "*Autoencode+TCN, Autoencoder+BLSTM, CNN+LSTM,*" performed well on this dataset with an over 99% precision score.
- *UNSW_NB15 [*a]:* Overall, none of the models, performed well on this dataset. Again, "*Autoencoder+BRNN*" performed poorly and was only able to detect "*Normal*" instances with 31.9% precision. No other models were able to detect other classes with good performance measures.
- *UNSW_NB15 [*c]:* Overall, none of the models performed well on this dataset. All models were able to detect only "*Generic and Normal*" classes with over 99.3% precision. The rest of the attack classes were not detected with good precision by any of the models. Again, "*Autoencoder+BRNN*" performed poorly and could not detect classes other than "*Generic and Normal*."
- *KDD CUP 99:* Two models, "*Autoencoder+TCN*" and "*CNN+LSTM,*" produced good precision results on detecting each of the classes. Again, "*Autoencoder+BRNN*" performed poorly and was unable to detect classes other than "*DoS,*" which is detected with a 79.2% precision rate. Models "*Autoencoder+LSTM, Autoencoder+BRNN, Autoencoder+BLSM*" were unable to detect

the "*U2R*" class but "*Autoencoder+TCN*" and "*CNN+LSTM*" were able to detect it with over 80% precision score.

5. Discussion

The empirical results and analysis presented in Section 4 reveal exciting findings that need further discussions. This section elaborates those points from both empirically and the scholarly literature.

5.1. Factors impacting model performance

A good, generalized performance of deep learning algorithms depends on many internal and external factors. Some of them are as the followings:

5.1.1. Trainable parameters in deep learning models

Trainable parameters in a neural network have an important role in model training. They depend on the number of nodes in the model input, hidden, and output layers. In DNN or CNN, the information constantly flows forward from input to multiple hidden layers. They have no connections to the node itself or any other perceptron on the same or previous layers; this results in a small number of trainable parameters. The CNN uses convolutional and pooling operations to reduce dimensionality and identify important features. As a result, it utilizes less trainable parameters. LSTM, on the other hand, consists of three gates (i.e. input gate, forget gate, output gate) and maintains a memory to remember information for an extended period, thus requiring more trainable parameters.

Trainable model parameters have several advantages. Even though a higher number of parameters require a longer training time, it provides a degree of freedom to the model. On the other hand, reducing the number of trainable parameters has several advantages, such as increased training time as a result of several factors: (1) reduced objects and smaller gradient, (2) smaller parameters are easy to adjust, requires less memory, and (3) fewer chances are for the classifier to overfit (Wu, 2020). Additionally, less trainable parameters reduce the need to add a dropout layer in some classifiers such as CNN, LSTM, and TCN.

5.1.2. Data size and impact on deep learning models

A lot of proposed IDS solutions based on traditional ML algorithms do not address computational challenges. The performance of conventional ML algorithms plateau once a specific threshold limit is reached due to dataset size (Ng et al., 2019). To overcome this challenge, DL provides optimal results as the dataset size grows. However, a limitation in DL is that it requires a longer training time as the training data size grows, i.e., the larger the training data size, the greater the required training time (Roopak et al., 2019). We empirically proved this point using our proposed algorithms in this paper and by performing additional experiments. The followings are the steps we applied toward this goal:

- We fixed the classifier and its attributes, fixed the trainable parameters, epochs, and the batch size, and then performed 27 experiments (with variations in the size of the dataset) on three datasets (Bot_IoT, N_BaIoT, and CICIDS2017). In particular, in those experiments, we evaluated three DL classifiers/models: MLP, CNN, and Autoencoder + TCN.
- After fixing the model settings, we trained the model on three dataset possible sizes: full, 70%, and 30% datasets, respectively, to evaluate the impact on model training time with varying training data size. Table 23 shows the results of all 27 experiments.
- The training time dropped considerably when reducing the training dataset from full to 70% or 30%.
- The results reveal that “Autoencoder + TCN” on N_BaIoT dataset took 39 min to train the model on the full dataset. However, with a reduced 70% dataset, the same model took 19 min to train (a reduction of 51.28%); and with 30% dataset, the same model took 8 min to train (a reduction of 79.49%).

5.1.3. DL models and features set selection

A critical step in machine learning is to select the right and least number of features contributing best to the classification model. Deep learning is known to perform well on large datasets with a large number of features without the need to select features manually (Roopak et al., 2019). However, after years of research, researchers still encounter problems optimally handling datasets with large feature sets and changing network traffic patterns (Veena, 2018). Training a DL classifier with a large dataset containing a large number of features would generate thousands of neurons, leading to the problem of “Curse of Dimensionality”; in such a situation, reducing the feature set helps solve this problem (Wojtowytsch and E, 2020). Therefore, several researchers adopted approaches to reduce total features and claim to obtain optimum performance in their models. For example, in one research study, the authors reduced the feature set using NSGA-II multi-objective optimization method and a hybrid classifier (CNN+LSTM) to propose their IDS model (Roopak et al., 2020b). The authors reported detecting DDoS attacks with 99.03% accuracy. The authors reported a 5-fold training time reduction in reduced feature sets compared to the full dataset model training approach to prove their intuition.

5.1.4. DL models and hyperparameters

Deep learning provides a wealth of activation functions, optimizers, and loss functions to optimize model performance. The settings of our proposed models in this paper consist of “ReLU” activation functions in the hidden layers, “Softmax” in the output layer, adam optimizer, and categorical cross-entropy as the loss function. A poor choice of loss function would end up getting the local optima instead of a global optimum. One of the goals of DL algorithms is to find the weights and biases that produce the lowest value for the loss function. For models trained with stochastic gradient descent, looking at the characteristics of the loss function helps to decide an optimal loss function to minimize generalization errors. A loss function that is more stable on input variations provides better generalization (Akbari et al., 2021). The architecture of feed-forward neural networks (MLP) is known to produce poor performance due to gradient decay (Ibitoye et al., 2019). Similarly, RNNs (BRNNs) are difficult to train and parallelize due to their inherent nature of vanishing gradient problems (Fu et al., 2019). In our hybrid classifier experiments in Table 17 the Autoencoder + BRNN took very long (350 min) to train the model on the NSL_KDD dataset. Even though LSTM is known to provide the best results for sequential data (Liang and Znati, 2019), they are not suitable for complex tasks that require explicit and external memory (Rezaei and Liu, 2019), causing them to impact model performance.

5.1.5. DL models and resource constraints

Data scientists often utilize various hardware-based optimization methods to improve model performance. System resources such as processing, memory, and storage have an important role in model training and inference phases. A common approach is the hardware architecture selection tradeoff between CPUs, GPUs, and TPUs. Tensor Processing Units (TPU) are relative recently launched in 2015 to increase deep learning performance. On average, a TPU processes deep learning algorithms 15x-30x faster and 80x more power-efficient than CPUs and GPUs (Jouppi et al., 2017).

Deploying deep learning models on resource constraint IoT devices has been challenging because DL models demand high

Table 17
Autoencoder + BRNN.

Dataset	Trainable Parameters	epoch	Batch Size	Training Time	Model Size	Accuracy
Bot-IoT	2,329,268	5	512	121 min	27 MB	100.00%
CICIDS2017	2,517,833	5	512	181 min	29 MB	99.40%
NSL-KDD [*a]	2,686,663	50	256	350 min	31 MB	73.60%
NSL-KDD [*b]	2,686,663	20	256	121 min	31 MB	51.90%
UNSW_NB15 [*a]	3,044,095	50	256	375 min	35 MB	31.90%
UNSW_NB15 [*c]	3,074,566	5	512	272 min	35 MB	95.70%
N-BaIoT	2,656,910	5	256	181 min	31 MB	23.30%
KDD CUP 99	2,669,483	10	128	228 min	31 MB	79.20%

Table 18
Autoencoder + BLSTM.

Dataset	Trainable Parameters	epoch	Batch Size	Training Time	Model Size	Accuracy
Bot-IoT	8,825,780	5	512	56 min	101 MB	100.00%
CICIDS2017	9,014,345	5	512	88 min	103 MB	99.90%
NSL-KDD [*a]	9,183,175	100	256	300 min	105 MB	76.00%
NSL-KDD [*b]	9,183,175	20	32	145 min	105 MB	98.70%
UNSW_NB15 [*a]	9,540,607	50	256	156 min	109 MB	38.10%
UNSW_NB15 [*c]	9,571,078	5	512	300 min	110 MB	97.70%
N-BaIoT	9,153,422	10	256	162 min	105 MB	90.80%
KDD CUP 99	9,165,995	10	128	105 min	105 MB	99.50%

Table 19
CNN + LSTM.

Dataset	Trainable Parameters	epoch	Batch Size	Training Time	Model Size	Accuracy
Bot-IoT	3115,925	20	256	162 min	36 MB	97.90%
CICIDS2017	3204,159	20	256	248 min	37 MB	99.90%
NSL-KDD [*a]	3285,909	100	256	88 min	38 MB	76.30%
NSL-KDD [*b]	3285,909	20	256	20 min	38 MB	98.70%
UNSW_NB15 [*a]	3439,594	20	256	25 min	39 MB	64.40%
UNSW_NB15 [*c]	3453,930	20	256	407 min	40 MB	98.00%
N-BaIoT	3271,675	20	256	125 min	38 MB	90.90%
KDD CUP 99	3277,717	20	256	62 min	38 MB	99.90%

Table 20
Performance metrics - hybrid classifiers.

		Autoencoder + TCN			**Autoencoder + LSTM**			**Autoencoder + BRNN**			**Autoencoder + BLSTM**			**CNN + LSTM**		
Dataset	**Attacks**	**PR**	**RE**	**F1**	**PR**	**RE**	**F1**	**PR**	**RE**	**F1**	**PR**	**RE**	**F1**	**PR**	**RE**	**F1**
Bot-IoT	DoS	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.999	0.955	0.976
	DDoS	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.962	0.999	0.981
	Reconnaissance	1.000	1.000	1.000	0.995	1.000	0.997	0.998	1.000	0.999	0.995	1.000	0.997	0.995	1.000	0.997
	Theft	1.000	0.958	0.979	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	Normal	0.992	0.916	0.953	0.000	0.000	0.000	0.727	0.671	0.698	0.000	0.000	0.000	0.000	0.000	0.000
CICIDS2017	DoS Hulk	0.999	1.000	0.999	0.991	0.997	0.994	0.973	0.999	0.986	0.999	1.000	0.999	0.998	1.000	0.999
	PortScan	0.993	0.999	0.996	0.991	0.998	0.995	0.993	0.996	0.995	0.993	0.999	0.996	0.994	0.999	0.997
	DDoS	1.000	0.998	0.999	0.996	0.988	0.992	0.998	0.990	0.994	1.000	0.999	0.999	1.000	0.999	0.999
	DoS GoldenEye	0.985	0.991	0.988	0.965	0.736	0.835	0.804	0.867	0.834	0.983	0.987	0.985	0.998	0.986	0.992
	FTP-Patator	0.998	0.992	0.995	0.938	0.996	0.966	0.877	0.959	0.916	0.998	0.991	0.995	0.998	0.998	0.998
	SSH-Patator	0.976	0.989	0.982	0.666	0.989	0.796	0.910	0.475	0.625	0.971	0.988	0.979	0.984	0.996	0.990
	DoS slowloris	0.971	0.971	0.971	0.770	0.347	0.479	0.816	0.656	0.727	0.982	0.963	0.972	0.999	0.958	0.978
	DoS Slowhttptest	0.965	0.982	0.974	0.507	0.572	0.537	0.766	0.631	0.692	0.968	0.971	0.969	0.959	0.997	0.977
	Bot	0.979	0.726	0.834	0.000	0.000	0.000	0.000	0.000	0.000	0.995	0.680	0.808	0.900	0.891	0.896
	Web Attack Brute Force	0.669	0.858	0.752	0.000	0.000	0.000	0.000	0.000	0.000	0.665	0.843	0.743	0.692	0.960	0.804
	Web Attack XSS	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.500	0.015	0.030
	Infiltration	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.308	0.727	0.432
	Web Attack Sql Injection	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.333	0.500	0.400
	Heartbleed	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	BENIGN	1.000	1.000	1.000	0.997	0.999	0.998	0.998	0.999	0.999	1.000	0.999	1.000	1.000	0.999	1.000

Table 21
Performance metrics - hybrid classifiers cont.

		Autoencoder + TCN			**Autoencoder + LSTM**			**Autoencoder + BRNN**			**Autoencoder + BLSTM**			**CNN + LSTM**		
Dataset	**Attacks**	**PR**	**RE**	**F1**	**PR**	**RE**	**F1**	**PR**	**RE**	**F1**	**PR**	**RE**	**F1**	**PR**	**RE**	**F1**
NSL-KDD [*a]	Dos	0.965	0.803	0.876	0.953	0.800	0.870	0.940	0.700	0.803	0.927	0.818	0.869	0.959	0.758	0.847
	Probe	0.682	0.626	0.653	0.854	0.789	0.820	0.614	0.774	0.685	0.669	0.644	0.656	0.652	0.730	0.689
	R2L	0.478	0.143	0.220	0.983	0.130	0.230	0.000	0.000	0.000	0.545	0.130	0.210	0.895	0.192	0.317
	U2R	0.480	0.358	0.410	0.850	0.254	0.391	0.000	0.000	0.000	0.341	0.209	0.259	0.810	0.254	0.386
	Normal	0.685	0.925	0.787	0.697	0.970	0.811	0.679	0.964	0.796	0.700	0.922	0.796	0.690	0.937	0.795
NSL-KDD [*b]	Dos	0.997	0.994	0.996	0.997	0.995	0.996	0.000	0.000	0.000	0.998	0.993	0.995	0.996	0.995	0.996
	Probe	0.982	0.992	0.987	0.978	0.987	0.983	0.000	0.000	0.000	0.959	0.991	0.975	0.975	0.980	0.978
	R2L	0.949	0.842	0.893	0.901	0.904	0.902	0.000	0.000	0.000	0.882	0.902	0.892	0.874	0.918	0.896
	U2R	0.760	0.528	0.623	0.383	0.500	0.434	0.000	0.000	0.000	0.645	0.556	0.597	0.442	0.528	0.481
	Normal	0.987	0.994	0.990	0.991	0.990	0.990	0.519	1.000	0.683	0.991	0.988	0.990	0.990	0.987	0.989
N-BaIoT	gafgyt_combo	0.965	0.992	0.978	0.781	0.877	0.826	0.000	0.000	0.000	0.994	0.985	0.989	0.994	0.992	0.993
	gafgyt_junk	0.985	0.924	0.953	0.703	0.481	0.571	0.000	0.000	0.000	0.974	0.985	0.980	0.985	0.987	0.986
	gafgyt_scan	0.994	0.999	0.997	0.885	0.940	0.912	0.000	0.000	0.000	0.986	0.998	0.992	0.992	1.000	0.996
	gafgyt_tcp	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.741	0.001	0.001
	gafgyt_udp	0.534	0.999	0.696	0.534	0.999	0.696	0.000	0.000	0.000	0.534	0.999	0.696	0.535	1.000	0.697
	mirai_ack	1.000	1.000	1.000	0.767	0.333	0.465	0.000	0.000	0.000	1.000	0.999	0.999	1.000	1.000	1.000
	mirai_scan	1.000	1.000	1.000	0.999	0.999	0.999	0.000	0.000	0.000	1.000	0.999	1.000	1.000	1.000	1.000
	mirai_syn	1.000	1.000	1.000	0.984	0.998	0.991	0.000	0.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000
	mirai_udp	1.000	1.000	1.000	0.770	0.956	0.853	0.233	1.000	0.378	1.000	1.000	1.000	1.000	1.000	1.000
	mirai_udpplain	1.000	1.000	1.000	0.993	0.997	0.995	0.000	0.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000
	Normal	0.997	0.997	0.997	0.968	0.934	0.950	0.000	0.000	0.000	0.997	0.994	0.996	1.000	0.998	0.999

Table 22
Performance metrics - hybrid classifiers cont.

		Autoencoder + TCN			**Autoencoder + LSTM**			**Autoencoder + BRNN**			**Autoencoder + BLSTM**			**CNN + LSTM**		
Dataset	**Attacks**	**PR**	**RE**	**F1**	**PR**	**RE**	**F1**	**PR**	**RE**	**F1**	**PR**	**RE**	**F1**	**PR**	**RE**	**F1**
UNSW_NB15 [*a]	Generic	0.987	0.980	0.984	0.915	0.132	0.230	0.000	0.000	0.000	0.020	0.000	0.000	0.975	0.967	0.971
	Exploits	0.617	0.866	0.721	0.354	0.291	0.319	0.000	0.000	0.000	0.627	0.222	0.328	0.630	0.321	0.425
	Fuzzers	0.518	0.170	0.256	0.210	0.212	0.211	0.000	0.000	0.000	0.209	0.217	0.213	0.502	0.229	0.314
	DoS	0.383	0.099	0.157	0.294	0.060	0.099	0.000	0.000	0.000	0.220	0.009	0.017	0.316	0.544	0.399
	Reconnaissance	0.861	0.533	0.658	0.739	0.168	0.273	0.000	0.000	0.000	0.691	0.175	0.279	0.730	0.259	0.383
	Analysis	0.000	0.000	0.000	0.098	0.002	0.004	0.000	0.000	0.000	0.031	0.002	0.004	0.000	0.000	0.000
	Backdoor	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.250	0.001	0.001
	Shellcode	0.219	0.409	0.286	0.339	0.034	0.062	0.000	0.000	0.000	0.300	0.124	0.175	0.592	0.051	0.094
	Worms	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.250	0.008	0.015	0.000	0.000	0.000
	Normal	0.763	0.968	0.853	0.445	0.944	0.605	0.319	1.000	0.484	0.380	0.952	0.543	0.585	0.892	0.707
UNSW_NB15 [*c]	Generic	0.996	0.981	0.988	0.994	0.984	0.989	1.000	1.000	1.000	0.995	0.979	0.987	0.996	0.982	0.989
	Exploits	0.579	0.918	0.710	0.588	0.904	0.713	0.292	1.000	0.451	0.559	0.930	0.698	0.577	0.912	0.707
	Fuzzers	0.578	0.534	0.555	0.550	0.612	0.580	0.000	0.000	0.000	0.601	0.363	0.452	0.647	0.501	0.565
	DoS	0.422	0.028	0.053	0.518	0.026	0.050	0.000	0.000	0.000	0.351	0.008	0.016	0.608	0.028	0.053
	Reconnaissance	0.878	0.728	0.796	0.852	0.725	0.784	0.000	0.000	0.000	0.862	0.630	0.728	0.866	0.781	0.821
	Analysis	0.333	0.007	0.015	0.583	0.009	0.017	0.000	0.000	0.000	0.500	0.010	0.020	0.309	0.052	0.089
	Backdoor	0.000	0.000	0.000	0.587	0.039	0.072	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.007	0.014
	Shellcode	0.458	0.530	0.491	0.613	0.517	0.560	0.000	0.000	0.000	0.423	0.336	0.374	0.558	0.662	0.605
	Worms	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.200	0.019	0.035
	Normal	0.995	0.996	0.995	0.996	0.995	0.996	1.000	0.981	0.991	0.993	0.997	0.995	0.995	0.997	0.996
KDD CUP 99	Dos	1.000	1.000	1.000	1.000	1.000	1.000	0.792	1.000	0.884	0.997	0.999	0.998	1.000	1.000	1.000
	Probe	0.991	0.982	0.987	0.991	0.970	0.980	0.000	0.000	0.000	0.796	0.913	0.851	0.991	0.991	0.991
	R2L	0.855	0.962	0.905	0.907	0.896	0.902	0.000	0.000	0.000	0.000	0.000	0.000	0.923	0.956	0.939
	U2R	0.800	0.500	0.615	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.846	0.688	0.759
	Normal	0.999	0.998	0.998	0.998	0.999	0.998	0.000	0.000	0.000	0.998	0.993	0.995	0.999	0.999	0.999

computational resources such as memory, processing, and storage (Lane et al., 2016). The researchers have proposed several techniques to incorporate deep learning in resource constraint devices such as optimizing parameters, weights, and hidden layers to create low resource pre-trained models; extract the computational operations from the device to network layers that are scalable, e.g., at the router level (Chaudhary and Gupta, 2019; Desai et al., 2019), gateway level (Kumar and Lim, 2019), fog level (Samy et al., 2020), or cloud level (Alabdulatif et al., 2019).

5.2. Individual vs. hybrid classifiers – comparison

This study provides an analysis of several well-known individual and hybrid deep learning classifiers and their performance on several benchmark IDS datasets. We provided several pieces of valuable model summary and detailed attack classification results. One of the goals was to provide benchmarks to future researchers when selecting a particular dataset to know which deep learning algorithm performs well and how such performance may vary while changing the dataset.

The classifier results are not directly comparable because the hybrid classifiers require more trainable parameters and longer training time; as a result, in our experiments, we had to adjust the number of epochs and batch sizes to complete model training in a reasonable time. For example, as shown in Table 11, TCN was trained with 20 epochs and batch size of 256, whereas Table 15 shows “Autoencoder + TCN” trained with ten epochs and a batch size of 512. Because of the bigger epoch and the smaller batch size, the individual classifier takes a longer training time (78 min) than the hybrid classifier’s training time (26 min).

Table 23
Impact of dataset size on DL model training.

						Full Dataset	70% Dataset		30% Dataset	
Dataset	Classifier	Trainable Parameters	epoch	Batch Size	Model Size	Training Time	Training Time	%age decrease	Training Time	%age decrease
Bot-IoT	MLP	32,581	20	256	434 KB	7 min	5 min	28.57%	2 min	71.43%
	CNN	12,901	20	256	198 KB	11 min	7 min	36.36%	3 min	72.73%
	Autoencoder + TCN	87,483	10	512	1 MB	26 min	18 min	30.77%	8 min	69.23%
N_BaIoT	MLP	42,411	20	256	550 KB	2 min	1.5 min	25.00%	0.5 min	75.00%
	CNN	22,683	20	256	313 KB	5 min	2 min	60.00%	1 min	80.00%
	Autoencoder + TCN	167,073	20	128	2 MB	39 min	19 min	51.28%	8 min	79.49%
CICIDS 2017	MLP	38,255	20	256	500 KB	5 min	3.5 min	30.00%	1.6 min	68.00%
	CNN	18,495	20	256	264 KB	9 min	6 min	33.33%	2.5 min	72.22%
	Autoencoder + TCN	140,315	10	512	2 MB	24 min	17 min	29.17%	7.5 min	68.75%

While comparing the two models in the given settings and on the CICIDS2017 dataset, TCN outperforms “Autoencoder + TCN” in detecting the attack class “*Web Attack XSS*”. The “Autoencoder + TCN” cannot detect this attack, while TCN could detect it with 0.616 precision, 0.597 recall, and 0.606 F1-score, respectively. The same is the case with NSL-KDD[*a] dataset, where TCN detected Probe (0.810), R2L (0.858), and U2R (0.684) better than “Autoencoder+TCN,” Probe (0.626), R2L (0.143), and U2R (0.358). However, for NSL-KDD[*b*], TCN could not detect the “U2R” attack, while “Autoencoder +*TCN*” could detect it with the precision value of 0.76. Lastly, for KDD CUP 99 dataset, “Autoencoder+TCN” outperformed TCN by successfully detecting U2R attack with the precision of 0.800 while TCN could not detect this same attack.

By looking at the available datasets and trained classifiers, we can provide an overall comparison based on the datasets to which the classifier provided optimal results. For example

- for BoT_IoT dataset, CNN from individual and “Autoencoder+TCN” from hybrid classifier provided best classification results
- for CICIDS2017 dataset, LSTM from individual and “CNN+LSTM” from hybrid classifiers performed well
- for NSL-KDD[*a] dataset, CNN from individual and “Autoencoder+LSTM” from hybrid classifiers performed well
- for NSL-KDD[*b] dataset, MLP from individual and “Autoencoder+TCN” from hybrid classifiers performed well
- for N_BaIoT dataset, CNN from individual and “CNN+LSTM” from hybrid classifier provided best classification results
- for UNSW_NB15[*a] dataset, LSTM from individual and “Autoencoder+TCN” from hybrid classifier provided best classification results
- for UNSW_NB15[*c] dataset, LSTM from individual and “CNN+LSTM” from hybrid classifier provided best classification results
- for KDD CUP 99 dataset, CNN from individual and “CNN+LSTM” from hybrid classifier provided best classification results.

The experiments produced mixed results and could not reflect an accurate comparison unless the same classifiers with the same settings are compared. We leave this for future work to provide a more comprehensive comparison.

5.3. Classifier considerations for IoTs

One of our goals for this study was to build an IDS benchmark for resource constraint IoT devices by comparing various DL classifiers and datasets. Among the various DL methods researchers proposed and from this study, adopting them on embedded systems is a major challenge and has significant barriers (Lane et al., 2015). One of the reasons for this is the complex nature of deep learning models and the resources it requires. IoT devices are limited in resources and consists of bottlenecks that do not exist in personal computing devices or environments. Some of the bottlenecks for implementing deep learning solutions in an IoT environment include the followings:

5.3.1. Memory constraints

IoTs are small computational devices with very little memory footprint. For all practical purposes, the users would not want to install models that are large in size and require longer installation and processing time. The experimental results presented in Section 4 reveal that on average, the model LSTM has a size of 38 MB, "*Autoencoder + LSTM*" has 38 MB, "*Autoencoder + BRNN*" has 31 MB, "*Autoencoder + BLSTM*" has 105 MB, and "*CNN + LSTM*" has 38 MB. Devices with small footprints are likely to prevent running these models due to high memory requirements. On the other hand, CNN and “Autoencoder+TCN” require very little space because they require a minimum number of trainable parameters compared to the rest of the models.

5.3.2. Computational challenges

MLP is known to be computationally expensive. Each node in an MLP connects to all the nodes in the subsequent hidden or output layers. The previous layer’s input is multiplied by a weight, added to a bias, and passed through an activation function. A typical MLP consists of tens of layers and hundreds of neurons in each layer, resulting in the number of parameters easily reaching millions (Lane et al., 2016). In a resource constraint environment, critical computational resources can quickly drain out if a large number of MLPs are operational, even with optimized MLP models (Han et al., 2016).

An IDS has to continuously detect intrusions and would require frequent computations. Executing the model that requires more extended training and inference time on resource constraint devices will not be accepted. The experimental results presented in Section 4 reveal that some models are extremely slow in training compared to others. For example, LSTM takes 532 min to train on the UNSW_NB15 dataset in individual classifiers, compared to CNN, which trains in 15 min on the same dataset. Similarly, LSTM takes 121 min to train on the N-BaIoT dataset, while CNN takes only 5 min to train on the same dataset. On the other hand, from the hybrid classifiers, “Autoencoder+BRNN” is the slowest algorithm, taking over 350 min to train on NSL-KDD and UNSW_NB15 datasets; whereas, “Autoencoder+TCN” produces the best results with training on the same datasets in less than 25 min.

5.3.3. Energy constraints

Energy constraints are directly related to computational complexities. Deep learning models that take long execution time require excessive energy during model training, re-training, and testing. Models using excessive energy would be prohibited from running in resource constraint environments, even if they perform well on detecting anomalies. In a recent MIT study, certain DL models that use graphical processing units (GPU) and tensor processing units (TPU) are shown to consume enormous energy and significantly impact the environment. The study revealed that certain DL models emit on average over 626,000 pounds of carbon dioxide, which is approximately five times the lifetime emissions of the average car in the U.S. (Strubell et al., 2019). As revealed in Section 4 that specific models (LSTM, Autoencoder+LSTM, Autoencoder+BRNN, Autoencoder+BLSTM, and CNN+LSTM) in our experimental stack are heavy in size, take very long training time, and are heavy in energy consumption. For future researchers, this study provides a benchmark and guidance to narrow down classifier selection for building an IDS for resource constraint devices.

5.4. Classifier performance rational

The strategy adopted in this study was to build an IDS that is comprehensive, diverse, and robust to detect a wide range of cyber-attacks. The empirical analysis revealed that the classifiers could successfully detect a number of attacks from different datasets. However, certain attack types have minority instances in datasets that were either difficult to detect by most classifiers or were not detected completely by any of the classifiers stacks. This section will review the classifiers against different threat models to understand why a particular classifier performed well or poorly against a certain attack type.

Section 4 results show that for the BoT_IoT dataset, CNN from individual and “Autoencoder+TCN” from hybrid classifiers provided the best classification results. In our experiments, we adopted 1D-CNN to build these classifiers. 1D-CNNs are known to perform well on sequential data (Z. Chen et al., 2017) and are widely used for network traffic analysis which consists of temporal ordering (Berman et al., 2019). Some of the advantages of CNN include their ability to share parameters, small footprint, and easy deployment in embedded systems (Fu et al., 2019). They achieve high performance in learning from raw data (Samy et al., 2020). On the other hand, MLP requires a large number of trainable parameters in hidden layers, making it difficult to work well with high-dimensional datasets (Rezaei and Liu, 2019). For high dimensional datasets, 1D-CNN performs better by using convolutional layers, which considerably reduces trainable parameters using a specified kernel size.

Network intrusions are of different types. DoS and DDoS have similar flooding motives to bring critical services and infrastructure down; however, a DDoS attack is more lethal and challenging to prevent because thousands or possibly millions of compromised bots simultaneously flood a service. Some DDoS attacks slowly deplete server resources and are hard to detect, like the "Slowloris" attack (Shorey et al., 2018), whereas others like "Jamming" attacks can quickly drain resources and have serious consequences (Fadele et al., 2019). Many network attacks consist of sequential characteristics; not every classifier is smart enough to consider temporal and sequential dependencies. Sequential characteristics of network attacks provide important information, and it must be considered (Liu et al., 2019). Keeping this consideration in mind, we adopted TCN, BRNN, and LSTM in our classifier selection because they perform well on temporal data. When a classifier predicts a new instance, it must consider current and the previous data and decide based on the temporal characteristics. Unlike RNN and LSTM, where predictions must wait for predecessor results (Bai et al., 2018), a TCN allows parallelization because they do not maintain any hidden states. Eventually, this allows better hardware GPU optimization during training stage (Gehring et al., 2017). As opposed to CNN, a TCN is more receptive to input range by utilizing dilation rate (Bai et al., 2018). Some researchers argue that CNN has limitations in predicting time-series data due to inconsistent input vectors and inconsistent input and output sizes (Yan et al., 2020). The authors also stated that TCNs simple structure and large receptive fields outperform their prediction accuracies and efficiencies compared to RNN and LSTM. Lastly, TCN’s low memory footprints and low computational complexity to make predictions make it a perfect choice on resource constraint devices for embedded classification (Ingolfsson et al., 2020).

The empirical results in Section 4 also reveal that certain attack types were either complex (e.g., gafgyt_tcp and gafgyt_udp in UNSW_NB15) to detect or had minority instances in the dataset, making it difficult for the classifiers to detect. Table 24 presents the detail of minority instances in each dataset and their percentage of total instances in the dataset. The percentages reflect that the datasets are imbalanced, and the instance counts are bare minimum compared to the full dataset. When training the models, these instances were further divided into training, validation, and test sets, making them even smaller for the models to get trained on these traffic patterns. For example, In the CICIDS2017 dataset, the Heartbleed attack only has 11 instances out of a total of 2.8 million instances in the dataset. We trained the classifier on eight instances only, keeping three instances for testing purposes. With such a small number of instances, it is challenging for the classifiers to detect these or similar minority attack types. Such attack classes are known classes, but the models are not confident and make low predictions (Lakkaraju et al., 2016). To improve classification results for these minority classes, we suggest alternative approaches such as (a) tuning the model hyperparameters, introducing additional dropout and hidden layers, and using regularization parameters (e.g., L1 and L2). These enhancements will mitigate model overfitting issues and help improve the accuracies, (b) Instead of training the model on all features, extract the top contributing features using commonly known dimensionality reduction techniques (e.g., PCA, ICA, and t-SNE). With the least number of features, the classifier will likely detect a slight variation in the input dataset.

6. Summary, conclusion, and future work

6.1. Summary

This study presents a comparative analysis of various benchmark datasets and deep learning models commonly proposed by researchers for IDSs in an IoT environment. Our empirical results reveal numerous valuable findings which highlight various strengths and weaknesses in adopting a certain deep learning model.

- IoTs operate in a resource constraint environment (Liang et al., 2020), whereas certain algorithms proposed by researchers require excessive computational power and resources to execute them. Without paying attention to resource limitations, certain algorithms may require long hours to converge, millions of trainable parameters to adjust, and require high storage space to save the models. Even if the resource and computational limitations are solved by utilizing Cloud or similar architecture; nonetheless, certain models such as "*Bi-directional Recurrent Neural Network (BRNN)*" produce very poor results.
- Our analysis reveals another important consideration of the models producing "*bias*" results when only trained on a single dataset and a single classifier; this is relevant to the paper [59]. Researchers tuned the classifier for a certain dataset to achieve higher performance metrics; however, those settings are model and dataset-specific and may not work for other datasets, as shown in our experimental results above. For example, one of our models, "*Autoencoder+TCN,*" performed well on the "*BoT_IoT*" dataset with almost 100% detection rate but performed very poorly on other datasets "*UNSW_NB15 and NSL-KDD*" when used with the same model settings.
- Our results of hybrid classifiers reveal that "*Autoencoder+TCN*" can mostly converge in less than 20 min which makes it a

Table 24
Minority instances in datasets.

Dataset	Minority classes	Instances	% Age of total instances
Bot_IoT	Theft	79	0.0022%
	Normal	477	0.0130%
CICIDS2017	Web Attack XSS	652	0.0231%
	Infiltration	36	0.0013%
	Web Attack SQL Injection	21	0.0007%
	Heartbleed	11	0.0004%
NSL_KDD	R2L	3704	2.4940%
	U2R	119	0.0801%
UNSW_NB15	Analysis	2677	1.0389%
	Backdoor	2329	0.9039%
	Shellcode	1511	0.5864%
	Worms	174	0.0675%
KDD CUP 99	R2L	1126	0.2279%
	U2R	52	0.0105%

good candidate for resource constraint environments. With such a short convergence time, *Autoencoder+TCN* can frequently accommodate new threats, information, and attacks frequently without significant overhead impact.

- Finally, our analysis highlights an important consideration for future researchers to adopt a holistic approach when selecting a dataset, classifier and training their proposed deep learning models.

Protecting the IoT environment from cyber-attacks has attracted many researchers to propose solutions using state-of-the-art deep learning models. A model that generates significant false-negative results compromises security and allows breaches to happen without notice. This study analyzed various benchmark datasets recently adopted by the researchers. We focused our study on analyzing deep learning algorithms only and not traditional machine learning algorithms due to various resource limitations in traditional machine learning approaches.

6.2. Conclusion

Our experimental analysis captured numerous valuable findings that helped us find the weak and robust performing algorithms and models. Our results clearly show examples of poor-performing classifiers (e.g., *Autoencoder + BRNN*), which took hours to train and still yield poor classification results. On the other hand, some classifiers (e.g., *Autoencoder + TCN*) converge quickly and yield the best results, achieving two important goals, performance, and accuracy.

Although deep learning provides significant benefits in detecting intrusions with minimum to no human involvement in selecting features or labeling traffic, still the trade-off between memory, computational challenges, and energy constraints may make deep learning impractical on low-end targeted IoT devices (Lane et al., 2015). As a result, researchers have started proposing solutions to offload computationally expensive model tasks to edge or cloud level (Roopak et al., 2019; Ferrag et al., 2020; Cuervo et al., 2010). Since the edge devices are close to end-users, they would provide better computational power and efficiencies in detecting malicious traffic. Nowadays, we see many applications in IoT devices running on the cloud. However, such solutions also bring further challenges related to privacy issues and network fluctuation issues such as throughput and latency. Researchers who propose offloading model execution to the edge, fog, or cloud level, also need to consider the usage expenses (i.e. monetary amount) and energy consumption when devices communicate to cloud.

Although our extensive results provide exciting findings on best-performing classifiers and their capabilities to successfully detect a vast number of cyber-attacks, we believe additional parameters' optimization specific to each model would yield better results when detecting individual attacks, specifically when detecting minority classes.

6.3. Future works and trends

The work performed in this paper reveals many insights for future researchers. Some of them are the followings:

- We would like to dig deep into good-performing models from this paper and look for the best settings to further optimize model performance and attack detection capabilities. We will also try to find a good match between the datasets and the models. We hope that our analysis will help future researchers to gain insights and guidance as to which algorithms perform well on a specific dataset and which do not. The findings can be used as part of future pre-training activities similar to embedding models already used in other domains.
- Another important consideration for future researchers is choosing the IDS dataset, which is recent and has a modern-day network traffic footprint. Researchers who build their classifiers on old datasets such as KDD CUP 99 (Dushimimana et al., 2020; Zong and Huang, 2019), or NSL-KDD (Ma et al., 2020; Das et al., 2019; Otoum et al., 2019) would not provide comprehensive protection against the latest known and zero-day attacks. These datasets were generated over a decade ago, and many network features and devices have changed since then. Anomaly-based detection depends on data, and critical decisions are made once they start reporting attacks in the environment. If an IDS yields a false negative result, it may have serious consequences because a real attack has happened, causing the environment to be compromised, while IDS could not detect it. For most models to run in a live environment, retraining with the latest and new information footprint is necessary. This is especially true in today's changing technology world, where new cyber-attacks emerge every day. In a live environment, we need to make sure that the model can be retrained frequently. Any model which takes long hours to retrain would not provide a workable solution.
- When using multiple models and datasets, it will be easy to directly compare and contrast models, e.g., individual vs. hybrid models, with the same settings and classifier types.
- When designing an IDS for resource constraint devices, it is important to consider models composed of fewer neurons, trainable parameters, and hidden layers. Devices with fewer resources tend to adapt models that have a minimum resource footprint. Deep learning models requiring heavy computations may produce the best detection results but would be impractical from running in an IoT environment.

- Due to the limited resources of IoT devices, changing network traffic and attack patterns require incremental model optimization at every stage of the IDS. For example, (a) choosing the right dataset is critical to train the model, (b) choosing the right classifier that is resource-efficient, (c) model's ability to train on new variants recursively, (d) optimize model parameters to enhance detection capabilities, (e) deploying models on a network layer that can optimize performance while putting least burden on IoT devices.
- The empirical results in Section 4 reveal that most of the classifiers performed well in detecting large-scale attacks such as DDoS, DoS, and Mirai. However, they could not detect other attacks with minority instances such as Heatbleed, Infiltration, and Web Attack SQL Injection. For future work, we will focus on techniques and algorithms that would perform better against a specific threat model (e.g., TCN vs. DDoS, autoencoder+TCN vs. DDoS)

Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Rasheed Ahmad: Conceptualization, Methodology, Software, Visualization, Writing – original draft. **Izzat Alsmadi:** Conceptualization, Methodology, Validation, Writing – review & editing. **Wasim Alhamdani:** Supervision, Writing – review & editing. **Lo'ai Tawalbeh:** Supervision, Writing – review & editing.

References

Ahmad, R., Alsmadi, I., 2021. Machine learning approaches to IoT security: a systematic literature review. Internet Things 14, 100365. doi:10.1016/j.iot.2021.100365.

Ahmad, Z., Khan, A.S., Shiang, C.W., Abdullah, J., Ahmad, F., 2020. Network intrusion detection system: a systematic study of machine learning and deep learning approaches. Trans. Emerg. Telecommun. Technol. e4150. doi:10.1002/ett.4150, n/a.

Akbari, A., Awais, M., Bashar, M., Kittler, J., 2021. How does loss function affect generalization performance of deep learning? Application to human age estimation. In: Proceedings of the International Conference on Machine Learning. Presented at the International Conference on Machine Learning, PMLR, pp. 141–151.

Alabdulatif, A., Khalil, I., Forkan, A.R.M., Atiquzzaman, M., 2019. Real-time secure health surveillance for smarter health communities. IEEE Commun. Mag. 57, 122–129. doi:10.1109/MCOM.2017.1700547.

Alsamiri, J., Alsubhi, K., 2019. Internet of things cyber attacks detection using machine learning. IJACSA 10. doi:10.14569/IJACSA.2019.0101280.

Anthi, E., Williams, L., Słowińska, M., Theodorakopoulos, G., Burnap, P., 2019. A supervised intrusion detection system for smart home IoT devices. IEEE Internet Things J. 6, 9042–9053. doi:10.1109/JIOT.2019.2926365.

Aydos, M., Vural, Y., Tekerek, A., 2019. Assessing risks and threats with layered approach to internet of things security. Meas. Control 52, 338–353. doi:10.1177/0020294019837991.

Aygun, R.C., Yavuz, A.G., 2017. Network anomaly detection with stochastically improved autoencoder based models. In: Proceedings of the IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud), pp. 193–198. doi:10.1109/CSCloud.2017.39 Presented at the 2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud).

Bai, S., Kolter, J. Z., & Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*.

Berman, D.S., Buczak, A.L., Chavis, J.S., Corbett, C.L., 2019. A survey of deep learning methods for cyber security. Information 10, 122. doi:10.3390/info10040122.

Binbusayyis, A., Vaiyapuri, T., 2019. Identifying and benchmarking key features for cyber intrusion detection: an ensemble approach. IEEE Access 7, 106495–106513. doi:10.1109/ACCESS.2019.2929487.

Chaabouni, N., Mosbah, M., Zemmari, A., Sauvignac, C., Faruki, P., 2019. Network intrusion detection for IoT security based on learning techniques. IEEE Commun. Surv. Tutor. 21, 2671–2701. doi:10.1109/COMST.2019.2896380.

Chang, S., Zhang, Y., Han, W., Yu, M., Guo, X., Tan, W., Cui, X., Hasegawa-Johnson, M., Huang, T. S., Witbrock, M.. 1). Dilated recurrent neural networks.

Charyyev, B., Gunes, M.H., 2020. Detecting anomalous IoT traffic flow with locality sensitive hashes. In: Proceedings of the GLOBECOM IEEE Global Communications Conference, pp. 1–6. doi:10.1109/GLOBECOM42002.2020.9322559 Presented at the GLOBECOM 2020 - 2020 IEEE Global Communications Conference.

Chaudhary, P., Gupta, B.B., 2019. DDoS detection framework in resource constrained internet of things domain. In: Proceedings of the IEEE 8th Global Conference on Consumer Electronics (GCCE), pp. 675–678. doi:10.1109/GCCE46687.2019.9015465 Presented at the 2019 IEEE 8th Global Conference on Consumer Electronics (GCCE).

Chen, C., Ghassami, A., Mohan, S., Kiyavash, N., Bobba, R.B., Pellizzoni, R., & Yoon, M. (2017). A Reconnaissance Attack Mechanism for Fixed-Priority Real-Time Systems. ArXiv, abs/1705.02561.

Chen, Z., He, K., Li, J., Geng, Y., 2017. Seq2Img: a sequence-to-image based approach towards IP traffic classification using convolutional neural networks. In: Proceedings of the IEEE International Conference on Big Data (Big Data), pp. 1271–1276. doi:10.1109/BigData.2017.8258054 Presented at the 2017 IEEE International Conference on Big Data (Big Data).

Cisco Annual Internet Report (2018–2023) white paper [WWW Document], 2020. Cisco. URL https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html (accessed 8.10.20).

Cuervo, E., Balasubramanian, A., Cho, D., Wolman, A., Saroiu, S., Chandra, R., Bahl, P., 2010. MAUI: making smartphones last longer with code offload. In: Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services, MobiSys '10. Association for Computing Machinery, New York, NY, USA, pp. 49–62. doi:10.1145/1814433.1814441.

Das, S., Mahfouz, A.M., Venugopal, D., Shiva, S., 2019. DDoS intrusion detection through machine learning ensemble. In: Proceedings of the IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C). IEEE, Sofia, Bulgaria, pp. 471–477. doi:10.1109/QRS-C.2019.00090.

De Michele, R., Furini, M., 2019. IoT healthcare: benefits, issues, and challenges. In: Proceedings of the 5th EAI International Conference on Smart Objects and Technologies for Social Good, GoodTechs '19. Association for Computing Machinery, New York, NY, USA, pp. 160–164. doi:10.1145/3342428.3342693.

DeBeck, C., Chung, J., McMillen, D., 2019. I can not believe Mirais: tracking the infamous IoT malware [WWW Document]. Secur. Intell.. URL https://securityintelligence.com/posts/i-cant-believe-mirais-tracking-the-infamous-iot-malware-2/ .

Derhab, A., Aldweesh, A., Emam, A.Z., Khan, F.A., 2020. Intrusion detection system for internet of things based on temporal convolution neural network and efficient feature engineering. Wireless Commun. Mob. Comput. 2020, 1–16. doi:10.1155/2020/6689134.

Desai, B.A., Divakaran, D.M., Nevat, I., Peter, G.W., Gurusamy, M., 2019. A feature-ranking framework for IoT device classification. In: Proceedings of the 11th International Conference on Communication Systems Networks (COMSNETS), pp. 64–71. doi:10.1109/COMSNETS.2019.8711210 Presented at the 2019 11th International Conference on Communication Systems Networks (COMSNETS).

Cui, Z., Ke, R., Pu, Z., Wang, Y., 2019. Deep bidirectional and unidirectional LSTM recurrent neural network for network-wide traffic speed prediction. Retrieved from https://ui.adsabs.harvard.edu/abs/2018arXiv180102143C

Dhamija, A. R., Günther, M., & Boult, T. E. (2018). Reducing network agnostophobia. *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 9175–9186.

Divekar, A., Parekh, M., Savla, V., Mishra, R., Shirole, M.. Benchmarking datasets for Anomaly-based Network Intrusion Detection.

Dushimimana, A., Tao, T., Kindong, R., Nishyirimbere, A., 2020. Bi-directional recurrent neural network for intrusion detection system (IDS) in the internet of things (IoT). IJAERS 7, 524–539. doi:10.22161/ijaers.73.68.

Elejla, O.E., Anbar, M., Belaton, B., Alijla, B.O., 2018. Flow-based IDS for ICMPv6-based DDoS attacks detection. Arab. J. Sci. Eng. 43, 7757–7775. doi:10.1007/s13369-018-3149-7.

Fadele, A., Othman, M., Hashem, I., Yaqoob, I., Imran, M., Shoaib, M., 2019. A novel countermeasure technique for reactive jamming attack in internet of things. Multimed. Tools Appl. 78. doi:10.1007/s11042-018-6684-z.

Feng, Z., Xu, C., Tao, D., 2019. Self-supervised representation learning from multi-domain data. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) doi:10.1109/ICCV.2019.00334.

Ferrag, M.A., Maglaras, L., Ahmim, A., Derdour, M., Janicke, H., 2020. RDTIDS: rules and decision tree-based intrusion detection system for internet-of-things networks. Future Internet 12, 44. doi:10.3390/fi12030044, Basel.

Fu, N., Kamili, N., Huang, Y., Shi, J., 2019. A novel deep intrusion detection model based on a convolutional neural network. Aust. J. Intell. Inf. Process. Syst..

García-Teodoro, P., Díaz-Verdejo, J., Maciá-Fernández, G., Vázquez, E., 2009. Anomaly-based network intrusion detection: Techniques, systems and challenges. Computers & Security 28 (1), 18–28. https://doi.org/10.1016/j.cose.2008.08.003.

Ge, M., Fu, X., Syed, N., Baig, Z., Teo, G., Robles-Kelly, A., 2019. Deep learning-based intrusion detection for IoT networks. In: Proceedings of the IEEE 24th Pacific Rim International Symposium on Dependable Computing (PRDC), pp. 256–25609. doi:10.1109/PRDC47002.2019.00056 Presented at the 2019 IEEE 24th Pacific Rim International Symposium on Dependable Computing (PRDC).

Gehring, J., Auli, M., Grangier, D., Yarats, D., Dauphin, Y. N., 2017. Convolutional sequence to sequence learning. Proceedings of the 34th International Conference on Machine Learning - 70, 1243–1252.

Hady, A.A., Ghubaish, A., Salman, T., Unal, D., Jain, R., 2020. Intrusion detection system for healthcare systems using medical and network data: a comparison study. IEEE Access 8, 106576–106584. doi:10.1109/ACCESS.2020.3000421.

Haider, S., Akhunzada, A., Mustafa, I., Patel, T.B., Fernandez, A., Choo, K.K.R., Iqbal, J., 2020. A deep CNN ensemble framework for efficient DDoS attack detection in software defined networks. IEEE Access 8, 53972–53983. doi:10.1109/ACCESS.2020.2976908.

Han, S., Shen, H., Philipose, M., Agarwal, S., Wolman, A., Krishnamurthy, A., 2016. MCDNN: an approximation-based execution framework for deep stream processing under resource constraints. In: Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services, MobiSys '16. Association for Computing Machinery, New York, NY, USA, pp. 123–136. doi:10.1145/2906388.2906396.

Hassen, M., Chan, P.K., 2020. Unsupervised open set recognition using adversarial autoencoders. In: Proceedings of the 19th IEEE International Conference on Machine Learning and Applications (ICMLA), pp. 360–365. doi:10.1109/ICMLA51294.2020.00064 Presented at the 2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA).

Hayashi, T., Watanabe, S., Toda, T., Hori, T., Le Roux, J., & Takeda, K. (2016, September). Bidirectional LSTM-HMM hybrid system for polyphonic sound event detection. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)* (pp. 35-39).

Humayun, M., Jhanjhi, N., Hamid, B., Ahmed, G., 2020. Emerging smart logistics and transportation using IoT and blockchain. IEEE Internet Things Mag. 3, 58–62. doi:10.1109/IOTM.0001.1900097.

Hwang, R.H., Peng, M.C., Huang, C.W., Lin, P.C., Nguyen, V.L., 2020. An unsupervised deep learning model for early network traffic anomaly detection. IEEE Access 8, 30387–30399. doi:10.1109/ACCESS.2020.2973023.

Hwang, R.H., Peng, M.C., Nguyen, V.L., Chang, Y.L., 2019. An LSTM-based deep learning approach for classifying malicious traffic at the packet level. Appl. Sci. 9. doi:10.3390/app9163414.

Ibitoye, O., Shafiq, O., Matrawy, A., 2019. Analyzing adversarial attacks against deep learning for intrusion detection in IoT networks. In: Proceedings of the IEEE Global Communications Conference (GLOBECOM), pp. 1–6. doi:10.1109/GLOBECOM38437.2019.9014337 Presented at the 2019 IEEE Global Communications Conference (GLOBECOM).

Ingolfsson, T. M., Hersche, M., Wang, X., Kobayashi, N., Cavigelli, L., Benini, L.. EEG-TCNet: An Accurate Temporal Convolutional Network for Embedded Motor-Imagery Brain–Machine Interfaces.

Ingre, B., Yadav, A., 2015. Performance analysis of NSL-KDD dataset using ANN. 10.1109/SPACES.2015.7058223

Jaidka, H., Sharma, N., Singh, R., 2020. Evolution of IoT to IIoT: applications & challenges (SSRN Scholarly Paper No. ID 3603739). Social Science Research Network, Rochester, NY. 10.2139/ssrn.3603739

Jouppi, N.P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., Borchers, A., Boyle, R., Cantin, P., Chao, C., Clark, C., Coriell, J., Daley, M., Dau, M., Dean, J., Gelb, B., Ghaemmaghami, T.V., Gottipati, R., Gulland, W., Hagmann, R., Ho, C.R., Hogberg, D., Hu, J., Hundt, R., Hurt, D., Ibarz, J., Jaffey, A., Jaworski, A., Kaplan, A., Khaitan, H., Killebrew, D., Koch, A., Kumar, N., Lacy, S., Laudon, J., Law, J., Le, D., Leary, C., Liu, Z., Lucke, K., Lundin, A., MacKean, G., Maggiore, A., Mahony, M., Miller, K., Nagarajan, R., Narayanaswami, R., Ni, R., Nix, K., Norrie, T., Omernick, M., Penukonda, N., Phelps, A., Ross, J., Ross, M., Salek, A., Samadiani, E., Severn, C., Sizikov, G., Snelham, M., Souter, J., Steinberg, D., Swing, A., Tan, M., Thorson, G., Tian, B., Toma, H., Tuttle, E., Vasudevan, V., Walter, R., Wang, W., Wilcox, E., Yoon, D.H., 2017. In-datacenter performance analysis of a tensor processing unit. In: Proceedings of the 44th Annual International Symposium on Computer Architecture. ACM, Toronto ON Canada, pp. 1–12. doi:10.1145/3079856.3080246 Presented at the ISCA '17: The 44th Annual International Symposium on Computer Architecture.

Karie, N.M., Sahri, N.M., Haskell-Dowland, P., 2020. IoT threat detection advances, challenges and future directions. In: Proceedings of the Workshop on Emerging Technologies for Security in IoT (ETSecIoT), pp. 22–29. doi:10.1109/ETSecIoT50046.2020.00009.

Kayyali, B., Knott, D., Van Kuiken, S., 2013. The big-data revolution in US health care: accelerating value and innovation | McKinsey [WWW Document]. URL https://www.mckinsey.com/industries/healthcare-systems-and-services/our-insights/the-big-data-revolution-in-us-health-care (accessed 4.17.21).

Kelly, C., Pitropakis, N., McKeown, S., Lambrinoudakis, C., 2020. Testing and hardening IoT devices against the Mirai botnet. In: Proceedings of the International Conference on Cyber Security and Protection of Digital Services (Cyber Security), pp. 1–8. doi:10.1109/CyberSecurity49315.2020.9138887 Presented at the 2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security).

Kumar, A., Lim, T. J., 2019. EDIMA: Early Detection of IoT Malware Network Activity Using Machine Learning Techniques. *2019 IEEE 5th World Forum on*. Internet of Things (WF-IoT),, 289–294. https://doi.org/10.1109/WF-IoT.2019.8767194.

Koroniotis, N., Moustafa, N., Sitnikova, E., Turnbull, B., 2019. Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset. Future Generation Computer Systems 100, 779–796. https://doi.org/10.1016/j.future.2019.05.041.

Lai, Y., Zhou, K., Lin, S., Lo, N., 2019. Flow-based anomaly detection using multilayer perceptron in software defined networks. In: Proceedings of the 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), pp. 1154–1158. doi:10.23919/MIPRO.2019.8757199 Presented at the 2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO).

Lakkaraju, H., Kamar, E., Caruana, R., Horvitz, E., 2016. Discovering unknown unknowns of predictive models. In: Proceedings of the 30th Conference on Neural Information Processing Systems, p. 5.

Lane, N., Bhattacharya, S., Mathur, A., Forlivesi, C., Kawsar, F., 2016. DXTK: enabling resource-efficient deep learning on mobile and embedded devices with the deepX toolkit. In: Proceedings of the 8th EAI International Conference on Mobile Computing, Applications and Services, MobiCASE'16. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), pp. 98–107. doi:10.4108/eai.30-11-2016.2267463 Brussels, BEL.

Lane, N.D., Bhattacharya, S., Georgiev, P., Forlivesi, C., Kawsar, F., 2015. An early resource characterization of deep learning on wearables, smartphones and internet-of-things devices. In: Proceedings of the International Workshop on Internet of Things Towards Applications, IoT-App '15. Association for Computing Machinery, New York, NY, USA, pp. 7–12. doi:10.1145/2820975.2820980.

Liang, F., Yu, W., Liu, X., Griffith, D., Golmie, N., 2020. Toward edge-based deep learning in industrial internet of things. IEEE Internet Things J. 7, 4329–4341. doi:10.1109/JIOT.2019.2963635.

Liang, X., Znati, T., 2019. A Long Short-Term Memory Enabled Framework for DDoS Detection. In: Proceedings of the IEEE Global Communications Conference (GLOBECOM), pp. 1–6. doi:10.1109/GLOBECOM38437.2019.9013450 Presented at the 2019 IEEE Global Communications Conference (GLOBECOM).

Liu, J., Liu, S., Zhang, S., 2019. Detection of IoT botnet based on deep learning. In: Proceedings of the Chinese Control Conference (CCC), pp. 8381–8385. doi:10.23919/ChiCC.2019.8866088 Presented at the 2019 Chinese Control Conference (CCC).

Liu, M., Xue, Z., Xu, X., Zhong, C., Chen, J., 2019. Host-based intrusion detection system with system calls: review and future trends. ACM Comput. Surv. 51, 1–36. doi:10.1145/3214304.

Liu, Y., Zhou, Y., Wen, S., Tang, C., 2014. A strategy on selecting performance metrics for classifier evaluation. Int. J. Mob. Comput. Multimed. Commun. 6, 20–35. doi:10.4018/IJMCMC.2014100102.

Ma, L., Chai, Y., Cui, L., Ma, D., Fu, Y., Xiao, A., 2020. A deep learning-based DDoS detection framework for internet of things. In: Proceedings of the ICC IEEE International Conference on Communications (ICC), pp. 1–6. doi:10.1109/ICC40277.2020.9148944.

Malik, J., Akhunzada, A., Bibi, I., Imran, M., Musaddiq, A., Kim, S.W., 2020. Hybrid deep learning: an efficient reconnaissance and surveillance detection mechanism in SDN. IEEE Access 8, 134695–134706. doi:10.1109/ACCESS.2020.3009849.

McHugh, J., 2000. Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory. ACM Trans. Inf. Syst. Secur. 3, 262–294. doi:10.1145/382912.382923.

Meidan, Y., Bohadana, M., Mathov, Y., Mirsky, Y., Breitenbacher, D., Shabtai, A., Elovici, Y., 2018. N-BaIoT: network-based detection of IoT botnet attacks using deep autoencoders. IEEE Pervasive Comput. 17, 12–22. doi:10.1109/MPRV.2018.03367731.

Mera, C., Branch, J.W., 2014. A survey on class imbalance learning on automatic visual inspection. IEEE Lat. Am. Trans. 12, 657–667. doi:10.1109/TLA.2014.6868867.

Mergendahl, S., Li, J., 2020. Rapid: robust and adaptive detection of distributed denial-of-service traffic from the internet of things. In: Proceedings of the IEEE Conference on Communications and Network Security (CNS), pp. 1–9. doi:10.1109/CNS48642.2020.9162278 Presented at the 2020 IEEE Conference on Communications and Network Security (CNS).

Mirsky, Y., Doitshman, T., Elovici, Y., & Shabtai, A. (2018). Kitsune: an ensemble of autoencoders for online network intrusion detection. arXiv preprint arXiv:1802.09089.

Mohammadi, M., Al-Fuqaha, A., Sorour, S., Guizani, M., 2018. Deep learning for IoT big data and streaming analytics: a survey. IEEE Commun. Surv. Tutor. 20, 2923–2960. doi:10.1109/COMST.2018.2844341.

Moussa, M.M., Alazzawi, L., 2020. Cyber attacks detection based on deep learning for cloud-dew computing in automotive IoT applications. In: Proceedings of the IEEE International Conference on Smart Cloud (SmartCloud), pp. 55–61. doi:10.1109/SmartCloud49737.2020.00019 Presented at the 2020 IEEE International Conference on Smart Cloud (SmartCloud).

Moustafa, N., Slay, J., 2015. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In: Proceedings of the Military Communications and Information Systems Conference (MilCIS) doi:10.1109/MilCIS.2015.7348942, Presented at the 2015 Military Communications and Information Systems Conference (MilCIS).

Moustafa, N., Turnbull, B., Choo, K.R., 2019. An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of internet of things. IEEE Internet Things J. 6, 4815–4830. doi:10.1109/JIOT.2018.2871719.

Nagisetty, A., Gupta, G.P., 2019. Framework for detection of malicious activities in IoT networks using keras deep learning library. In: Proceedings of the 3rd International Conference on Computing Methodologies and Communication (ICCMC), pp. 633–637. doi:10.1109/ICCMC.2019.8819688 Presented at the 2019 3rd International Conference on Computing Methodologies and Communication (ICCMC).

Narla, S.R.K., Stowell, H.G., 2019. Connected and automated vehicles. Institute of transportation engineers. ITE J. 28–33. https://search.proquest.com/docview/2197276353?accountid=10378.

Naveed, K., Wu, H., 2020. Poster: a semi-supervised framework to detect botnets in IoT devices. In: Proceedings of the IFIP Networking Conference (Networking), pp. 649–651 Presented at the 2020 IFIP Networking Conference (Networking).

Ng, W., Minasny, B., Mendes, W., De, S., Demattê, J.A.M., 2019. Estimation of effective calibration sample size using visible near infrared spectroscopy: deep learning vs machine learning. SOIL Discuss. 1–21. doi:10.5194/soil-2019-48.

Otoum, Y., Liu, D., Nayak, A., 2019. DL-IDS: a deep learning–based intrusion detection framework for securing IoT. Trans. Emerg. Telecommun. Technol. doi:10.1002/ett.3803.

Rezaei, S., Liu, X., 2019. Deep learning for encrypted traffic classification: an overview. IEEE Commun. Mag. 57, 76–81. doi:10.1109/MCOM.2019.1800819.

Roopak, M., Tian, G.Y., Chambers, J., 2020a. Multi-objective-based feature selection for DDoS attack detection in IoT networks. IET Netw. 9, 120–127. doi:10.1049/iet-net.2018.5206.

Roopak, M., Tian, G.Y., Chambers, J., 2020b. An intrusion detection system against DDoS attacks in IoT networks. In: Proceedings of the 10th Annual Computing and Communication Workshop and Conference (CCWC), pp. 0562–0567. doi:10.1109/CCWC47524.2020.9031206 Presented at the 2020 10th Annual Computing and Communication Workshop and Conference (CCWC).

Roopak, M., Yun Tian, G., Chambers, J., 2019. Deep learning models for cyber security in IoT networks. In: Proceedings of the IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), pp. 0452–0457. doi:10.1109/CCWC.2019.8666588.

Said Elsayed, M., Le-Khac, N.-A., Dev, S., Jurcut, A.D., 2020. Network anomaly detection using LSTM based autoencoder. In: Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks. ACM, Alicante Spain, pp. 37–45. doi:10.1145/3416013.3426457 Presented at the MSWiM '20: 23rd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems.

Samy, A., Yu, H., Zhang, H., 2020. Fog-based attack detection framework for internet of things using deep learning. IEEE Access 8, 74571–74585. doi:10.1109/ACCESS.2020.2988854.

Sarker, I.H., Shahriar, B., Watters, P., Ng, A., 2020. Cybersecurity data science: an overview from machine learning perspective. J. Big Data 7. doi:10.1186/s40537-020-00318-5, Heidelberg.

Sharafaldin, I., Habibi Lashkari, A., Ghorbani, A.A., 2018. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: Proceedings of the 4th International Conference on Information Systems Security and Privacy. SCITEPRESS - Science and Technology Publications, Funchal, Madeira, Portugal, pp. 108–116. doi:10.5220/0006639801080116 Presented at the 4th International Conference on Information Systems Security and Privacy.

Shorey, T., Subbaiah, D., Goyal, A., Sakxena, A., Mishra, A.K., 2018. Performance comparison and analysis of slowloris, goldenEye and xerxes DDoS attack Tools. In: Proceedings of the International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 318–322. doi:10.1109/ICACCI.2018.8554590 Presented at the 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI).

Shurman, M., Khrais, R., Yateem, A.R., 2020. DoS and DDoS attack detection using deep learning and IDS. Int. Arab J. Inf. Technol. 17, 655–661. doi:10.34028/iajit/17/4A/10.

Soe, Y.N., Santosa, P.I., Hartanto, R., 2019. DDoS Attack detection based on simple ANN with SMOTE for IoT environment. In: Proceedings of the Fourth International Conference on Informatics and Computing (ICIC), pp. 1–5. doi:10.1109/ICIC47613.2019.8985853.

Sriram, S., Vinayakumar, R., Alazab, M., KP, S., 2020. Network flow based IoT botnet attack detection using deep learning. In: Proceedings of the IEEE INFOCOM IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 189–194. doi:10.1109/INFOCOMWKSHPS50562.2020.9162668 Presented at the IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS).

Strubell, E., Ganesh, A., McCallum, A.. Energy and Policy Considerations for Deep Learning in NLP.

Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A.A., 2009. A detailed analysis of the KDD CUP 99 data set. In: Proceedings of the IEEE Symposium on Computational Intelligence for Security and Defense Applications, pp. 1–6. doi:10.1109/CISDA.2009.5356528.

Veena, K., 2018. A Survey on Network Intrusion Detection. International Journal of Scientific Research in Science, Engineering and Technology 4 (8). https://doi.org/10.32628/IJSRSET1848160.

Wang, S., Minku, L.L., Yao, X., 2018. A systematic study of online class imbalance learning with concept drift. IEEE Trans. Neural. Netw. Learn. Syst. 29, 4802–4821. doi:10.1109/TNNLS.2017.2771290.

Wojtowytsch, S., E, W., 2020. Can shallow neural networks beat the curse of dimensionality? A mean field training perspective. arXiv:2005.10815 [cs, math, stat].

Wu, C.W., 2020. Simplifying neural networks via look up tables and product of sums matrix factorizations. In: Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1–11. doi:10.1109/ISCAS45731.2020.9180985 Presented at the 2020 IEEE International Symposium on Circuits and Systems (ISCAS).

Yan, J., Mu, L., Wang, L., Ranjan, R., Zomaya, A.Y., 2020. Temporal convolutional networks for the advance prediction of ENSO. Sci. Rep. 10, 8055. doi:10.1038/s41598-020-65070-5.

Zhou, H., Hu, Y., Yang, X., Pan, H., Guo, W., Zou, C.C., 2020. A worm detection system based on deep learning. IEEE Access 8, 205444–205454. doi:10.1109/ACCESS.2020.3023434.

Zong, Y., Huang, G., 2019. A feature dimension reduction technology for predicting DDoS intrusion behavior in multimedia internet of things. Multimed. Tools Appl. 1–14. doi:10.1007/s11042-019-7591-7, Dordrecht.

Rasheed Ahmad received his master's degree in 2006 in computer science from the City University of New York (CUNY), New York, and an MBA with specialization in Information Technology from Capella University, Minneapolis in 2016. He is currently pursuing a Ph.D. degree with a cybersecurity specialization at the University of the Cumberlands, USA. His-research interests include the Internet of Things (IoT), intrusion detection systems, cybersecurity, and large-scale attacks

Izzat Alsmadi is an Associate Professor in the department of computing and cyber security at the Texas A&M, San Antonio. He has his master and PhD in Software Engineering from North Dakota State University in 2006 and 2008. He has more than 100 conference and journal publications. His-research interests include: Cyber intelligence, Cyber security, Software security, software engineering, software testing, social networks and software defined networking. He is lead author, editor in several books including: Springer, The NICE Cyber Security Framework Cyber Security Intelligence and Analytics, 2019, Practical Information Security: A Competency-Based Education Course, 2018, Information Fusion for Cyber-Security Analytics (Studies in Computational Intelligence), 2016. The author is also a member of The National Initiative for Cybersecurity Education (NICE) group, which meets frequently to discuss enhancements on cyber security education at the national level

Wasim Alhamdani is a Professor in the department of computer science at the University of the Cumberlands, Kentucky, USA. He has a PhD in Computer Science from University of East Anglia, Norwich, UK in 1985 and a M.*Sc.* in Computer Science from Loughborough University of Technology, Loughborough, UK in 1981. His-general research interests are in Cyber Security, Cryptography, and Cyber Security Management. His-current research areas include Information Security Mathematical Modeling, Cyber Security Resilient Architecture Design, Ontologies with Cybersecurity, and Ethics with Cryptography use. He is currently a Cyber Security Curriculum Adviser for two international universities

Dr Lo'ai Tawalbeh (IEEE Senior Member) completed his PhD degree in Electrical & Computer Engineering from Oregon State University in 2004, and MSc in 2002 from the same university with GPA 4/4. Dr. Tawalbeh is currently an Associate professor at the department of Computing and Cyber Security at Texas A&M University-San Antonio. Before that he was a visiting researcher at University of California-Santa Barbra. Since 2005 he taught/developed more than 25 courses in different disciplines of computer engineering and science with focus on cyber security for the undergraduate/graduate programs at: NewYork Institute of Technology (NYIT), DePaul's University, and Jordan University of Science and Technology. Dr. Tawalbeh supervised successfully more than 40 graduate student, and he won many research grants and awards with over than 3 Million USD. He has over 120 research publications in refereed international Journals and conferences