Knowledge-Based Systems 279 (2023) 110941

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

A stacking ensemble of deep learning models for IoT intrusion detection

Riccardo Lazzarini [a,*], Huaglory Tianfield [a], Vassilis Charissis [b]

[a] School of Computing, Engineering and Build Environment, Glasgow Caledonian University (GCU), Cowcaddens Road, Glasgow, G4 0BA, UK
[b] School of Arts and Creative Industries, Edinburgh Napier University, 10 Colinton Road, Edinburgh, EH10 5DT, UK

ARTICLE INFO

Article history:
Received 27 March 2023
Received in revised form 27 July 2023
Accepted 26 August 2023
Available online 1 September 2023

Keywords:
Internet of things
Intrusion detection systems
Deep learning
Ensemble learning
Stacking

ABSTRACT

The number of Internet of Things (IoT) devices has increased considerably in the past few years, which resulted in an exponential growth of cyber attacks on IoT infrastructure. As a consequence, the prompt detection of attacks in IoT environments through the use of Intrusion Detection Systems (IDS) has become essential. This article proposes a novel approach to intrusion detection in IoT based on a stacking ensemble of deep learning (DL) models. This approach is named Deep Integrated Stacking for the IoT (DIS-IoT) and it combines four different DL models into a fully connected DL layer, creating a standalone ensemble model. DIS-IoT is evaluated on three open-source datasets, namely ToN_IoT, CICIDS2017 and SWaT, in binary and multi-class classification and compared results with other standard DL methods. Experiments demonstrate that DIS-IoT is capable of a high-level accuracy with a very low False Positive rate (FPR) in all datasets. Results were also compared against other state-of-the-art works available in the literature, which used similar methods on the same ToN_IoT dataset. DIS-IoT achieves comparable performance with others in binary classification and outperforms them in multi-class classification.

1. Introduction

The Internet of Things (IoT) paradigm has pushed modern, interconnected networks to change rapidly with new, small and connection-ready devices being developed everyday. Industry 4.0, Smart Transportation, Smart Homes, and machine-to-machine communication are just some of the areas in which massive amounts of devices are being connected, making traffic running on the Internet much more diverse and heterogeneous [1,2]. The exponential increase of devices has also allowed for an equally large increase of network attack surface, that malicious hackers could potentially exploit. Hence, detecting intrusions in IoT environments has become ever more paramount. As part of a layered defence approach, Network Intrusion Detection Systems (NIDS) have an essential role in identifying attacks before they do any damage. Intrusion detection can be categorized according to its detection methods: signature-based and anomaly-based [3]. Signature-based detection uses traffic signatures to recognize attacks. For this type of technology to work well an up-to-date database of signatures is essential. On the other hand, anomaly-based detection attempts to identify attacks by initially estimating typical traffic and recognizing anomalies whenever the deviation between the monitored traffic and the estimated traffic exceeds a predefined threshold [4]. Signature-based detection works very well for known attacks, but it is not capable to detect new types of attacks until a signature exists in the database. Anomaly-based detection does well at detecting unknown attacks but suffers from large numbers of false positives - i.e. benign traffic being classified as anomalous.

Anomaly-based detection uses many techniques to identify attacks. These can be based on mathematical models and Machine Learning (ML). The latter has become a widely used engine for anomaly-based IDS. Different approaches of ML can be used: supervised learning, semi-supervised learning and unsupervised learning. In supervised learning input data to the classifier is labelled. In other words, the classifier has knowledge of the type of class each sample of the data belongs to. In semi-supervised learning, input data is only partially labelled, while unsupervised learning works on unlabelled data and attempts to make sense of it by identifying similar patterns [5,6]. Deep Learning (DL) is a branch of machine learning that has become widely popular in many fields, including science, finance, medicine and engineering [7]. DL for cybersecurity and particularly for intrusion detection has also become increasingly popular as it allows for a more sophisticated analysis of network traffic and more precise detection of anomalies compared to traditional ML methods [8].

A common paradigm in ML is the use of *ensemble* methods. The concept behind an ensemble method is to combine multiple

* Corresponding author.
E-mail addresses: rlazza200@caledonian.ac.uk (R. Lazzarini), h.tianfield@gcu.ac.uk (H. Tianfield), v.charisis@napier.ac.uk (V. Charissis).

https://doi.org/10.1016/j.knosys.2023.110941

models to tackle the same problem. Instead of relying on a single model to analyse the data, the ensemble focuses on using a set of models and combining the outcomes. Research has demonstrated that using an ensemble of models gives superior accuracy compared to a single model [9]. Several ensemble methods exist, with the most common aggregators being Bagging, Boosting and Stacking [10].

Bagging or Bootstrapping aggregating is a technique that aggregates outcomes from the classification of different samples of data by multiple learners [11]. Boosting is a family of algorithms that aim at improving predictions by training the classifier sequentially. The idea is to have later models correcting wrong predictions obtained by earlier *weak* models [12].

In this article, we present a novel approach based on stacking to detect intrusions in IoT environment. We call this method Deep Integrated Stacking for the IoT or DIS-IoT. Our ensemble model combines four DL algorithms into a further, fully connected, layer of neurons to reach a final classification of the input data. In order to examine the efficacy of our proposed ensemble model, we have evaluated DIS-IoT against other common DL algorithms. Furthermore, we compare DIS-IoT against state of art IDS models based on similar approaches, recently published in the literature on the same datasets. The experiments show that DIS-IoT outperforms the other DL models as well as offering results on par with other research work in the area.

Key contributions of this work can be summarized as follows:

- We propose a novel method for the detection of attacks in IoT network environments which is based on a stacking ensemble. The ensemble model, named DIS-IoT, uses four DL models, a shallow Multilayer Perceptron (MLP), a Deep Neural Network (DNN), and a CNN-based and an LSTM-based models. The primary objective is the detection and classification of attacks in IoT environment.
- We evaluate the model on three open source datasets, namely ToN_IoT, CICIDS2017 and SWaT, offering high level of accuracy with a low False Positive rate (FP) in all scenarios.
- Our method is evaluated on binary and multi-class classification with ToN_IoT and CICIDS2017 and binary for SWaT showing excellent performance in all types of classification.
- We compare results from our experiments on the ToN_IoT dataset against other published work using the same dataset, showing that performance from DIS-IoT is at least on par with other similar models.

The remainder of this article is organized as follows: Section 2 provides a review of the related work in the area of IDS in IoT environment using DL and ensemble learning. In Section 3 we propose our DIS-IoT method for intrusion detection in IoT environments. Datasets and methodology for experiments are explained in Section 4. Results are discussed in Section 5. The conclusion is drawn in Section 6

2. Literature review

In this section, we present a review of recent work. Emphasis is given to research that used DL and ensemble learning as a method to detect attacks in IoT environment.

2.1. Deep learning in IoT intrusion detection

Intrusion detection is a hot topic within the cybersecurity industry [13–16]. In recent years, research work in this area has focused increasingly on DL learning solutions. Many examples of this, have been proposed in several IT domains, such as cloud computing [17,18] but particularly in computer networking [19–26]. Since IoT devices are becoming an integral part of our daily lives, much of the recent IDS research has also proposed DL solutions in this domain. For instance, Latif et al. [27] proposed a novel lightweight solution based on a Dense Random Neural Network (DnRaNN) to detect intrusions in IoT networks. To evaluate their approach, the authors conducted meticulous experiments against the ToN_IoT dataset in binary and multi-class classification scenarios with excellent results. The same dataset is used by Kumar et al. [28] for studying their DL-driven cyber threat modelling framework. The framework is designed to automate the detection and extraction of cyber threats in IoT-enabled Maritime Transportation Systems (MTS). Similar to the previous work they evaluated the solution in both binary and multi-class classification, also achieving promising results. An approach to intrusion detection based on an Adaptive Particle Swarm Optimization Convolutional Neural Network (APSO-CNN) is proposed in [29]. The approach is to use the APSO algorithm to automatically adjust hyper parameters of the one-dimensional CNN. They evaluated the solution on the N-BaIoT [30] dataset and compared results against three other models. The solution provided the highest scores in all metrics. Other studies involve CNN-based solutions. Although CNN was originally designed to solve problems in computer vision applications (e.g. facial or shape recognition), it has become a popular and effective solution for NIDS as demonstrated by research work [31–34]. Shallow DL methods have also been proposed for IDS in IoT. For instance, an approach based on a shallow ANN has been proposed by [35] on the UNSW-15 dataset [36], while [37] proposed a similar approach, defined as a Multi-layer Perceptron (MLP), to detect Denial of Service (DoS) attacks in IoT environments on their own test-bed for experiments. Another approach based on a combination of shallow and deep ANN is proposed by [38].

Models based on Recurrent Neural Networks (RNN) are also a popular solution. Long-Short Term Memory or LSTM is part of the RNN-based family of algorithms. This is a class of ANN where neurons maintain a hidden state, obtained from the output of the previous layer. This can be seen as a way to memorize information helping the model generate an output based on a previous state. Because of the ability to store previous states, RNN works well with data that involves sequences, where some data point depends on the previous one. Examples of applications where RNN typically finds its use are speech recognition and text generation. However, its use is also common in IoT intrusion detection. Azumah et al. [39] have proposed an LSTM-based approach for intrusion detection in Smart Home networks achieving very high accuracy. LSTM is also used in [40] to detect attacks in Fog computing. The authors used the ISCX2012 dataset [41] and another dataset based on traffic from an 802.11 network. While the results seem positive, they compared them only against another approach based on logistic regression (LR). A more in-depth comparison with more advanced DL solutions would have provided a better evaluation of their work. The automobile environment is another area of testing for LSTM as in [42], where a manually generated dataset based on traffic from a Controller Area Network (CAN) is used as a test-bed for their approach. They have achieved excellent results on data from a real car and confirmed them using an open-source CAN dataset.

2.2. Ensemble methods in IoT intrusion detection

As the focus of this research work is ensemble stacking, we have reviewed several publications using various types of stacking approaches in IoT intrusion detection. Stacked Generalization, often described as *Stacking*, was first introduced by Wolpert in 1992 [43]. Stacking has the main objective of putting together

the outcomes from the classification of data by *base learners* and then utilizing a *meta-learner* to process this data to achieve a final outcome. One of the key advantages of stacked generalization is the ability to use heterogeneous models as base learners to form an ensemble. The outcome from this can then be combined using a different classifier, trained using outputs from members in the ensemble and leading to a stacking model [44].

Several solutions for intrusion detection in IoT, using stacked generalization with DL and/or ML, have been proposed in recent years. For instance, an approach using a stacked ResNet model is proposed in [45]. The model stacks together five ResNet blocks which train data independently. The output is then concatenated using two fully connected layers of neurons. The approach is evaluated on two open-source datasets in binary classification and compared with other similar approaches. The stacked ResNet model is shown to be more accurate than all others and offers real-time detection.

Another approach based on a stacking ensemble technique is proposed by [46]. The method uses multiple layers of RNN with the objective of detecting anomalies in highly imbalanced data from a smart home network. The work is applied on the BoT-IoT dataset [47], evaluating the model on binary, 5-class and 11-class classification. While they offer an excellent evaluation of that approach, the work would have benefited from a further validation on a different dataset.

A stacking ensemble model to detect attacks in IoT environments was proposed by [48]. The method offers an interesting solution combining several ML models in the ensemble. They performed experiments by changing the algorithms of base learners and of the meta learner according to the type of data they fed into the model. Three different datasets are used with promising results: NDL-KDD [49], UNSW-NB15 and the Credit Card Fraud Detection [50].

Another study that proposes a stacking ensemble approach to IoT anomaly detection is [51]. The work proposes an interesting IoT malware detection model based on an ensemble learner built using a combination of CNN and RNN. They evaluated the model on a dataset built by combining malware samples from various sources with normal samples from images of routers of various brands. While their work offered the advantage of using realistic data, it used a relatively small dataset that may not offer enough heterogeneity to provide an adequate training platform for their model. Evaluating the model on a larger, multidimensional dataset, could have benefited the work.

An interesting stacking approach is offered by [52]. The work uses a combination of boosting and stacking ensemble techniques. The approach is termed B-Stacking as it uses ML algorithms, K-Nearest Neighbour (kNN) and Random Forest (RF) together with XGBoost as base learners. The idea is to use the boosting approach to reduce bias and variance of the dataset fed into the level-1 model. While research work is directed at IoT, they use two datasets based purely on networking data. The results are promising, but they may not provide an indication of the performance of the model in IoT environments.

Kumar et al. [53] designed a stacking ensemble method based on a fog-cloud architecture tailored at the Internet of Medical Things (IoMT). Their method aggregates outcomes from three based learners, Decision Tree (DT), RF and Naive Bayes (NB), with a meta-learner based on XGBoost. The approach was tested on the ToN_IoT network dataset achieving excellent results. However, using an additional dataset to test their method could have offered an additional validation of their results.

Khan et al. [54] have offered a solution based on a stacking ensemble of LSTM models. They used DT as the method for aggregation. Their work is also based on the ToN_IoT network dataset and it is applied to the IoMT using the Fog cloud paradigm. While results are positive, their work is also limited to binary classification and to the use of a single dataset.

Alotaibi and Ilyas [55] proposed an ensemble method also based on stacking. They have used RF, DT, LR and kNN as the base learner and LR as the meta-learner. Their approach offers promising results. However, it is only tested in binary classification and on the ToN_IoT telemetry data. Multi-class classification and the use of an additional dataset could have considerably improved their work.

Two more approaches for IoT IDS, based on stacked generalization are [56,57].

Different types of ensemble approaches have also been proposed in the literature. An example of this is the ensemble learning approach used by [58] to detect botnet attacks in IoT networks. The system, named *ELBA-IoT* reached a detection rate of 99.6% on the N-BaIoT dataset. The approach used a combination of three Decision Tree- based models, namely AdaBoosted DT, RUSboosted DT and bagged DT.

A method based on XGBoost is proposed by Gad et al. [59]. Their approach uses the ToN_IoT dataset in binary and multi-class classification. They tested their method with three of the four versions of the ToN_IoT: network data, Linux and Windows with considerable success.

Several ensemble techniques are presented by [60]. Their work applies XGBoost, Adaboost, Bagging, RF and Extra Trees on intrusion detection in Industrial IoT (IIoT) using the ToN_IoT Telemetry datasets. XGBoost performs better than the rest in both binary and multi-class classification. While their results are positive overall, they indicate that their approach does not cope well with highly imbalanced data. Additional testing on other types of data could have provided more insight into this issue. Additionally, they tested their methods with each separate ToN_IoT telemetry dataset. These, taken on their own are quite small, increasing the likelihood of overfitting especially when proposing complex ensemble models.

Detection of attacks in SCADA using an ensemble method is presented in [61]. The work developed a framework using an ensemble of Deep Belief Networks achieving promising results on a SCADA dataset.

Other ensemble techniques for IoT IDS have been presented in [62–65]. A summary table comparing methods for IoT intrusion detection is given in Table 1

3. Proposed deep integrated stacking model

Stacking is an ensemble technique where a set of models are used as *base learners* to classify given input data. The outcomes of the classification from the base learners, or *level-0* learners, are used as input to the *meta* learner or *level-1*. Stacking allows for the use of heterogeneous learners. In other words, different types of algorithms can be used as base learners to classify data independently. The advantage of this approach is the ability to classify data in drastically different methods, thus improving the capability of analysing highly diverse data. Our stacking ensemble approach uses four different models to build the ensemble. These are a simple Multi-layer Perceptron (MLP), a Deep Neural Network (DNN), a Convolutional Neural Network (CNN) and a Long-Short Term Memory (LSTM). The base learners are then integrated, through the meta-learner, into a larger neural network making the stacking ensemble to be treated as a single large model. Data is fed independently into the models and each is trained until their optimal parameters are found. These parameters are then saved for each model and loaded later to create the full DIS-IoT model. The overall framework of DIS-IoT is illustrated in Fig. 1, while each base learner and the meta learner are described in the sub-sections that follow.

Table 1
Comparison of methods for IoT intrusion detection.

Work	Year	Ensemble method	Dataset	Pros	Cons
[45]	2020	Stacked ResNet	Power system N-BaIoT	High accuracy Real-time detection	Binary detection only
[46]	2021	Stacked RNN	BoT-IoT	Multiple multi-class scenarios High performance in highly imbalanced data	Tested with one dataset
[48]	2022	Stacked ML models	NDL-KDD UNSW-NB15 Credit card fraud detection	Stacking tested with many models Three types of dataset used	Accuracy is not particularly high. Outdated datasets used.
[51]	2020	Stacked CNN & RNN	Proprietary	Dataset created from real samples Cross-architecture malware samples	Tested with one dataset Dataset not validated by others.
[52]	2022	B-Stacking		Lightweight algorithm.	Not using data based on IoT
[53]	2021	Stacked ML models & XGBoost	ToN_IoT	Good method for IoMT	Tested with one dataset only Binary classification only
[54]	2023	Stacked LSTM & DT	ToN_IoT	Good method for IoMT	Tested with one dataset only Binary classification only
[55]	2023	Stacked ML models	ToN_IoT	Tested with both Stacking and Voting	Tested with one dataset only Binary classification only
[58]	2022	ELBA-IoT	N-BaIoT	High performance Multiple models tested Multiple multi-class scenarios	Tested with one dataset only
[59]	2022	XGBoost	ToN_IoT	Distributed method for network & host IDS	Tested with one dataset only
[60]	2023	Bagging; RF XGBoost; ET Adaboost	ToN_IoT	Multiple ensemble models tested	Tested only with telemetry datasets
[61]	2018	Ensemble DBN	SCADA	Data from real devices High performance	Tested with one dataset only

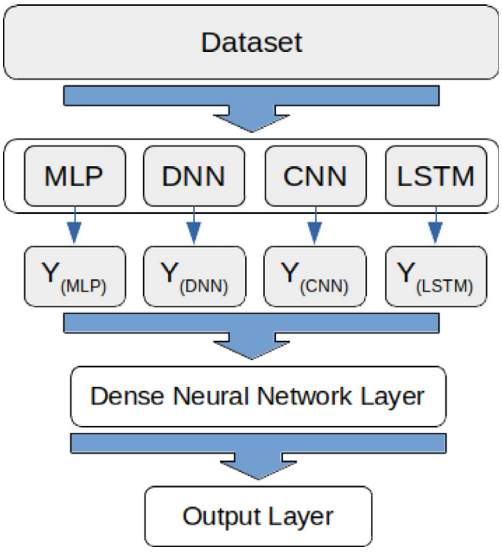

Fig. 1. Deep integrated stacking model.

3.1. Base learners

As mentioned, four models were used to build DIS-IoT. The choice of the base learners was made with the objective of using DL models with different characteristics and capabilities. Each model used provides a different method of classification ensuring heterogeneity during data analysis. However, each model uses the same loss and activation functions, which are selected following an empirical approach. The activation function is the *Rectified Linear Unit* or ReLU, while the loss function changes according to the type of classification. We use *Binary cross-entropy* or *Categorical cross-entropy* depending on whether we conduct binary or multi-class classification. Similarly, the activation function of the output layer changes accordingly. We use *sigmoid* for binary classification or *softmax* for multi-class classification.

3.1.1. Multi layer perceptron

The MLP model is designed to be a simple, but computationally efficient, neural network where layers are fully connected among each other. Tensorflow defines these types of layers as *dense*. The model is made of only three layers (input, hidden and output) making it, by definition, a shallow ANN. Fig. 2, illustrates the MLP Base Learner as one of the four sub-models. The four boxes represent its input data plus the three layers. The input data is shown as a two-dimensional tuple (None, 43). The term *None* represents the samples in the data and it is used as a generic term to allow for input flexibility (e.g. batching). The number 43

represents the number of features used in the ToN_IoT dataset. The MLP represents the simpler of the models used and offers good performance without a notable need for high resources.

3.1.2. Deep Neural Network

The Deep Neural Network model or DNN differs from the MLP in the extra number of layers that it uses. It ensures a more in-depth analysis of the data, requiring more processing power. DNN is composed of four fully connected layers. Moreover, to reduce chances of overfitting, Dropout and Batch Normalization are used in the hidden layers. DNN is also illustrated in Fig. 2. Note that the input data, Batch Normalization and Dropout are all represented as extra layers.

3.1.3. CNN

A Convolutional Neural Network (CNN) is also used as part of the ensemble. CNN was chosen as it offers a well-known method capable of processing large amounts of data, often combined with accurate predictions. On the downside, it does require higher resources and longer processing time during training compared to the other models used. The CNN model built for DIS-IoT makes use of two one-dimensional (1D) convolutional layers. These can be applied to two-dimensional data such as the type of data used in this project. The choice of a 1D Convolutional layer is due to ensure that the computational complexity of the model is reduced, hence that it does not require special hardware setup, making it more suitable for applications such as intrusion detection in all sorts of devices [66]. Our model is made of two 1D convolutional layers, connected to another dense layer before the output layer. Fig. 2 show that CNN is the most complex of all models used. As a requirement for CNN, the input data is reshaped to add an extra dimension. Max Pooling and Dropout are all represented as extra layers.

3.1.4. LSTM

LSTM is used to create the fourth model of the ensemble. The reasoning behind its choice is the fact that LSTM offers an improved ability to analyse sequential data [67] when compared to the other methods. Network traffic, from typical TCP/IP protocols, is normally sequential. The use of LSTM enables DiS-IoT to model this type of data more efficiently. The model contains two LSTM layers, connected to another hidden dense layer. Lastly, the output layer returns the predictions. Dropout is also used as part of the LSTM model as illustrated in Fig. 2

3.2. Meta learner

Rather than using a separate model as the meta-learner, we proposed an integrated neural network layer to combine predictions from the base learners. Its architecture is described next.

3.2.1. Architecture

The meta learner for DIS-IoT is a fully connected neural network layer that combines the predictions of each sub-model and performs an additional training step to reach the final outcome. This makes DIS-IoT a standalone model that integrates four independent sub-models, an additional fully connected layer and a last output layer to produce a final prediction on the given data. As each sub-model uses a different method of classification, the aim is to take advantage of their combination to enable DIS-IoT to model the different characteristics in the data more efficiently. Fig. 2 illustrates the full process. The output layer shows one output class as it represents the outcome from the binary classification of the CICIDS2017 dataset. This changes depending on the type of classification and the dataset. Binary has just one outcome, while the multi-class classification would have multiple output classes.

3.2.2. The DIS-IoT algorithm

With each base-learner trained and their parameters saved, DIS-IoT is built by loading these into the ensemble model with their layers set as *not trainable*. This is because their weights and biases have already been adjusted previously and do not require further updates. A separate copy of the data is inputted to each sub-learner model. Their outputs are then concatenated as input to the meta learner which is trained to provide a final prediction on the data. A detailed step-by-step procedure of the DIS-IoT model is illustrated in Fig. 2. Furthermore, the pseudo-code of the DIS-IoT model is given in Algorithm 1.

Algorithm 1 The DIS-IoT Algorithm

Input Data: $D = \{X_n, Y_n\}$
Base Learners: $M1$ = MLP; $M2$ = DNN; $M3$ = CNN; $M4$ = LSTM
Base Learner Predictions: $\hat{Y}_{Mi}$
Meta Learner: M_L
Final Predictions: $\hat{Y}$

Step 1: Classify D with Base Learners ($M1$ to $M4$)
for `i` $= 1$ `to` 4 **do**
    $\hat{Y}_{Mi} = M_i\{D\}$
    Return $\hat{Y}_{Mi}$
end for
Concatenate Data: $D' = \{(\hat{Y}_{M1}, \hat{Y}_{M2}, \hat{Y}_{M3}, \hat{Y}_{M4}), Y_n\}$

Step 2: Train Meta-Learner M_L
$\hat{Y} = M_L\{D'\}$
Return $\hat{Y}$

The input data D is fed into each of the sub-models *M1 to M4*. Each model classifies the data to obtain predictions (i.e. $\hat{Y}_{M1}$ to $\hat{Y}_{M4}$), which are then concatenated with the actual class labels Y_n to form data D'. This is the input data to the meta-learner M_L, which after training returns the final predictions $\hat{Y}$.

4. Datasets, experiments process and performance metrics

The experiment's setup is a workstation running on an Intel© Core™ i7-5960X CPU with 32 GB of main memory and an NVIDIA GeForce RTX 2060 GPU. Linux Mint 20.3 Cinnamon is the workstation operating system (OS). The model training and testing are completed using a Docker container running Python 3.8.10 with Tensorflow 2.7.0, Scikit-Learn 1.0.2 and Pandas 1.4.1. The process work follows a typical data science process where data is imported, pre-processed and then trained and tested. Each step is described in the next sections.

4.1. Datasets

The experiments are completed using three open-source datasets: ToN_IoT, CICIDS2017 and SWaT. The first contains data obtained from large IoT networks. The second is purely based on a typical network environment. Lastly the third contains traffic from a water treatment plant. The reasoning behind their choice is to ensure that the DIS-IoT model is capable of performing efficiently in different network environments. Each dataset is briefly described in the next sub-sections.

4.1.1. ToN_IoT dataset

The ToN_IoT dataset [68] was collected using a large-scale network created by the University of New South Wales (UNSW) at the Australian Defence Force Academy (ADFA). This network included physical systems, virtual devices, cloud platforms and IoT sensors offering a large number of heterogeneous sources.

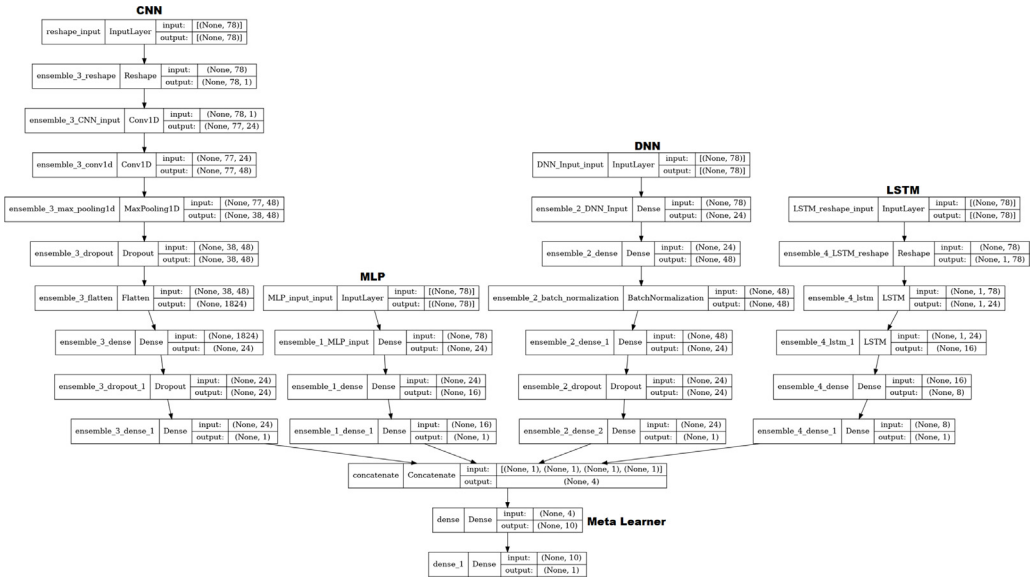

Fig. 2. Network graph of DIS-IoT.

Table 2
ToN_IoT traffic type.

Numerical ID	Traffic type	No of samples
0	Backdoor	20 000
1	DDoS	20 000
2	DoS	20 000
3	Injection	20 000
4	MITM	1043
5	Normal	300 000
6	Password	20 000
7	Ransomware	20 000
8	Scanning	20 000
9	XSS	20 000

Table 3
CICIDS2017 traffic type.

Numerical ID	Traffic type	Number of samples
0	Benign	2 273 097
1	Bot	1966
2	DDoS	128 027
3	DoS GoldenEye	10 293
4	DoS Hulk	230 124
5	DoS slowhttptest	5499
6	DoS slowloris	5796
7	FTP-Patator	7938
8	Heartbleed	11
9	Infiltrator	36
10	PortScan	158 930
11	SSH-Patator	5897
12	Web Attacks - Brute Force	1507
13	Web attack - SQL Inj	21
14	Web attack XSS	652

The data includes several captures from devices with different perspectives of the network: IoT/IIoT, Network, Linux and Windows. For this set of experiments, the network data was used for the model training. Preference was given to the *train_test_network* data as it provides a sample of the network data, as a single file in CSV format, specifically created with the intent of evaluating the efficiency of machine learning applications. The data contains 43 features in total and includes a large sample of normal traffic plus nine different types of attacks. These are listed in Table 2. The *Numerical ID* represents the value used by the algorithm to classify the samples during multi-class classification.

4.1.2. CICIDS2017 dataset

The CICIDS2017 [69] was created by the Canadian Institute for Cybersecurity (CIC) and was specifically designed to help develop solutions for anomaly detection. The dataset contains traffic generated from a network captured over several days and includes a diverse range of attack scenarios. This is a larger dataset compared to the ToN_IoT in numbers of samples, features and classes. The diversity of data is one of the reasoning behind its choice as it offers a more complex environment for network traffic analysis. In total, the dataset contains 79 features with each data sample labelled as either normal or as a specific attack type. A list of all classes is presented in Table 3.

4.1.3. SWaT dataset

The SWat dataset [70] was created by the Singapore University of Technology, and contains traffic obtained from a modern water treatment plant. The data contains 51 features labelled as either normal or attack.

4.2. Data pre-processing

The data, in csv format, is loaded for model training using the Pandas library. Before training, the data was pre-processed using the Scikit-learn library. The process involves several steps. Firstly the data is checked for *null* values. The rows containing these are removed as they represent an insignificant portion of the data in both datasets. Categorical objects are also identified and encoded into numerical form to ensure data could be inputted into the DL models. The next step is to ensure that the data is normalized (i.e. values scaled between 0 and 1). This is an important step to ensure that no outliers exist in the data that could otherwise bias the outcome of the model training. Again, the Scikit-learn library with its *MinMaxScaler* class is used to complete this task. Mathematically, normalization is carried out by Eq. (1).

$$x_i = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{1}$$

To conclude the pre-processing of the data, the dataset is divided into train and test data using a 3:10 ratio, where 30% of the data is kept aside for testing the model with previously unseen data. This is a standard process for ML, as it allows to validate results obtained from training using previously unseen

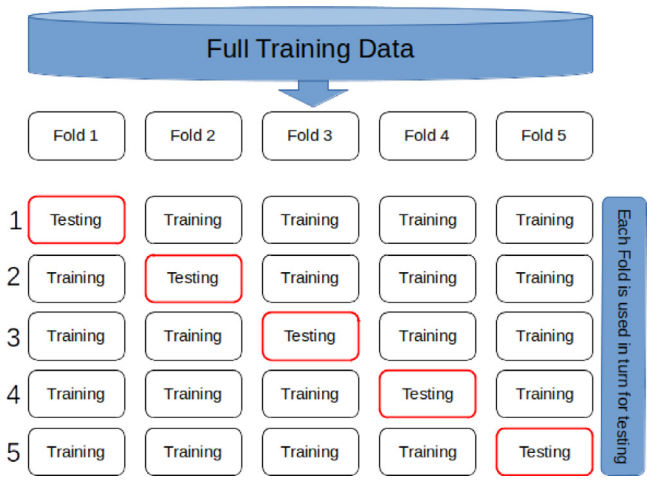

Fig. 3. Illustration of Kfold cross validation.

data. This step ensures that ML models used in operational environments, with live data, can achieve similar results as to their performance during training. Stratified KFold cross-validation is also used in training the other 70% of the data. This process is used to reduce the chance of the model overfitting or being influenced by *selection bias* [71]. In KFold Cross validation the data is divided into n subsets. In turn, each subset, or fold, is kept for testing while the rest of the data is used for training. This implies that training is performed on the data n times - i.e. the number of folds created beforehand. While this works well on balanced data, where each label is distributed evenly, it may not be the best solution for data that contains an unbalanced distribution of their classes. The reason for this is that KFold CV splits the data without taking into account the class ratio. Therefore, a fold could have a large proportion of a type of class compared to others, while another fold could only contain a small amount of the same class. As the data used in this project is unbalanced, standard Kfold CV is not suitable and Stratified KFold CV is used instead. Stratified KFold CV maintains the same class ratio as in the source data for each of the folds created. In short by using Stratified KFold CV, the class ratio is preserved in all subsets of the data. Hence, for each fold, similar results are expected.

4.3. Model training and testing

For this experiment, a 5 fold stratified CV is used. The train set is split into 5 subsets, each maintaining the same ratio of classes in them. Out of the 5 subsets, one is kept in turn for testing while the other four are used for training the models. Effectively, the training process is repeated 5 times - i.e. one for each fold. Fig. 3 illustrates this process.

Once this process is completed, the remaining 30% of the data is used as test data for a further run. This ensures validation of the models with previously unseen data.

4.4. Performance metrics

Evaluation of ML and DL models for classification problems such as the one presented in this work is mostly based on metrics obtained from a confusion matrix (CM). This is a cross table that reports how often a model is capable of correctly classifying a data sample with its real label. The model attempts to discover the correct type of data sample. This prediction is recorded and compared against the real type. The CM is used to calculate the number of occurrences the model correctly or incorrectly classifies data. In the context of anomaly or intrusion detection, a CM can be used to verify the rate at which a model manages to:

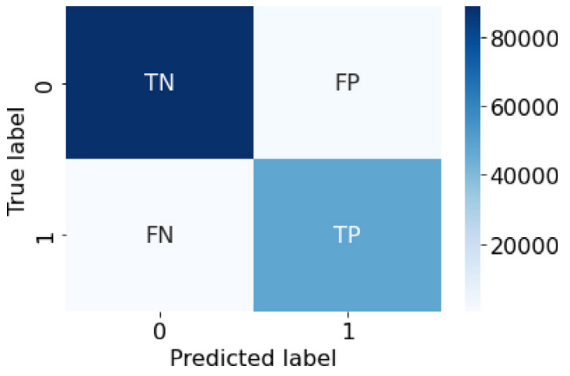

Fig. 4. Confusion matrix.

- detect anomalies or attacks correctly - i.e. True Positives (TP)
- detect normal traffic correctly - i.e. True Negatives (TN)
- confuse normal traffic as anomalous - i.e. False Positives (FP)
- confuse anomalous traffic as normal - i.e. False Negatives (FN)

A CM is often displayed in a tabular format similar to Fig. 4. On the right side, the numbers indicate the matching colour code (e.g. Dark Blue indicates numbers in the order 80K).

An ideal model would identify all TP and TN correctly and never confuse one class of traffic for the other. Of course, this is not realistically achievable. However, the rates of FP and FN should be kept to a minimum. A CM allows for certain important metrics to be calculated. These are:

- **Accuracy** - This is the ratio of correctly classified instances among the total number and it is obtained by Eq. (2).

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{2}$$

- **FPR** - False Positive Rate or FPR, is the ratio between normal instances detected as anomalies and the overall number of normal instances. It can be obtained by Eq. (3)

$$FPR = \frac{FP}{FP + TN} \tag{3}$$

- **Precision** - This provides the rate of elements that have been classified as positive and that are actually positive. It is obtained by dividing correctly classified anomalies (TP), by the total number of positive instances (FP +TP) as shown in Eq. (4).

$$Precision = \frac{TP}{FP + TP} \tag{4}$$

- **Recall** - Also defined as True Positive Rate (TPR), recall is obtained from the correctly classified attacks (TP) divided by the total number of attacks (TP) + (FN) and measures the model's ability to identify all positive instances (i.e. attacks) in the data. Recall is calculated by Eq. (5).

$$Recall = \frac{TP}{TP + FN} \tag{5}$$

- **F1-Score** - This uses both Precision and Recall to calculate their harmonic mean as shown in Eq. (6). The higher the score the better the model.

$$\text{F1-Score} = 2 * \frac{Recall * Precision}{Recall + Precision} \tag{6}$$

In addition to the metrics above, we plot the Receiver Operating Characteristic (ROC) curve for each of the tested models.

Table 4
Performances of binary classification on ToN_IoT dataset.

Model	Accuracy	Precision	FPR	Recall	F1-score	AUC
MLP	0.984	0.972	0.014	0981	0.977	0.999
DNN	0.985	0.977	0.012	0.982	0.979	0.999
CNN	0.995	0.994	0.002	0.991	0.993	1.000
LSTM	0.988	0.989	0.005	0.976	0.983	0.999
DIS-IoT	0.996	0.994	0.002	0.994	0.994	1.000

This provides a visual representation of the ability of the model to distinguish between classes. The area visible below the ROC curve is defined as the *Area Under the Curve* or AUC and it provides a measure of performance of the ability of the model to distinguish between normal and anomalous instances. The higher the AUC score, the more efficient the model. We present FPR, AUC score and the ROC curve only for binary classification to identify the rate of normal traffic being classified as anomalous. All of the other metrics are calculated for both binary and multi-class classification. However, in the latter, as we are dealing with a CM that contains multiple class results, it is necessary, as explained by [72], to use an averaging technique to obtain an overall score for each metric. Specifically, a *weighted* averaging score will be used in a multi-class classification where class imbalance is considered according to the number of samples of each class in the data.

5. Results and discussions

In this section, we present results from the experiments on ToN_IoT, CICIDS2017 and the SWaT datasets using DIS-IoT. To evaluate the performance of DIS-IoT four common DL models are used for comparison. Classification of ToN_IoT and CICIDS2017 is presented for both binary and multi-class classification. In binary classification, all classes of attacks are put together as a single group of anomalies with the same label ID. On the other hand, only binary classification was completed on SWaT as its data is only labelled as either normal or attack. Moreover, the results from the experiments on the ToN_IoT dataset are validated further by comparing them against other recent work using the same data. Results are presented in tabular format and evaluated using the metrics described in the previous section.

5.1. Binary classification

Binary classification aims at identifying anomalies in network traffic without attempting to discover the specific type of attack that the anomaly relates to. Consequently, the problem in itself is simpler than a multi-class classification problem for any DL algorithm. This section presents results obtained from the binary classification on the ToN_IoT, CICIDS2017 and SWaT datasets.

5.1.1. Binary classification of the ToN_IoT dataset

To begin with, Table 4 presents the scores obtained by each model during the binary classification of ToN_IoT.

Results show that all algorithms perform extremely well in detecting anomalies with very high accuracy rate and low FPR rates. However, DIS-IoT offers a clear improvement over the other models. The stacking ensemble manages to perform better than any other algorithms in terms of accuracy, recall and F1-Score, with CNN being the only one matching DIS-IoT results in Precision and FPR. The AUC value is extremely high for all models as shown in Table 4 and Fig. 5

Table 5
Performances of binary classification on CICIDS2017 dataset.

Model	Accuracy	Precision	FPR	Recall	F1-score	AUC
MLP	0.984	0.949	0.012	0.972	0.961	0.999
DNN	0.981	0.927	0.018	0.984	0.955	0.998
CNN	0.985	0.939	0.015	0.987	0.962	0.998
LSTM	0.983	0.956	0.010	0.959	0.957	0.999
DIS-IoT	0.987	0.959	0.010	0.976	0.967	0.999

Table 6
Performances of binary classification on SWaT dataset.

Model	Accuracy	Precision	FPR	Recall	F1-score	AUC
MLP	0.990	0.991	0.008	0.999	0.995	0.983
DNN	0.992	0.992	0.007	0.999	0.996	0.995
CNN	0.993	0.993	0.006	0.999	0.996	0.997
LSTM	0.996	0.996	0.003	0.999	0.998	0.998
DIS-IoT	0.996	0.997	0.002	0.999	0.998	0.999

Table 7
Performances of multi-class classification on ToN_IoT dataset.

Model	Accuracy	Precision	Recall	F1-score
MLP	0.994	0.994	0.994	0.994
DNN	0.992	0.992	0.992	0.992
CNN	0.991	0.991	0.991	0.991
LSTM	0.990	0.990	0.990	0.990
DIS-IoT	0.997	0.997	0.997	0.997

5.1.2. Binary classification of the CICIDS2017 dataset

A similar trend in the results is obtained from the binary classification on the CICIDS2017 data. The models perform not so high as with the previous data. Nonetheless, this is an expected outcome given that the CICIDS2017 dataset contains more samples and has a higher diversity. The similarities with the previous experiments is in the fact that the DIS-IoT improves performance compared to the other models, with higher scores in all metrics. This is shown in Table 5 and Fig. 6.

5.1.3. Binary classification of the SWaT dataset

Classification of the SWaT dataset resulted in an excellent outcome from DIS-IoT. The base models perform well, but again the ensemble model achieves higher results than the rest. Results from these experiments are given in Table 6 and Fig. 7.

5.2. Multi-class classification

Multi-class classification aims at mapping classes of data to their target labels. Specifically, in the case of network intrusion detection, the aim is to identify attacks and classify them correctly. In other words, rather that just identifying anomalies as with binary classification, here the challenge is to identify the type of anomaly that is hidden in the data.

5.2.1. Multi-class classification of the ToN_IoT dataset

The ToN_IoT Dataset contains ten different classes of data. Nine of them are attacks, while the remaining class indicates normal traffic. These are described in Table 2, which shows the type of attacks that exist in the data and their corresponding number that is used in the multi-class classification. A CM of the classification results obtained by DIS-IoT is given in Fig. 8. Table 7 shows the metrics used in multi-class classification.

Results show that DIS-IoT performs extremely well in multi-class classification too, achieving extremely high scores in all metrics. Furthermore, it can be seen that the other models are outperformed in this type of classification too. The confusion matrix in Fig. 8, shows that DIS-IoT tends to have small rates of FP and FN, making the label prediction extremely accurate.

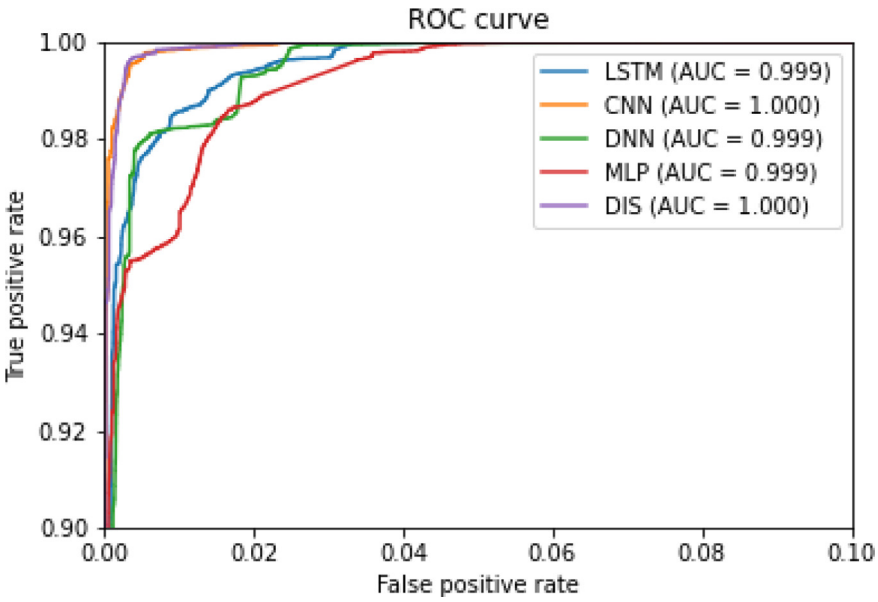

Fig. 5. ROC curves of DIS-IoT and the other methods on ToN_IoT dataset.

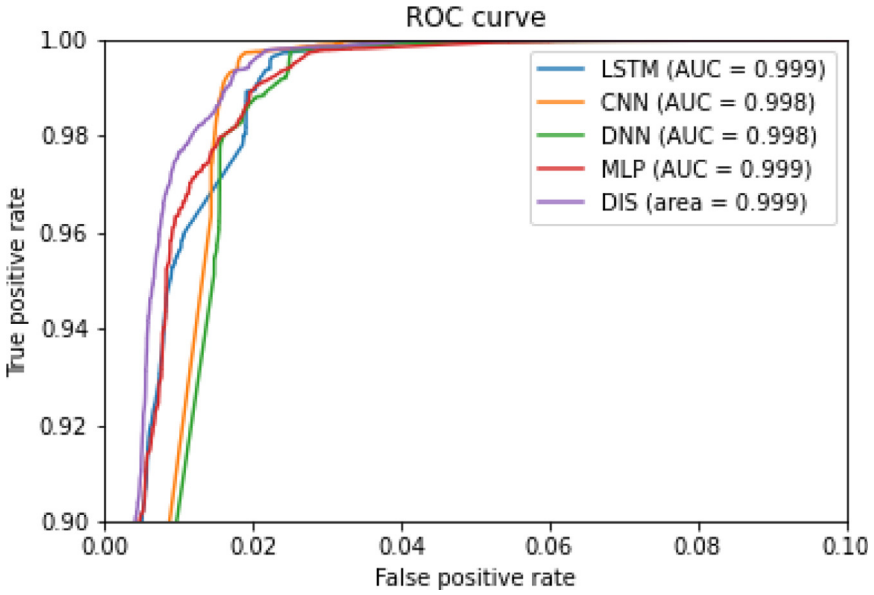

Fig. 6. ROC curves of DIS-IoT and the other methods on CICIDS2017 dataset.

Table 8
Performance of multi-class classification on CICIDS2017 dataset.

Model	Accuracy	Precision	Recall	F1-score
MLP	0.982	0.984	0.982	0.982
DNN	0.980	0.981	0.980	0.980
CNN	0.980	0.980	0.980	0.980
LSTM	0.985	0.984	0.985	0.984
DIS-IoT	0.987	0.987	0.987	0.986

5.2.2. Multi-class classification of the CICIDS2017 dataset

As shown in Table 3, CICIDS2017 contains 15 classes of data, making a more complex scenario than ToN_IoT dataset. A confusion matrix for the outcome of the experiments is shown in Fig. 9. Metrics from the multi-class classification of the CICIDS2017 dataset are given in Table 8.

Given the higher complexity of the experiments, DIS-IoT achieves lower scores than previously. However, it still offers an excellent performance and outperforms all the other models.

5.3. Comparison with other works

As a final evaluation of the results offered above, we compare DIS-IoT with other similar State-of-the-Art approaches [27,28,53,54,59], all of which are described in Section 2. One of the main reasoning for the use of this approach in comparison to ours is that all of these methods use the ToN_IoT network dataset, which is the same data used in this work. Some of them also propose ensemble stacking solutions, offering another important characteristic for comparison. Tables 9 and 10 compare results from binary and multi-class classification respectively. Several of these approaches have been evaluated only on binary classification. Hence, the reason for the missing values in Table 10. From the metrics given, it can be seen that, while DIS-IoT outperforms

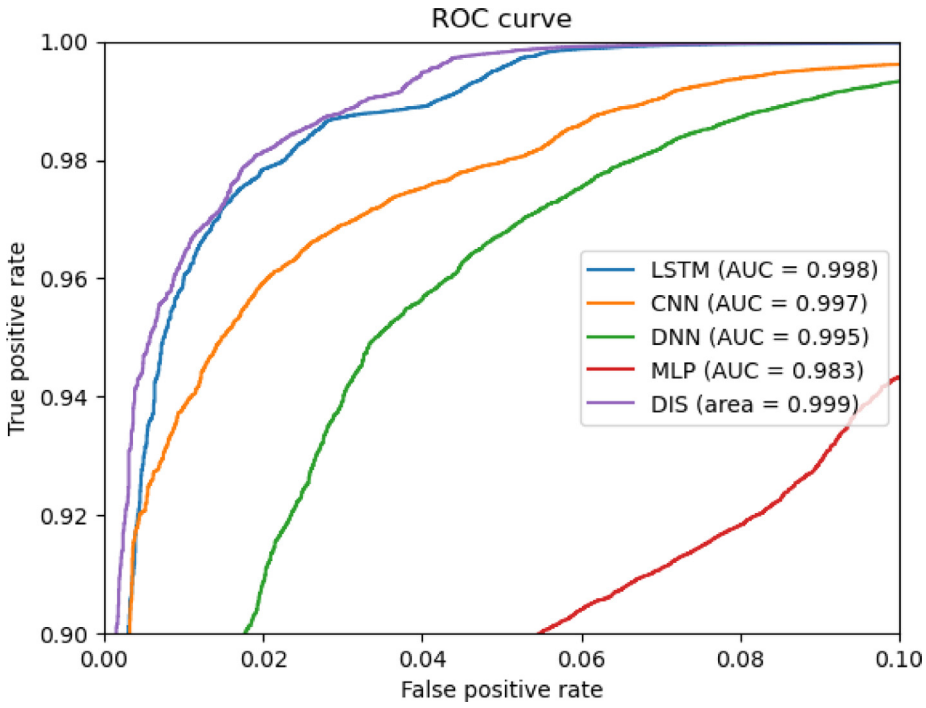

Fig. 7. ROC curves of DIS-IoT and other methods on the SWaT dataset.

Table 9
Comparison with recent State-of-the-Art IDS binary classification on ToN_IoT dataset.

Authors	Method	Ensemble (yes/no)	Accuracy	Precision	Recall	F1-score
Kumar et al. [28]	DLTIF	No	0.999	0.999	0.999	0.999
Latif et al. [27]	DnRaNN	No	0.985	0.990	0.991	0.990
Kumar et al. [53]	Stacking	Yes	0.963	0.905	0.999	0.950
Khan et al. [54]	Stacking	Yes	0.985	0.947	0.998	0.973
Gad et al. [59]	XGBoost	Yes	0.991	0.984	0.991	0.987
Our model	DIS-IoT	Yes	0.996	0.994	0.994	0.994

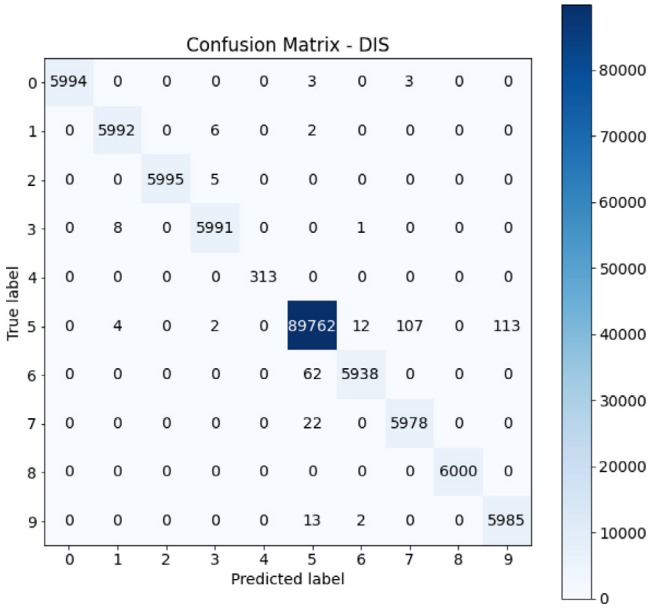

Fig. 8. Multi-class confusion matrix for DIS-IoT on the ToN_IoT dataset.

most of the other approaches in both binary and multi-class classification. Only DLTIF [28] offers slightly improved metrics in binary classification.

6. Conclusions

Detection of anomalies and malicious attacks in IoT has gained paramount importance in recent years. With the number of attacks increasing, it is important to create tools capable of detecting anomalies swiftly and accurately. In this article, a novel method, called DIS-IoT, was introduced. DIS-IoT is capable of detecting attacks in an IoT environment while maintaining a low FP and FN rate. DIS-IoT combines four different DL models, namely MLP, DNN, CNN and LSTM, to take advantage of the different classification characteristics each sub-model is able to offer. The work is evaluated on two open source datasets, ToN_IoT and CICIDS2017 in binary and multi-class classification and on the SWaT dataset in binary classification only. Results are compared against models used for the ensemble. Moreover, results are compared with previous work published that used the ToN_IoT network dataset.

Experiments showed that DIS-IoT is capable of achieving excellent scores in accuracy, precision, recall and F1-Score in both binary and multi-class classification and on both datasets. DIS-IoT outperforms all the other models in all metrics. Comparison with other work also shows that our approach outperforms most of the other works using the ToN_IoT dataset in both binary and multi-class classification. Only in binary classification, DIS-IoT achieved lower scores by a small margin compared to DLTIF [28]. However, this seems to be marginal and could be due to a variety of reasons, which cannot be immediately evident from the outcome of metrics used. Instead, it is crucial to notice that our proposed method,

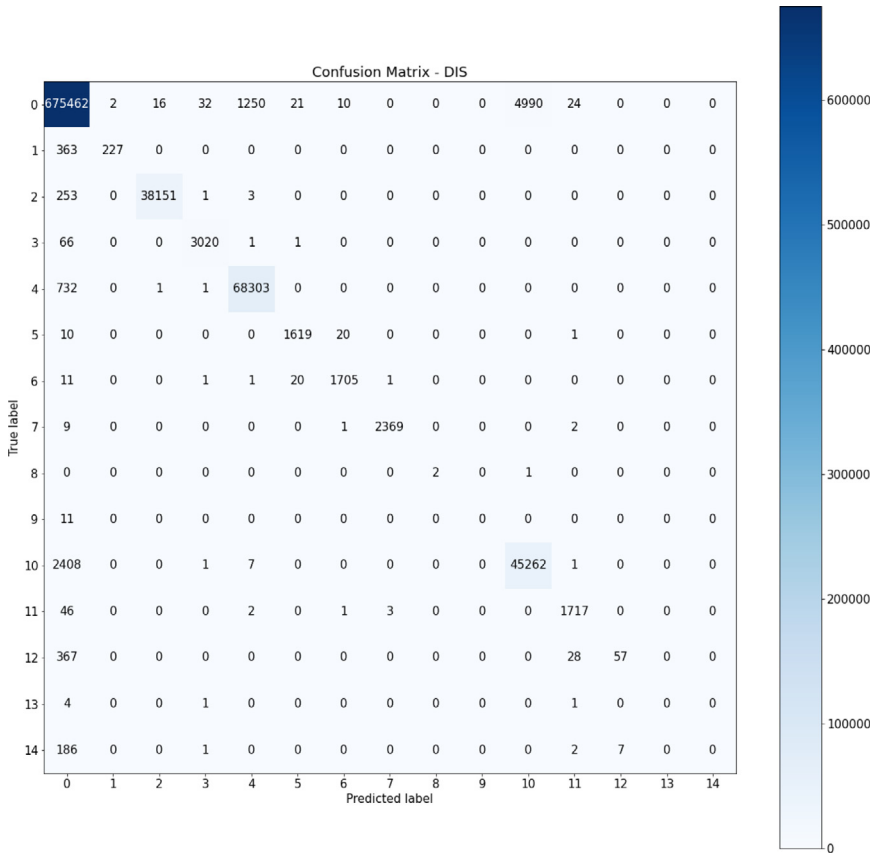

Fig. 9. Multi-class confusion matrix for DIS-IoT on the CICIDS2017 dataset.

Table 10
Comparison with recent State-of-the-Art IDS using ToN_IoT - Multi-class scenario approaches.

Authors	Method name	Ensemble (yes/no)	Accuracy	Precision	Recall	F1-score
Kumar et al. [28]	DLTIF	No	0.993	0.994	0.972	0.985
Latif et al. [27]	DnRaNN	No	0.989	0.993	0.994	0.993
Kumar et al. [53]	Stacking	Yes	–	–	–	–
Khan et al. [54]	Stacking	Yes	–	–	–	–
Gad et al. [59]	XGBoost	Yes	0.983	0.945	0.953	0.949
Our model	DIS-IoT	Yes	0.997	0.997	0.997	0.997

performs consistently well with different datasets, making it a sound solution for intrusion detection in IoT.

In conclusion, our work of combining different models into an integrated stacking ensemble offers an excellent solution as an IDS, demonstrating the ability to detect the vast majority of attacks while maintaining particularly low rates of FP and FN.

In future work, it is recommended to investigate the use of DIS-IoT with real IoT devices. This would allow for the evaluation of the model's computational overhead and real-time performance with live data. Given that DIS-IoT is an ensemble of four different models, it is expected to require significant resources. Therefore it would be more appropriate to experiment with DIS-IoT in a topology based on edge networking rather than running it directly on low-power IoT devices.

CRediT authorship contribution statement

Riccardo Lazzarini: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Project administration, Resources, Software, Writing – original draft, Writing – review & editing. **Huaglory Tianfield:** Project administration, Supervision, Validation, Writing – review & editing. **Vassilis Charissis:** Supervision, Writing – review & editing.

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.

Data availability

Data will be made available on request.

References

[1] A. Thakkar, R. Lohiya, A review on machine learning and deep learning perspectives of IDS for IoT: Recent updates, security issues, and challenges, Arch. Comput. Methods Eng. 28 (4) (2021) 3211–3243, http://dx.doi.org/10.1007/s11831-020-09496-0.

[2] M. Frank, D. Drikakis, V. Charissis, Machine-learning methods for computational science and engineering, MDPI Comput. 8 (15) (2020) 1–35, http://dx.doi.org/10.3390/computation8010015.

[3] A. Khraisat, A. Alazab, A critical review of intrusion detection systems in the internet of things: techniques, deployment strategy, validation strategy, attacks, public datasets and challenges, Cybersecurity 4 (1) (2021) http://dx.doi.org/10.1186/s42400-021-00077-7.

[4] M. Aljabri, S.S. Aljameel, R.M.A. Mohammad, S.H. Almotiri, S. Mirza, F.M. Anis, M. Aboulnour, D.M. Alomari, D.H. Alhamed, H.S. Altamimi, Intelligent techniques for detecting network attacks: Review and research directions, (ISSN: 14248220) 2021, http://dx.doi.org/10.3390/s21217070.

[5] D.S. Berman, A.L. Buczak, J.S. Chavis, C.L. Corbett, A survey of deep learning methods for cyber security, Information (Switzerland) 10 (4) (2019) http://dx.doi.org/10.3390/info10040122, URL www.mdpi.com/journal/informationhttps://www.mdpi.com/2078-2489/10/4/122.

[6] T. Saranya, S. Sridevi, C. Deisy, T.D. Chung, M.K. Khan, Performance analysis of machine learning algorithms in intrusion detection system: A review, in: Procedia Computer Science, Vol. 171, Elsevier, 2020, pp. 1251–1260, http://dx.doi.org/10.1016/j.procs.2020.04.133.

[7] S. Sengupta, S. Basak, P. Saikia, S. Paul, V. Tsalavoutis, F. Atiah, V. Ravi, A. Peters, A review of deep learning with special emphasis on architectures, applications and recent trends, Knowl.-Based Syst. 194 (2020) 105596, http://dx.doi.org/10.1016/j.knosys.2020.105596, arXiv:1905.13294.

[8] Z. Ahmad, A. Shahid Khan, C. Wai Shiang, J. Abdullah, F. Ahmad, Network intrusion detection system: A systematic study of machine learning and deep learning approaches, Trans. Emerg. Telecommun. Technol. 32 (1) (2021) http://dx.doi.org/10.1002/ett.4150.

[9] Z.H. Zhou, Ensemble Methods: Foundations and Algorithms, CRC Press, 2012, pp. 1–218, http://dx.doi.org/10.1201/b12207.

[10] Y. Ren, L. Zhang, P.N. Suganthan, Ensemble classification and regression-recent developments, applications and future directions [review article], (ISSN: 1556603X) 2016, http://dx.doi.org/10.1109/MCI.2015.2471235.

[11] L. Breiman, Bagging predictors, Mach. Learn. 24 (2) (1996) 123–140, http://dx.doi.org/10.1007/bf00058655.

[12] Z.-H. Zhou, Ensemble learning, in: Machine Learning, Springer Singapore, 2021, pp. 181–210, http://dx.doi.org/10.1007/978-981-15-1967-3_8.

[13] Y. Zhen, L. Xiaodong, D.W. Tong Li, W. Jinjiang, Z. Yunwei, H. Han, A systematic literature review of methods and datasets for anomaly-based network intrusion detection, Elsevier Comput. Secur. 116 (102675) (2022) 1–20, http://dx.doi.org/10.1016/j.cose.2022.102675.

[14] T. Bayu Adhi, L. Sunghoon, Ensemble learning for intrusion detection systems: A systematic mapping study and cross-benchmark evaluation, Elsevier Comput. Sci. Rev. 39 (15) (2021) 1–35, http://dx.doi.org/10.1016/j.cosrev.2020.100357.

[15] T.Y. Win, H. Tianfield, Q. Mair, Big data based security analytics for protecting virtualized infrastructures in cloud computing, IEEE Trans. Big Data 4 (1) (2018) 11–25, http://dx.doi.org/10.1109/TBDATA.2017.2715335.

[16] H. Tianfield, Cyber security situational awareness, in: 2016 IEEE International Conference on Internet of Things (IThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), pp. 782–787, http://dx.doi.org/10.1109/iThings-GreenCom-CPSCom-SmartData.2016.165.

[17] B.S. Khater, A.W.B.A. Wahab, M.Y.I.B. Idris, M.A. Hussain, A.A. Ibrahim, A lightweight perceptron-based intrusion detection system for fog computing, Appl. Sci. (Switzerland) 9 (1) (2019) http://dx.doi.org/10.3390/app9010178, URL www.mdpi.com/journal/applsci.

[18] J. Gao, Network intrusion detection method combining CNN and BiLSTM in cloud computing environment, Comput. Intell. Neurosci. (2022) http://dx.doi.org/10.1155/2022/7272479.

[19] R. Atefinia, M. Ahmadi, Network intrusion detection using multi-architectural modular deep neural network, J. Supercomput. 77 (4) (2021) 3571–3593, http://dx.doi.org/10.1007/s11227-020-03410-y.

[20] A. Krishna, M.A. Ashik Lal, A.J. Mathewkutty, D.S. Jacob, M. Hari, Intrusion detection and prevention system using deep learning, in: Proceedings of the International Conference on Electronics and Sustainable Communication Systems, ICESC 2020, 2020, pp. 273–278, http://dx.doi.org/10.1109/ICESC48915.2020.9155711.

[21] S.A. Althubiti, E.M. Jones, K. Roy, LSTM for anomaly-based network intrusion detection, in: 2018 28th International Telecommunication Networks and Applications Conference, ITNAC 2018, 2019, http://dx.doi.org/10.1109/ATNAC.2018.8615300.

[22] J. Kim, J. Kim, H. Kim, M. Shim, E. Choi, CNN-based network intrusion detection against denial-of-service attacks, Electronics (Switzerland) 9 (6) (2020) 1–21, http://dx.doi.org/10.3390/electronics9060916, URL www.mdpi.com/journal/electronics.

[23] S. Potluri, C. Diedrich, Accelerated deep neural networks for enhanced Intrusion Detection System, in: IEEE International Conference on Emerging Technologies and Factory Automation, ETFA, Vol. 2016-Novem, 2016, http://dx.doi.org/10.1109/ETFA.2016.7733515.

[24] Y. Imrana, Y. Xiang, L. Ali, Z. Abdul-Rauf, A bidirectional LSTM deep learning approach for intrusion detection, Expert Syst. Appl. 185 (2021) 115524, http://dx.doi.org/10.1016/j.eswa.2021.115524.

[25] J. Kim, N. Shin, S.Y. Jo, S.H. Kim, Method of intrusion detection using deep neural network, in: 2017 IEEE International Conference on Big Data and Smart Computing, BigComp 2017, 2017, pp. 313–316, http://dx.doi.org/10.1109/BIGCOMP.2017.7881684.

[26] P. Wu, H. Guo, LuNet: A deep neural network for network intrusion detection, in: 2019 IEEE Symposium Series on Computational Intelligence, SSCI 2019, 2019, pp. 617–624, http://dx.doi.org/10.1109/SSCI44817.2019.9003126, arXiv:1909.10031.

[27] S. Latif, Z.e. Huma, S.S. Jamal, F. Ahmed, J. Ahmad, A. Zahid, K. Dashtipour, M. Umar Aftab, M. Ahmad, Q.H. Abbasi, Intrusion detection framework for the internet of things using a dense random neural network, IEEE Trans. Ind. Inform. (2021) http://dx.doi.org/10.1109/TII.2021.3130248.

[28] P. Kumar, G.P. Gupta, R. Tripathi, S. Garg, M.M. Hassan, DLTIF: Deep learning-driven cyber threat intelligence modeling and identification framework in IoT-enabled maritime transportation systems, IEEE Trans. Intell. Transp. Syst. (2021) 1–10, http://dx.doi.org/10.1109/tits.2021.3122368.

[29] X. Kan, Y. Fan, Z. Fang, L. Cao, N.N. Xiong, D. Yang, X. Li, A novel IoT network intrusion detection approach based on adaptive particle swarm optimization convolutional neural network, Inform. Sci. 568 (2021) 147–162, http://dx.doi.org/10.1016/j.ins.2021.03.060.

[30] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, Y. Elovici, N-BaIoT-network-based detection of IoT botnet attacks using deep autoencoders, IEEE Pervasive Comput. 17 (3) (2018) 12–22, http://dx.doi.org/10.1109/MPRV.2018.03367731, arXiv:1805.03409 URL http://archive.ics.uci.edu/ml/datasets/detection.

[31] A. Derhab, A. Aldweesh, A.Z. Emam, F.A. Khan, Intrusion detection system for internet of things based on temporal convolution neural network and efficient feature engineering, Wirel. Commun. Mob. Comput. 2020 (2020) http://dx.doi.org/10.1155/2020/6689134.

[32] Y. Li, Y. Xu, Z. Liu, H. Hou, Y. Zheng, Y. Xin, Y. Zhao, L. Cui, Robust detection for network intrusion of industrial IoT based on multi-CNN fusion, Meas.: J. Int. Meas. Confed. 154 (2020) http://dx.doi.org/10.1016/j.measurement.2019.107450.

[33] A. Li, S. Yi, Intelligent intrusion detection method of industrial internet of things based on CNN-BiLSTM, Secur. Commun. Netw. 2022 (2022) 1–8, http://dx.doi.org/10.1155/2022/5448647.

[34] A. Alferaidi, K. Yadav, Y. Alharbi, N. Razmjooy, W. Viriyasitavat, K. Gulati, S. Kautish, G. Dhiman, Distributed deep CNN-LSTM model for intrusion detection method in IoT-based vehicles, Math. Probl. Eng. 2022 (2022) 1–8, http://dx.doi.org/10.1155/2022/3424819.

[35] S. Hanif, T. Ilyas, M. Zeeshan, Intrusion detection in IoT using artificial neural networks on UNSW-15 dataset, in: 2019 IEEE 16th International Conference on Smart Cities: Improving Quality of Life using ICT & IoT and AI (HONET-ICT), IEEE, 2019, pp. 152–156, http://dx.doi.org/10.1109/HONET.2019.8908122, URL https://ieeexplore.ieee.org/document/8908122/.

[36] N. Moustafa, J. Slay, UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set), in: 2015 Military Communications and Information Systems Conference, MilCIS 2015 - Proceedings, 2015, http://dx.doi.org/10.1109/MilCIS.2015.7348942, URL https://cve.mitre.org/.

[37] E. Hodo, X. Bellekens, A. Hamilton, P.L. Dubouilh, E. Iorkyase, C. Tachtatzis, R. Atkinson, Threat analysis of IoT networks using artificial neural network intrusion detection system, in: 2016 International Symposium on Networks, Computers and Communications, ISNCC 2016, 2016, http://dx.doi.org/10.1109/ISNCC.2016.7746067, arXiv:1704.02286.

[38] M. Al-Zewairi, S. Almajali, M. Ayyash, Unknown security attack detection using shallow and deep ann classifiers, Electronics (Switzerland) 9 (12) (2020) 1–27, http://dx.doi.org/10.3390/electronics9122006, URL www.mdpi.com/journal/electronics.

[39] S.W. Azumah, N. Elsayed, V. Adewopo, Z.S. Zaghloul, C. Li, A deep LSTM based approach for intrusion detection IoT devices network in smart home, in: 7th IEEE World Forum on Internet of Things, WF-IoT 2021, 2021, pp. 836–841, http://dx.doi.org/10.1109/WF-IoT51360.2021.9596033.

[40] A. Diro, N. Chilamkurti, Leveraging LSTM networks for attack detection in fog-to-things communications, IEEE Commun. Mag. 56 (9) (2018) 124–130, http://dx.doi.org/10.1109/MCOM.2018.1701270.

[41] A. Shiravi, H. Shiravi, M. Tavallaee, A.A. Ghorbani, Toward developing a systematic approach to generate benchmark datasets for intrusion detection, Comput. Secur. 31 (3) (2012) 357–374, http://dx.doi.org/10.1016/j.cose.2011.12.012.

[42] M.D. Hossain, H. Inoue, H. Ochiai, D. Fall, Y. Kadobayashi, LSTM-based intrusion detection system for in-vehicle can bus communications, IEEE Access 8 (2020) 185489–185502, http://dx.doi.org/10.1109/ACCESS.2020.3029307.

[43] D.H. Wolpert, Stacked generalization, Neural Netw. 5 (2) (1992) 241–259, http://dx.doi.org/10.1016/S0893-6080(05)80023-1.

[44] R. Polikar, Ensemble learning, in: Ensemble Machine Learning, Springer, Boston, MA, 2012, pp. 1–34, http://dx.doi.org/10.1007/978-1-4419-9326-7_1, URL https://link.springer.com/chapter/10.1007/978-1-4419-9326-7_1.

[45] B. Alotaibi, M. Alotaibi, A stacked deep learning approach for IoT cyberattack detection, J. Sens. 2020 (2020) http://dx.doi.org/10.1155/2020/8828591.

[46] S.I. Popoola, B. Adebisi, M. Hammoudeh, H. Gacanin, G. Gui, Stacked recurrent neural network for botnet detection in smart homes, Comput. Electr. Eng. 92 (2021) 107039, http://dx.doi.org/10.1016/j.compeleceng.2021.107039.

[47] N. Koroniotis, N. Moustafa, E. Sitnikova, B. Turnbull, Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset, Future Gener. Comput. Syst. 100 (2019) 779–796, http://dx.doi.org/10.1016/j.future.2019.05.041, arXiv:1811.00701.

[48] R. Soleymanzadeh, M. Aljasim, M.W. Qadeer, R. Kashef, Cyberattack and fraud detection using ensemble stacking, AI 3 (1) (2022) 22–36, http://dx.doi.org/10.3390/ai3010002, URL https://www.mdpi.com/2673-2688/3/1/2/htmhttps://www.mdpi.com/2673-2688/3/1/2.

[49] M. Tavallaee, E. Bagheri, W. Lu, A.A. Ghorbani, A detailed analysis of the KDD CUP 99 data set, in: IEEE Symposium on Computational Intelligence for Security and Defense Applications, CISDA 2009, 2009, http://dx.doi.org/10.1109/CISDA.2009.5356528.

[50] A. Dal Pozzolo, G. Boracchi, O. Caelen, C. Alippi, G. Bontempi, Credit card fraud detection: A realistic modeling and a novel learning strategy, IEEE Trans. Neural Netw. Learn. Syst. 29 (8) (2018) 3784–3797, http://dx.doi.org/10.1109/TNNLS.2017.2736643, URL http://www.ieee.org/publications_standards/publications/rights/index.html.

[51] D. Vasan, M. Alazab, S. Venkatraman, J. Akram, Z. Qin, MTHAEL: Cross-architecture iot malware detection based on neural network advanced ensemble learning, IEEE Trans. Comput. 69 (11) (2020) 1654–1667, http://dx.doi.org/10.1109/TC.2020.3015584.

[52] S. Roy, J. Li, B.J. Choi, Y. Bai, A lightweight supervised intrusion detection mechanism for IoT networks, Future Gener. Comput. Syst. 127 (2022) 276–285, http://dx.doi.org/10.1016/j.future.2021.09.027.

[53] P. Kumar, G.P. Gupta, R. Tripathi, An ensemble learning and fog-cloud architecture-driven cyber-attack detection framework for IoMT networks, Comput. Commun. 166 (2021) 110–124, http://dx.doi.org/10.1016/j.comcom.2020.12.003.

[54] F. Khan, M.A. Jan, R. Alturki, M.D. Alshehri, S.T. Shah, A. ur Rehman, A secure ensemble learning-based fog-cloud approach for cyberattack detection in IoMT, IEEE Trans. Ind. Inform. (2023) 1–9, http://dx.doi.org/10.1109/tii.2022.3231424.

[55] Y. Alotaibi, M. Ilyas, Ensemble-learning framework for intrusion detection to enhance internet of things' devices security, Sensors 23 (12) (2023) 5568, http://dx.doi.org/10.3390/S23125568, URL https://www.mdpi.com/1424-8220/23/12/5568/htmhttps://www.mdpi.com/1424-8220/23/12/5568, 2023, Vol. 23, Page 5568.

[56] M. Saharkhizan, A. Azmoodeh, A. Dehghantanha, K.K.R. Choo, R.M. Parizi, An ensemble of deep recurrent neural networks for detecting IoT cyber attacks using network traffic, IEEE Internet Things J. 7 (9) (2020) 8852–8859, http://dx.doi.org/10.1109/JIOT.2020.2996425, URL https://www.ieee.org/publications/rights/index.html.

[57] P.L. Indrasiri, E. Lee, V. Rupapara, F. Rustam, I. Ashraf, Malicious traffic detection in iot and local networks using stacked ensemble classifier, Comput. Mater. Contin. 71 (1) (2022) 489–515, http://dx.doi.org/10.32604/cmc.2022.019636.

[58] Q.A. Al-Haija, M. Al-Dala'ien, ELBA-IoT: An ensemble learning model for botnet attack detection in IoT networks, J. Sens. Actuat. Netw. 11 (1) (2022) 18, http://dx.doi.org/10.3390/jsan11010018.

[59] A.R. Gad, M. Haggag, A.A. Nashat, T.M. Barakat, A distributed intrusion detection system using machine learning for IoT based on ToN-IoT dataset, Int. J. Adv. Comput. Sci. Appl. 13 (6) (2022) 548–563, http://dx.doi.org/10.14569/IJACSA.2022.0130667, URL www.ijacsa.thesai.org.

[60] J.B. Awotunde, S.O. Folorunso, A.L. Imoize, J.O. Odunuga, C.C. Lee, C.T. Li, D.T. Do, An ensemble tree-based model for intrusion detection in industrial internet of things networks, Appl. Sci. (Switzerland) 13 (4) (2023) 2479, http://dx.doi.org/10.3390/app13042479, URL https://www.mdpi.com/2076-3417/13/4/2479/htmhttps://www.mdpi.com/2076-3417/13/4/2479.

[61] S. Huda, J. Yearwood, M.M. Hassan, A. Almogren, Securing the operations in SCADA-IoT platform based industrial control system using ensemble of deep belief networks, Appl. Soft Comput. 71 (2018) 66–77, http://dx.doi.org/10.1016/j.asoc.2018.06.017.

[62] E. Tsogbaatar, M.H. Bhuyan, Y. Taenaka, D. Fall, K. Gonchigsumlaa, E. Elmroth, Y. Kadobayashi, DeL-IoT: A deep ensemble learning approach to uncover anomalies in IoT, Internet Things (Netherlands) 14 (2021) 100391, http://dx.doi.org/10.1016/j.iot.2021.100391.

[63] D. Yang, M. Hwang, Unsupervised and ensemble-based anomaly detection method for network security, 2022, pp. 75–79, http://dx.doi.org/10.1109/kst53302.2022.9729061, URL https://github.com/yangdonghun3/oae.

[64] K. Albulayhi, F.T. Sheldon, An adaptive deep-ensemble anomaly-based intrusion detection system for the internet of things, in: 2021 IEEE World AI IoT Congress, AIIoT 2021, 2021, pp. 187–196, http://dx.doi.org/10.1109/AIIoT52608.2021.9454168.

[65] S. Tang, Z. Gu, Q. Yang, S. Fu, Smart home IoT anomaly detection based on ensemble model learning from heterogeneous data, in: Proceedings - 2019 IEEE International Conference on Big Data, Big Data 2019, 2019 pp. 4185–4190, http://dx.doi.org/10.1109/BigData47090.2019.9006249.

[66] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, D.J. Inman, 1D convolutional neural networks and applications: A survey, Mech. Syst. Signal Process. 151 (2021) http://dx.doi.org/10.1016/j.ymssp.2020.107398, arXiv:1905.03554.

[67] H. Gwon, C. Lee, R. Keum, H. Choi, Improvement in network intrusion detection based on LSTM and feature embedding, J. KIISE 48 (4) (2021) 418–424, http://dx.doi.org/10.5626/jok.2021.48.4.418, arXiv:1911.11552v1.

[68] T.M. Booij, I. Chiscop, E. Meeuwissen, N. Moustafa, F.T. Hartog, ToN_IoT: The role of heterogeneity and the need for standardization of features and attack types in IoT network intrusion data sets, IEEE Internet Things J. 9 (1) (2022) 485–496, http://dx.doi.org/10.1109/JIOT.2021.3085194, URL https://www.ieee.org/publications/rights/index.html.

[69] I. Sharafaldin, A.H. Lashkari, A.A. Ghorbani, Toward generating a new intrusion detection dataset and intrusion traffic characterization, in: ICISSP 2018 - Proceedings of the 4th International Conference on Information Systems Security and Privacy, Vol. 2018-Janua, SciTePress, 2018, pp. 108–116, http://dx.doi.org/10.5220/0006639801080116.

[70] J. Goh, S. Adepu, K. Junejo, A. Mathur, A dataset to support research in the design of secure water treatment systems, 2016.

[71] G.C. Cawley, N.L. Talbot, On over-fitting in model selection and subsequent selection bias in performance evaluation, J. Mach. Learn. Res. 11 (2010) 2079–2107.

[72] M. Grandini, E. Bagli, G. Visani, Metrics for multi-class classification: an overview, 2020, arXiv:2008.05756 URL http://arxiv.org/abs/2008.05756.