Cybersecurity

RESEARCH Open Access

An ensemble deep learning based IDS for IoT using Lambda architecture

Rubayyi Alghamdi and Martine Bellaiche*

Abstract

The Internet of Things (IoT) has revolutionized our world today by providing greater levels of accessibility, connectivity and ease to our everyday lives. It enables massive amounts of data to be traversed across multiple heterogeneous devices that are all interconnected. This phenomenon makes IoT networks vulnerable to various network attacks and intrusions. Building an Intrusion Detection System (IDS) for IoT networks is challenging as they enable a massive amount of data to be aggregated, which is difficult to handle and analyze in real time mainly because of the heterogeneous nature of IoT devices. This inefficient, traditional IDS approach accentuates the need to develop advanced IDS techniques by employing Machine or Deep Learning. This paper presents a deep ensemble-based IDS using Lambda architecture by following a multi-pronged classification approach. Binary classification uses Long Short Term Memory (LSTM) to differentiate between malicious and benign traffic, while the multi-class classifier uses an ensemble of LSTM, Convolutional Neural Network and Artificial Neural Network classifiers to detect the type of attacks. The model training is performed in the batch layer, while real-time evaluation is carried out through model inferences in the speed layer of the Lambda architecture. The proposed approach gives high accuracy of over 99.93% and saves useful processing time due to the multi-pronged classification strategy and using the lambda architecture.

Keywords IoT, IDS, Lambda architecture, Cyber-attacks, Deep learning, Ensemble learning

Introduction

The Internet has been overgrown in the last decade resulting in easy access to information, data and resources online (Lata and Kumar 2022). This high connectivity has resulted in the growth and everyday use of the Internet of Things (IoT) (Nair 2019). These days the applications of IoT range from many areas, including smart homes, medicine, industries, smart enterprises, infrastructures, monitoring systems and defense sectors resulting in totally transforming the world we live in today (Khattak et al. 2019). This widespread use has accentuated the need for further developing the IoT environment to tackle the growing challenges of the future. These challenges include performance and computational issues, protocol-related vulnerabilities, the heterogeneous nature of IoT devices and above all, security and privacy concerns.

The security and privacy aspects of IoT have lately been a widely discussed topic among researchers because of the massive surge in IoT-related attacks. Tackling these attacks in an IoT environment are relatively different from other networks because the IoT devices are heterogeneous, low-powered and have resource constraints making the problem of dealing with these attacks much more complex (Aswale et al. 2019). Another important aspect related to the IoT networks is the ownership and/or custodianship of devices which is not clear and regulated, adding confusion about responsibility in case of any attack on confidentiality, integrity and availability of data and information. Moreover, the trust of users on the security aspect of these IoT devices and the network is critical since all

*Correspondence:
Martine Bellaiche
martine.bellaiche@polymtl.ca
Computer and Software Engineering Department, École Polytechnique, Montreal, Canada

interconnected nodes and devices exchange, store, traverse, share, transform, and translate data which makes it vital to answer the big question of building and maintaining end-user trust.

Along with the information storage and management challenges faced by IoT, security and privacy issues, including confidentiality, integrity, and availability, are the most critical requirements that need to be addressed in the early phase conducted to design the architecture of a smart system. IoT networks face various security attacks that can compromise the prime functionality of data storage and communication by risking users' data privacy. Attacks like Denial of Service (DoS) will render a complete network or specific nodes unavailable, disrupting the information flow. Attacks like scanning, enumeration, and reconnaissance will help attackers compromise user privacy by revealing sensitive data, which would be helpful in tearing down different aspects of security associated with the IoT network. IoT devices are a lucrative target for attackers because of the following reasons:

1. The contact internet connectivity, i.e. 24/7 switched-on state of the IoT devices raises the probability of attacks as it is easy for the attackers to access the device
2. Security hardening in a network composed of different sorts of IoT devices is a cumbersome task in comparison to hardening a single machine.
3. Weak or no encryption along with weak or default passwords are a major cause of attacks in most IoT devices
4. The interconnected nature of these devices adds to the vulnerability exposure as multiple devices can be accessed from one compromised device.

The basic IoT architecture comprises five layers Perception, Transport, Processing, Application and Business. These layers are responsible for performing different tasks on data as it moves from one layer to the other (Agarwal et al. 2022). Moreover, all these layers have security issues that must be addressed to guarantee safe and secure data flow between these IoT devices. These security issues target not just the data and information being traversed on the network but also the network protocols and the IoT devices. Table 1 explains the IoT layers, their purpose and security issues at all these layers in detail.

The most obvious solution to tackle such attacks is to build an Intrusion Detection System (IDS) which first detects and then thwarts these attacks (Saha et al. 2020). Researchers in recent years have focused on protecting IoT devices from all sorts of attacks by proposing various intrusion detection solutions that guarantee data security and privacy and build users' trust. Initially, these IDS solutions were signature-based, which could only detect anomalies based on a set of predefined rules and signatures. These IDS solutions perform well against

Table 1 IoT layers and security issues

Layers	Purpose	Security Issues
Perception	The perception layer represents the physical IoT devices like health monitors, security systems, smart home devices, autonomous vehicles and robotics-related devices	Eavesdropping Fake/malicious nodes Node capturing/Jamming Replay attacks Timing related attacks Social engineering attacks
Transport	This layer is responsible for sending the collected data to either the cloud or edge device for further processing. It relies on internet gateways for moving data from the perception layer for onward processing	Denial of service (DoS) attacks Main-in-the-middle (MiTM) attacks Storage related attacks Exploitation attacks RPL attacks
Processing	Once the data reaches the cloud or edge device the server transforms this data into information. Modern IoT architectures use ML and AI for data processing and analysis	Malware-related attacks Exhaustion attacks Resource depletion attacks
Application	At this layer the Network administrators manage IoT device orchestration, create rule sets, and set service-level agreements for their IoT architecture	Cross site scripting (XSS) attacks Malicious code attacks Network disturbance Data loss Code Injection Buffer overflow attacks Phishing attacks Side-channel attacks
Business	At this layer, information is transformed into business intelligence that drives decision-making by executives and stakeholders	Business logic-related attacks Zero-day attacks

already-known intrusions but fail against sophisticated attacks. Moreover, with the increased frequency of zero-day attacks, signature-based solutions have become obsolete and ineffective. Lately, researchers have shifted towards more innovative techniques like anomaly-based IDS that involve machine learning and deep learning techniques to thwart unknown network intrusions. The anomaly-based IDS is based on two phases, training and testing. In the training phase, a model is trained on known data which is later tested on unseen data in the testing phase. However, the attackers have kept themselves abreast with the latest defensive techniques and equipped themselves with more advanced techniques to compromise security. One such method widely adopted by attackers is packet obfuscation and encryption techniques. The proposed research presents an Intrusion Detection approach for IoT environments based on an ensemble deep learning method using the Lambda architecture.

The main research contributions of the proposed framework are as follows:

1. We have developed a Hybrid Intrusion Detection System (IDS) that is capable of discriminating between normal network traffic and attack traffic of five different types, in a real-time environment effectively.
2. In order to process the real-time traffic, our system employs a two-staged strategy, i.e. a quicker binary classifier that differentiates between normal and attack traffic, and in the second stage, the attack traffic is further classified into different types of attacks.
3. Both classification models are trained using deep learning algorithms. Binary classifiers best perform on LSTM, whereas, in order to improve the accuracy of multi-classification, both deep and hybrid ensemble-based approaches were used.
4. System is developed and evaluated using Lambda Architecture having two modes, Batch Mode and Stream Mode. The model training is performed in Batch Mode, whereas the real-time traffic monitoring (using the trained models) is performed in Stream Mode.
5. Performance analysis is done in terms of accuracy as well as throughput. The results demonstrate that the multi-staged strategy ensures that the decisions required to be taken in real time are separated from the in-depth analysis and model training, thus improving the accuracy as well as the throughput.

The remainder of the paper is organized as follows: "Related work" section presents the related work in IDS-IoT. "Proposed framework" section presents a proposed framework in a Lambda architecture, including an overview of the architecture scheme and highlighting the advantages of utilizing Lambda architecture and the deep learning models to handle the stream and batch data classification to detect IoT environment attacks. "An evaluation performance of the proposed approach" section introduces the proposed approach's evaluation after our experiments utilizing the most common evaluation metrics. The last "Conclusion and future work" section includes concluding remarks.

Related work

Many research works have focused on detecting network intrusions in IoT using machine learning and deep learning approaches.

Ali and Cotae (2018) presents a deep learning-based IDS for classifying the traffic flow in the network. The research used binary and multiclass classification models to detect DoS, DDoS, information gathering and information theft attacks. Alenezi et al. (2021) presented a machine learning and deep learning-based approach to detect Denial of Service attacks in IoT networks. Docs (2020) presents a technique which uses a spider monkey optimization (SMO) algorithm along with a stacked-deep polynomial network (SDPN) for achieving optimal intrusion detection. The proposed work results in better accuracy and precision. Maniath et al. (2017), Fang et al. (2018) and Sarker et al. (2022) have used an ensemble approach using traditional machine learning and not deep learning approaches for detecting network intrusions in IoT. In Gustavsson (2019) the author used the CICIDS2017 dataset and Zeek tool to convert PCAP files into log files. Initially, the author uses two sets of features to detect network attacks. Simple and complex features consist of 10 and 50 features, respectively. Later, the result of both sets barely differed, so the author used a simpler set of features instead for improved speed. In Sarhan et al. (2020), they converted four widely known datasets into NetFlow. NetFlow is an industry-standard protocol for network traffic collection. Zeek extracted 49 and 44 features from UNSW-NB15 and ToN-IoT, respectively. Argus extracted 42 features from BOT-IoT, while 73 features were extracted from CSE-CIC-IDS2018 using CICFlowMeter-V3. Random forest was used to perform binary and multiclass classification. The binary classifiers performed better on all four datasets. However, the classifier did not perform well on some datasets. The authors decided to increase the number of features for better performance. Similarly, in another study (Otoum et al. 2022), authors presented a deep learning-based intrusion detection system for IoT devices that uses an optimization algorithm known as spider monkey optimization (SMO) algorithm integrated with the stacked-deep polynomial network (SDPN) where SMO performs the optimal features selection, and SDPN performs the classification. They performed detection for denial of service (DoS), user-to-root (U2R) attacks, probe attacks, and remote-to-local (R2L) attacks. In another study, Khan et al. (2022) performed a critical analysis of various deep learning algorithms for IoT environments using various network-based datasets. In another work reported in 2021, Idrissi et al. (2021) presented a deep learning-based algorithm using the CNN network to detect Botnet attacks. They claimed that the results produced by their methods are 99.04% accurate with a prediction time is 0.34ms.

Ahmad et al. (2022) presented an ensemble deep learning approach by combining Deep Neural Network (DNN) and Decision Tree (DT) classifiers. The approach does not identify attack types based on a binary classification approach. Martins et al. (2022) also presented an ensemble approach using machine learning techniques for detecting attacks in IPv6 Routing Protocol for low-powered and lossy IoT networks with high accuracy. Malik et al. (2022) also presented an ensemble anomaly detection approach to detect DoS, DDoS, reconnaissance and key-logging attacks by combining a C5 classifier and a One-Class Support Vector Machine classifier. Pan et al. (2022) proposed an ensemble-based approach based on Deep Belief Networks and issued the critical problem of redundant features by adopting a Minimized Redundancy Discriminative Feature Selection approach.

Siddique et al. (2018) presented a deep learning-based ensemble approach by combining two Dimensional Convolutional Neural Network (2-D CNN), Artificial Neural Network (ANN) and Multi-Layer Perceptron (MLP) models. The proposed work covers major cyber-attacks with accuracy. The proposed approach lacks real-time detection and rather works on a benchmark dataset. Ghimire and Rawat (2022) presented an improved CNN and LSTM-based approach to detect Distributed Denial of Service Attacks (DDoS) in social media and outperforms the classic implementation of CNN and LSTM. The proposed work does not cover other prominent cyber-attacks and does not rely on Lambda architecture for model training and testing. Mehedi et al. (2022) used CNN and LSTM models along with character encoding for Spatial Feature Learning (SFL) techniques to differentiate obfuscated and encrypted HTTP traffic from normal traffic. The technique is not ensemble based and focuses on obfuscated packets only. Prabha and Kumar (2022) made use of Deep Belief Networks (DBNs) to detect intrusions in the Edge-of-Things (EoT). Table 2 shows the comparison of existing IDS approaches for IoT.

IoT networks in real-world environments generate massive amounts of data that traditional methods cannot handle, thereby requiring some advanced mechanism to be deployed for handling that massive amount of data. For that purpose, one obvious choice is to use the Lambda architecture (Hertel et al. 2020), which provides a regimented methodology for processing massive quantities of data by providing access to batch-processing and stream-processing methods using a hybrid approach. Researchers have employed Lambda architecture for proposing IDS solutions for big data generating IoT environments to save the implementation and computational cost and bring efficiency (Lahasan and Samma 2022).

Ma (2020) made use of the Lambda architecture for dealing with massive amounts of data generated by the Smart home environments. Lambda architecture has also been used for quality assessment purposes in smart road networks (Roopak et al. 2019) and for real-time visualization of sensors generated data in smart manufacturing environments (Diro and Chilamkurti 2018). Amanullah

Table 2 Existing IDS approaches for IoT

Research work	Ensemble model	Multi-class classifier	Attacks covered	Lambda architecture
Ali and Cotae (2018)	x	✓	DoS, DDoS, reconnaissance, information stealing	x
Alenezi et al. (2021)	x	x	DoS Attacks	x
Docs (2020)	x	✓	DoS, user-to-root (U2R) attack, remote-to-local (R2L) attack, probe attacks	x
Ahmad et al. (2022)	✓	x	Cyber attacks	x
Martins et al. (2022)	✓	x	Sybil, sinkhole, blackhole, cloning , selective forwarding, hello flooding and local repair attacks	x
Malik et al. (2022)	✓	x	DoS, DDoS, reconnaissance and key-logging attacks	x
Pan et al. (2022)	✓	x	Network attacks	x
Siddique et al. (2018)	✓	x	Nine common cyber attacks	x
Ghimire and Rawat (2022)	✓	x	DDoS attacks	x
Mehedi et al. (2022)	x	x	Web intrusions, obfuscation attacks	x
Prabha and Kumar (2022)	x	x	Network intrusion attacks	x

et al. (2020) employs the lambda architecture for devising a Security Operations Centre (SOC) to thwart adversarial attacks using extreme learning machine neural network with Gaussian Radial Basis Function kernel. The proposed approach is not for IoT environments. Patan and Babu (2018) present a cloud-based Snort network IDS using docker containers and lambda architecture for detecting and visualizing network-related intrusions. Lopez et al. (2018) proposed a technique that combines cloud computing and distributed stream processing techniques to accurately and quickly detect network threats. The proposed approaches are not for IoT environments. Kayode (2020) present a cloud-based security solution for transmitted data in an IoT smart plug using the lambda architecture. Tangsatjatham and Nupairoj (2016) proposed a log anomaly detection technique that uses Apache Spark for data processing and Apache Flume for data collecting. Using the Lambda architecture helps support log processing in large environments. Table 3 presents further analysis of Lambda architecture to address big data issues.

The main shortcomings in the existing IDS approach in IoT are listed below:

1. Most research works focus on a single or limited number of IoT attacks and fail to cover a wide variety of IoT attacks.
2. Most research works do not focus on detecting intrusions in real-time and rather perform all experimentation on offline datasets
3. Most proposed works only rely on single binary classifiers differentiating between malicious and benign packets and fail to detect the type of IoT attacks
4. Many research works do not employ Ensemble based techniques for achieving better accuracy and precision.
5. Many proposed research works have not been evaluated and developed using big data architectures.

The proposed research work addresses all these issues by presenting a deep ensemble IDS approach using lambda architecture and is the extension of our previously published work (Alghamdi and Bellaiche 2021).

Proposed framework

Recently, several papers over-viewed model buildings to classify streaming data. This section highlights data processing, data storage, and data ingestion systems that can be used to build the Lambda architecture to handle the stream and batch data for classification purposes using deep learning approaches to detect attacks in IoT devices. Moreover, data preparation, feature selection, training steps, and the detection process are covered.

Datasets used

The selection of datasets plays a vital role in training any deep learning or ensemble-based classifier. The more comprehensive the dataset, the more elaborately and accurately the trained model performs on the unknown dataset. In this proposed research work, we have used a prominent benchmark dataset publicly available and has widely been used by security researchers in devising IDS solutions for IoT-based environments. IoT-23 (Str 2020) is used, which was published in Jan 2020, is a dataset of network traffic from the Internet of Things (IoT) devices. The dataset comprises malware-affected and benign traffic for IoT devices, with 20 and 3 captures, respectively for each type of traffic. The complete size of the dataset is 20GB. The dataset is distributed in a way that 60% of it was used for training, 20% for validation (dev-set) and the remaining 20% for testing purposes.

Table 3 Using Lambda architecture to address big data issues

Research work	Application area	Relevant to IoT	Weaknesses
Ma (2020)	Smart home environments	✓	Does not address security and network attacks. Focuses on data processing, management and storage
Roopak et al. (2019)	Smart road environments	✓	Does not address security and network attacks. Focuses on quality assessment using large-scale data
Diro and Chilamkurti (2018)	Smart manufacturing environments	✓	Does not address security and network attacks. Focuses on real-time visualization of sensors generated data
Amanullah et al. (2020)	Security operations centre (SOC)	x	Does not employ simple or deep-ensemble based approaches. Not designed for IoT environments
Patan and Babu (2018)	Cloud-based snort network IDS	x	Not designed for IoT environments
Lopez et al. (2018)	Network threat detection	x	Not designed for IoT environments
Kayode (2020)	Cloud-based security solution	x	Not designed for IoT environments
Tangsatjatham and Nupairoj (2016)	Log anomaly detection technique	x	Not designed for IoT environments

Data pre-processing

After reading all the IoT-23 dataset files, we pre-processed the dataset as follows:

- Dropping useless features like “ts”,“uid”, “id.orig_h”, ’id.orig_p”. Using those features can lead to overfitting. Removing such features will help us achieve a generalizable solution to our main problem.
- Dropping features which hold only NaN values. Features are: “local_resp”, “local_orig”, “history”.
- Dropping duplicated rows as those can appear in both training and testing sets and can also lead to overfitting.
- Mapping label names to their correct labels as they were completely messed.
- Imputing NaN values for features origin_bytes and dest_bytes by -1 to make it easier for the model to distinguish those values as missing or as non usual values.
- Encoding Categorical features using Ordinal encoding for features proto, service, conn_state. Using ordinal encoding. Ordinal encoding allows us to encode features in a way that features do not have any ordered relationship between them.

Target features are encoded as follows:

- Benign:0
- C &C:1
- PartOfAHorizontalPortScan:2
- DDos:3
- Okiru:4

The final dataset contains 8,887,466 rows and 13 columns. Random undersampling was performed to correct imbalanced data to reduce the risk of their analysis or machine learning algorithm skewing toward the majority and having fair results for all the classes. At the time of training the machine and deep learning algorithms, we applied MinMaxScaling in [0,1] range to the dataset to improve the quality of our data and help models converge faster.

Lambda architecture

Lambda architecture provides a structured way of dealing with and processing large amounts of data in real-time, making it a suitable architectural solution for dealing with IoT environments. The use of this architecture adds flexibility, scalability, agility and high availability to the system. The basic working of the lambda architecture is explained in Fig. 1 while the layer-wise details are mentioned in Fig. 2.

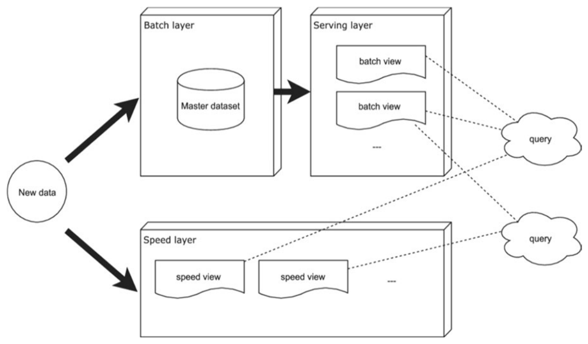
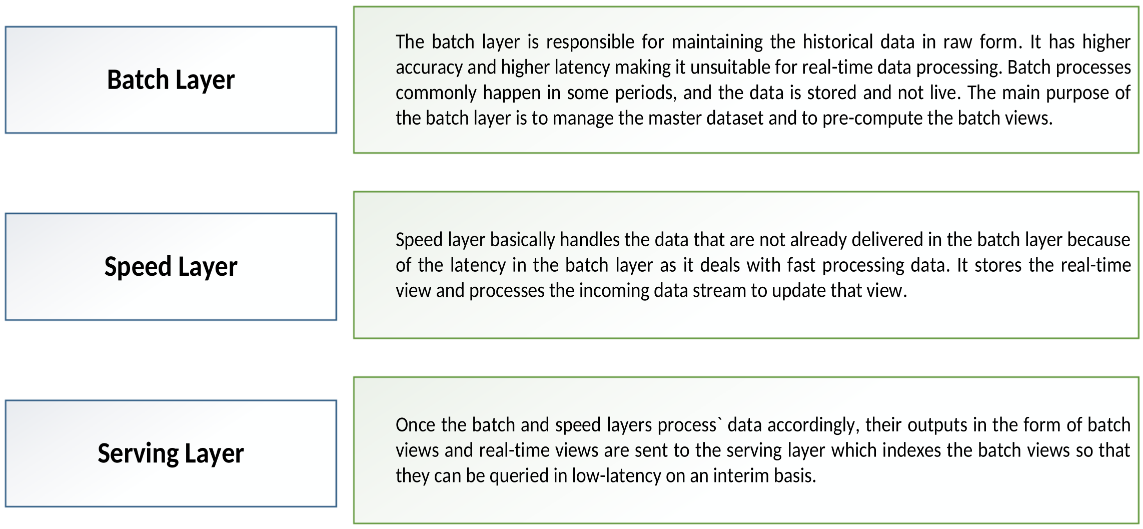


Fig. 1 The Lambda architecture (Yang et al. 2017)

It is a data-processing architecture that uses both batch and stream processing to manage large amounts of data. By employing batch processing to provide thorough and accurate views of batch data and real-time stream processing to provide views of live data, this approach to design aims to strike a balance between latency, throughput, and fault tolerance. It is possible to combine the two view outputs before display. The lambda architecture is associated with the development of large data, real-time analytics, and the desire to reduce map latency.

Feature engineering

Selecting features from the network traffic is an important step as it directly affects the performance of our classifiers. These features deeply reflect the nature of network traffic and contain detailed information and characteristics of the network-related benign and malicious traffic. Selecting features for the binary classifier is different from selecting network features for the multi-class classifier because, in the later case, different features help in determining different kinds of network intrusions. For instance, a feature might be very useful in determining DoS attacks but not that useful for detecting malicious code attacks. Therefore, the selected network features must be related to the type of attack under study. Similarly, all network features do not play a part in determining anomalous traffic, as few network features might not be helpful at all in determining anomalous traffic. Since the dataset under consideration had many features which were categorical and deep learning engines are good at working with numerical data, therefore, feature transformation was required. For that purpose, we used the *Fit_Transform*, which is available in Scikit_Learn Python library (Bisong 2019). The encoder assign values ranging from *0* to *n-1* where *n* is the total number of records under consideration.



Batch Layer

The batch layer is responsible for maintaining the historical data in raw form. It has higher accuracy and higher latency making it unsuitable for real-time data processing. Batch processes commonly happen in some periods, and the data is stored and not live. The main purpose of the batch layer is to manage the master dataset and to pre-compute the batch views.

Speed Layer

Speed layer basically handles the data that are not already delivered in the batch layer because of the latency in the batch layer as it deals with fast processing data. It stores the real-time view and processes the incoming data stream to update that view.

Serving Layer

Once the batch and speed layers process` data accordingly, their outputs in the form of batch views and real-time views are sent to the serving layer which indexes the batch views so that they can be queried in low-latency on an interim basis.

Fig. 2 Layers of Lambda architecture

Batch layer: training of classification models

The batch layer is the lambda architecture's high latency and high accuracy layer, which deals with batches of data, not in a real-time environment. We have utilized these characteristics of the batch layer for training both classifiers in the batch layer using a publicly available benchmark dataset. In this regard, we are not worried about the latency and time associated with training these classifiers. The entire training process at the batch layer is further highlighted in Fig. 3.

For training both the binary and multi-class classifiers, we used three deep learning classifiers ANN, CNN and LSTM.

Convolutional neural networks (CNN)

A CNN comprises one or more convolutional layers, which are further linked by one or more fully connected layers (Shahid et al. 2022). Here the input and output layers are combined through multiple hidden layers, which generally contain a sequence of convolutional layers. CNN's are good at detecting malware attacks, malicious scripts and codes and finding abnormal traffic patterns.

Long short term memory (LSTM)

LSTM is a particular type of recurrent neural network which are capable of learning long-time dependencies (Azumah et al. 2021). Their default behavior is to

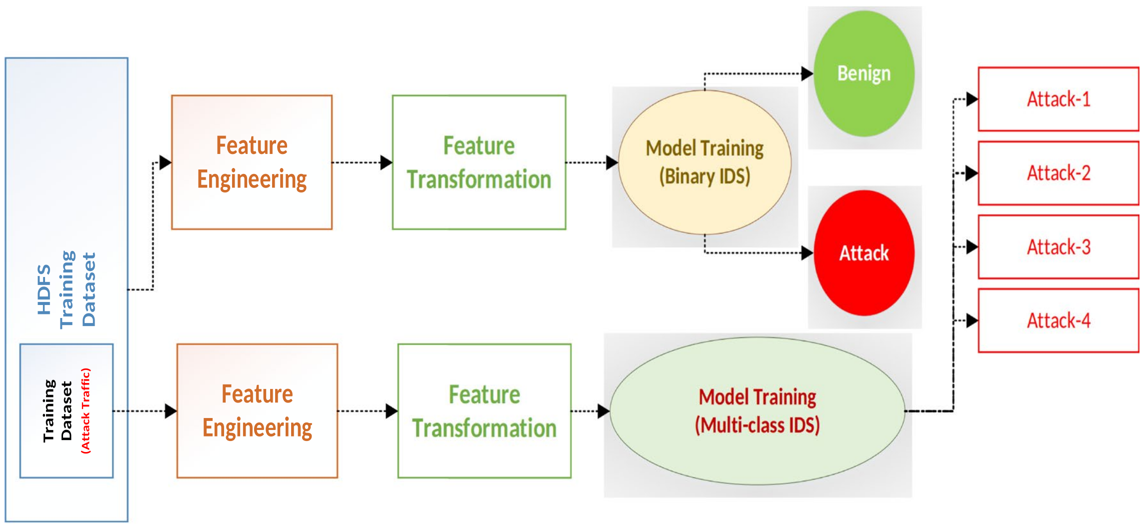


Fig. 3 Training classifiers in batch layer

remember information for longer periods of time. This is possible because of four neural network layers in the repeating module, which interact in a special way. They are useful in detecting attacks related to state maintenance like Cross Site Scripting, DDoS and other application layer attacks.

Binary classifier

In order to train the binary classifier in the batch layer, we individually applied and trained all three deep-learning classifiers on the training dataset. We later found out in the testing phase that the LSTM binary classifier outperforms the CNN and ANN-based binary classifiers in terms of accuracy and intrusion detection. This classifier only detects whether a network packet is benign or malicious and does not detect the type of attack. If a packet is found malicious, it is the responsibility of the ensemble-based multi-class classifier to detect its type. Since in the real-time majority of the network traffic in IoT environments are benign, this approach will save useful processing time as benign packets do not have to pass through the ensemble-based multi-class classifier. The detailed working of the Binary Classifier is shown in Algorithm 1.

Algorithm 1 Phase:1 - Discriminating between Normal and Attack Traffic

```
Input: traffic_stream $\mathbf{x} = (x_1, x_2, ... x_n)$
Define:
$\eta$: set_of_features
$\epsilon$: transformed_features
Output: traffic_type T
procedure BINARY_CLASSIFICATION(x)
    $\eta$ = extract_feature (x)
    $\epsilon$ = feature_transformation ($\eta$)
    T ⟵ ApplyLSTM($\epsilon$)
    if (T = Normal) then
        execute
        continue
    else
        multi_classification(T)    ▷ Calling the method to find the attack type (Algo-2)
    end if
end procedure
```

Deep ensemble multi-class classifier

In order to train a classifier to perform multi-class classification, we initially used three individual classifiers, i.e. LSTM, CNN and ANN. We performed their hyper-parameter tuning as well. However, the best performance was 98.20%, achieved in the case of LSTM. In order to further improve the performance, we used *Majority voting*, an ensemble technique shown in Algorithm 2. The majority voting relies on the performance of the individual learners (called as base learners at Level 0), and produces the final prediction based on the majority voting at Level-1 (as shown in Fig. 4. The results achieved using this ensemble technique gave an accuracy of 98.20%. Later, in order to improve the accuracy and decrease the computational complexity of deep learning classifiers, we used another ensemble approach by combining CNN with the Random Forest classifier and the Decision Tree classifier, as shown in Fig. 5. This approach gave an accuracy of 99.93%.

Algorithm 2 Phase-2 - Analyzing the Attack Traffic to find Attack Type

```
Input: transformed_features $\epsilon$;
Define:
$\epsilon$: transformed_features,
$\alpha$: prediction_of_classifier1,
$\beta$: prediction_of_classifier2,
$\gamma$: prediction_of_classifier3,
$A$: prediction_of_MajorityVoting
Output: attack_type A
procedure MULTI_CLASSIFICATION($\epsilon$)
    $\alpha$ ⟵ ApplyClassifier1($\epsilon$)
    $\beta$ ⟵ ApplyClassifier2($\epsilon$)
    $\gamma$ ⟵ ApplyClassifier3($\epsilon$)
    $A$ ⟵ MajorityVoting($\alpha$,$\beta$,$\gamma$)  ▷ depicts the final prediction of attack type
end procedure
```

Also we adopted another technique which is weighted ensemble as shown in Figs. 6 and 7 and reflected in Algorithm 3 that is based on combining predicted probabilities for each model based on the equation:

$$\widehat{y} = argmax(sum(p_i * w_i))$$

where p_i is the predicted probabilities for classifier i and w_i is the weight that can be assigned to the classifier i where $sum(w_i) = 1$. This approach gave an accuracy of 99.6%.

Since this classifier is only responsible for detecting the type of attack as it always receives malicious network

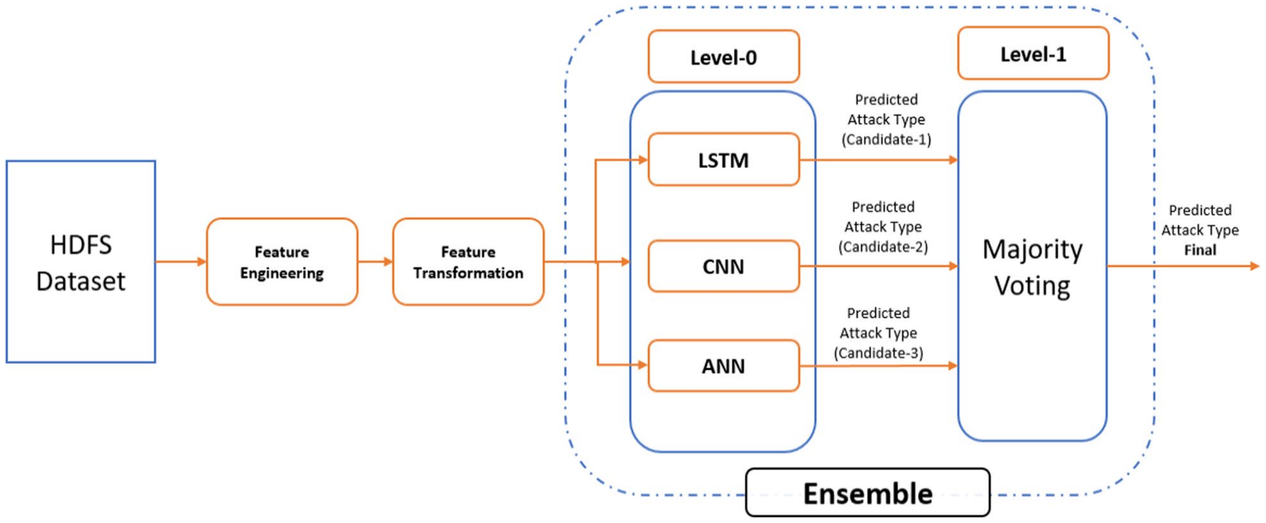

HDFS Dataset
Feature Engineering
Feature Transformation
Level-0
LSTM
CNN
ANN
Predicted Attack Type (Candidate-1)
Predicted Attack Type (Candidate-2)
Predicted Attack Type (Candidate-3)
Level-1
Majority Voting
Predicted Attack Type Final
Ensemble

Fig. 4 Deep ensemble configuration for multi-class prediction

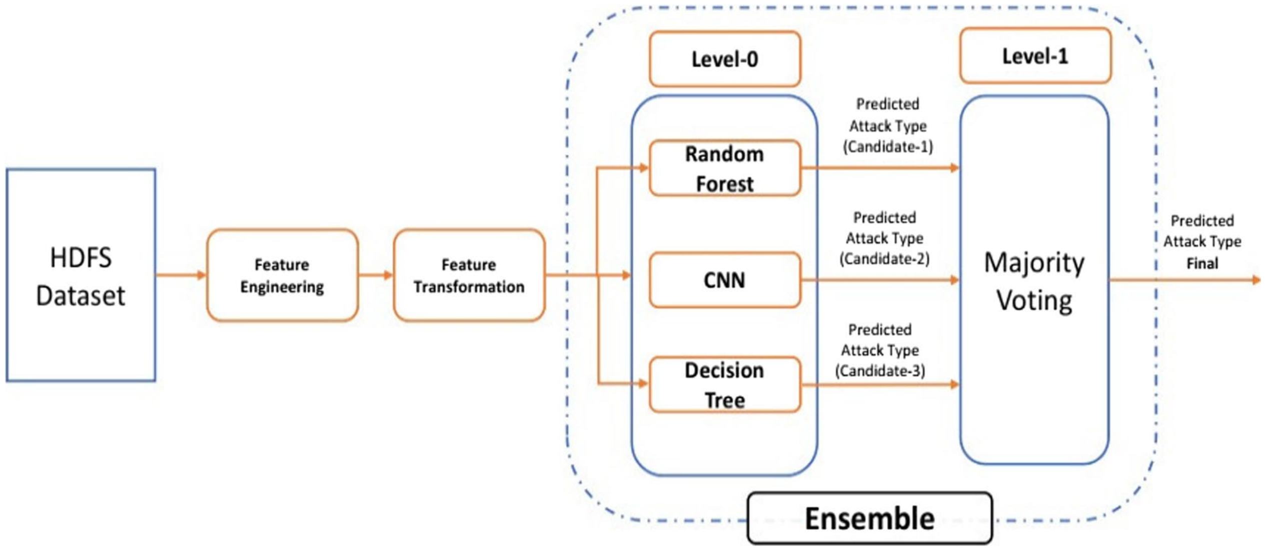

HDFS Dataset
Feature Engineering
Feature Transformation
Level-0
Random Forest
CNN
Decision Tree
Predicted Attack Type (Candidate-1)
Predicted Attack Type (Candidate-2)
Predicted Attack Type (Candidate-3)
Level-1
Majority Voting
Predicted Attack Type Final
Ensemble

Fig. 5 Hybrid ensemble configuration for multi-class prediction

traffic, therefore we created a subset of the training dataset in a way that it only contains the attack traffic of multiple IoT network attacks and does not contain benign samples. Table 4 gives details about hyper-parameters of all three models in both binary and multi-class classifiers.

All the models were trained using Rectified Linear Unit (ReLU) as an activation function for all layers except for the last, where we used the sigmoid function for the binary classification models and the softmax function for the multi-classification models. The number of epochs was fixed at 300 epochs for all the models, batch size = 64, learning rate = 0.001, with Adam used as an optimizer. In addition, we used an early stopping callback with parameter patience set to 10 and minimum change in the monitored quantity to qualify as an improvement = 0.001 to prevent overfitting where we were tracking the validation loss.

Table 4 Hyper-parameters detail in all classifiers

Model	Classification	Hyperparameters
ANN	Binary	Four hidden dense layers 128, 256,256 and 128 nodes are set with 1 final output dense layer with 1 node
	Multi	Four hidden dense layers 128, 256, 256, and 128 nodes are set with 1 final output dense layer with 5 nodes
CNN	Binary	2 Conv2D filters=32, kernel_size=(1,3) 2 Conv2D with filters=64, kernel_size=(1,3) 2 hidden dense layers 256,512 2 MaxPool2D with 1 final output dense layer with 1 node
	Multi	2 Conv2D filters=32, kernel_size=(1,3) 2 Conv2D with filters=64, kernel_size=(1,3) 2 hidden dense layers 256,512 2 MaxPool2D with 1 final output dense layer with 5 node
LSTM	Binary	3 LSTM layers 60,120,120 with final output dense layer with 1 node
	Multi	3 LSTM layers 60,120,120 with final output dense layer with 5 node

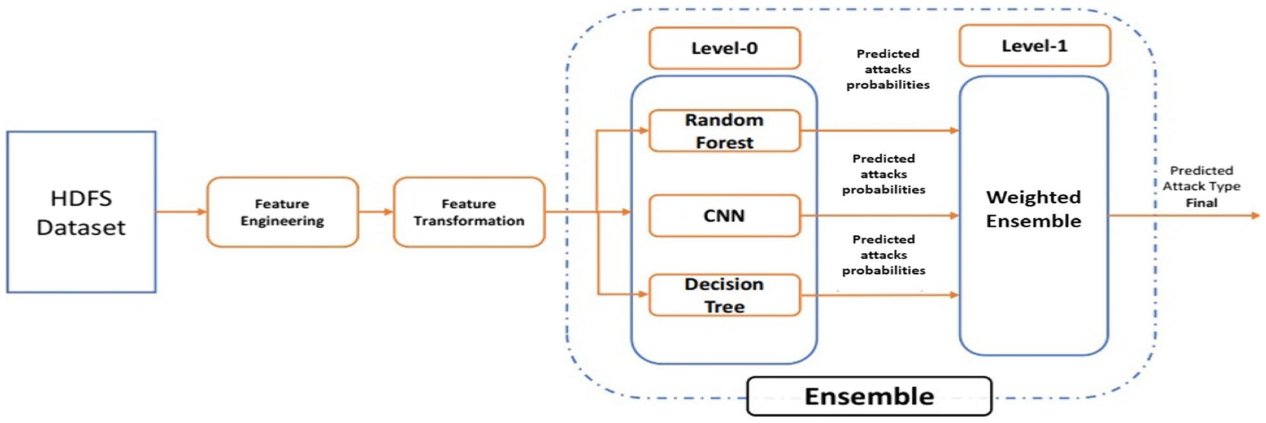


Fig. 6 Deep ensemble configuration for multi-class prediction

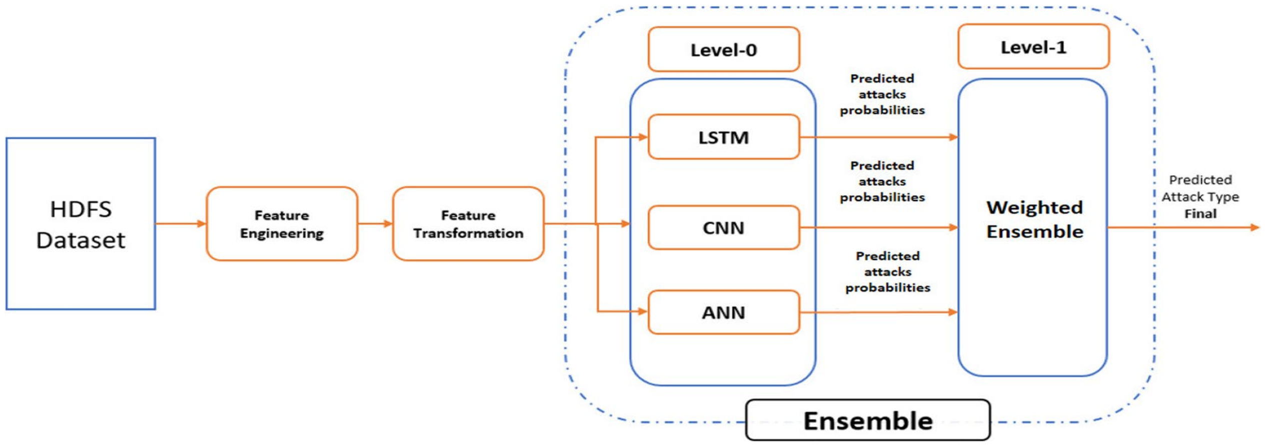

HDFS Dataset
Feature Engineering
Feature Transformation
Level-0
LSTM
CNN
ANN
Predicted attacks probabilities
Predicted attacks probabilities
Predicted attacks probabilities
Level-1
Weighted Ensemble
Predicted Attack Type Final
Ensemble

Fig. 7 Hybrid ensemble configuration for multi-class prediction

Algorithm 3 Phase-2 - Analyzing the Attack Traffic to find Attack Type -Weighted Ensembling

```
Input: transformed_features $\epsilon$;
Define:
$\epsilon$: transformed_features,
$\alpha$: coefficient_of_classifier1,
$\beta$: coefficient_of_classifier2,
$\gamma$: coefficient_of_classifier3,
Output: attack_type A
procedure MULTI_CLASSIFICATION($\epsilon$)
    classifier1probabilities $\leftarrow$ ApplyClassifier1($\epsilon$)
    classifier2probabilities $\leftarrow$ ApplyClassifier2($\epsilon$)
    classifier3probabilities $\leftarrow$ ApplyClassifier3($\epsilon$)
    A $\leftarrow$ argmax($\alpha$*classifier1probabilities +$\beta$*classifier2probabilities +$\gamma$*classifier3probabilities)    $\triangleright$ $\alpha+\beta+\gamma=1$
end procedure
```

Speed layer: analysing network traffic in real time

The speed layer in the lambda architecture deals with a real stream of data and is good at dealing with problems which need minimizing latency. This layer works efficiently in a limited time and handles data that is not delivered to the batch layer because of its latency issues. Here the ingress data stream is fast processed, and the real-time view of data is promptly updated. In the speed layer, we made use of the model inferences we earlier trained at the batch layer for analyzing network intrusions in real-time. The binary and multi-class classifier's inference engines are exposed to a data stream, as shown in Fig. 8.

Emulation of real-time data using Kafka

For real-time emulation of data, we used a popular event streaming platform Kafka which is used to provide data integrity and sequence in real-time stream data. Kafka does this event streaming of IoT network's data in a highly scalable, elastic and fault-tolerant way. The processed data can be stored in the cloud for later offline processing, and analysis (Carnero et al. 2021). The loud storage feature provided by Kafka helps further refine the overall system and understand the attack paradigm during later analysis.

Model inference (binary classifier)

The incoming data stream is first routed toward the binary classifier's inference engine, which categorizes malicious and benign packets. As discussed earlier, this engine will not detect the type of network attack/intrusion. Traffic packets categorized as benign will pass on smoothly, while those detected as malicious will be routed toward the multi-class classifier.

Model inference (multi-class classifier)

Once the binary classifier detects a network packet or flows as malicious, it would be directed to the ensemble-based multi-class classifier, which is composed of ANN, LSTM and CNN deep learning models. This module will categorize the attack family and label the type of attack using a voting-based ensemble approach. All this

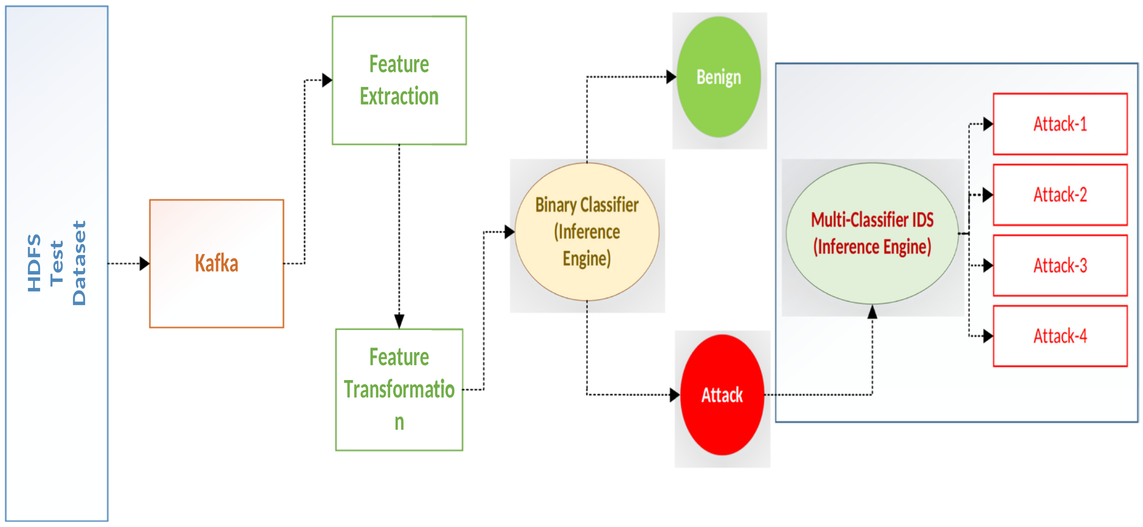


Fig. 8 Real time model deployment

Table 5 Model results for binary classification

Model	Type	Precision	Recall	F1-score	Accuracy	Processing Time (ms)
ANN	Normal	91.58	79.33	85.01	94.54	0.025
	Attack	95.14	98.23	96.66		
CNN	Normal	92.08	79.05	85.07	94.58	0.03
	Attack	95.09	98.35	96.69		
LSTM	Normal	93.6	99.2	96.3	98.20	0.07
	Attack	99.1	92.3	95.5		

is happening in real time as Kafka emulates the testing traffic to look like real-world IoT traffic containing data streams.

An evaluation performance of the proposed approach

Environment setup

We implemented the proposed deep ensemble-based IDS model in Python 3.7 with Tensorflow 2.6. to validate the efficacy of the proposed architecture. The experiment was done on a core-i5 machine with 64-bit Operating System (OS) and 16GB RAM. The software stack contained Java (JDK) 11, Hadoop 2.7, Spark v3.0, Pyspark 3.0, and Kafka 2.6.

Evaluation metrics

We used the most critical performance indicators for evaluating the proposed IDS that uses deep learning models for attack detection. The most commonly used evaluation parameters by researchers are mentioned below:

1. *Recall* This evaluation metric measures the proportion of real positive instances that have been predicted positive (Miao and Zhu 2021) and can be calculated as described in Eq. 1.

$$Recall = \frac{TP}{TP + FN} \tag{1}$$

2. *Precision* Precision is also called predictive values as it refers to the proportion of predicted positive and negative results that are true positive and true negative results, respectively (Davis and Goadrich 2006). Precision is applied to a variety of areas to describe the performance model such as machine learning, data mining, and information retrieval. It is calculated using Eq. 2.

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

3. *Accuracy*Accuracy can be explained as the overall performance of the classification model (Visa et al. 2011). It can be calculated using Eq. 3.

$$Accuracy = \frac{(TP + TN)}{TP + FP + TN + FN} \tag{3}$$

4. *F1-score*It is also known as the F-measure that considers both precision and recall (Chicco and Jurman 2020). F1-score can be calculated using Eq. 4.

$$F1 = 2 \times \frac{(precision \times recall)}{precision + recall} \tag{4}$$

5. *Throughput* It refers to the number of results produced per unit of time. This is measured in units of flow in our case (Grochowski et al. 2004).

Performance of binary classifiers (LSTM, CNN, ANN)

For evaluating the binary classifier, we individually deployed and tested all three deep learning models to analyze their performance on the testing dataset. The model training and validation accuracy and loss were calculated, which helped realize the most suitable binary classification model with the highest performance evaluation parameters.

Table 5 shows all performance indicators for ANN, CNN and LSTM-based classifiers for the binary classification model. It is evident that LSTM outperforms both ANN and CNN-based classifiers in terms of accuracy and precision. For that matter, we eventually used only the LSTM in the binary classification model that was trained at the batch layer. The training and validation accuracy and loss of the LSTM classifier are reflected in Figs. 9 and 10 respectively. The time is for processing a single sample (Table 6).

Performance of multi-class classifiers (LSTM, CNN, ANN)

We initially trained, deployed and validated all three deep learning models individually for the multi-class

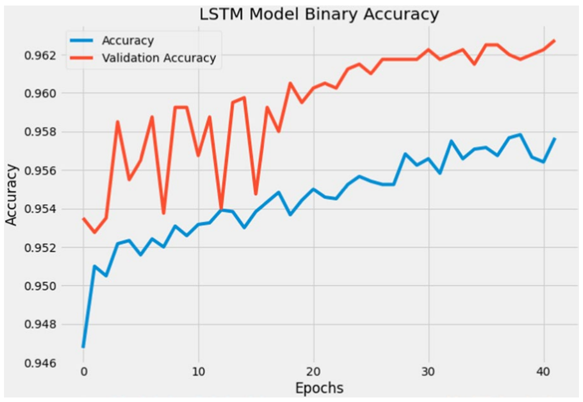


Fig. 9 Training and validation accuracy of LSTM

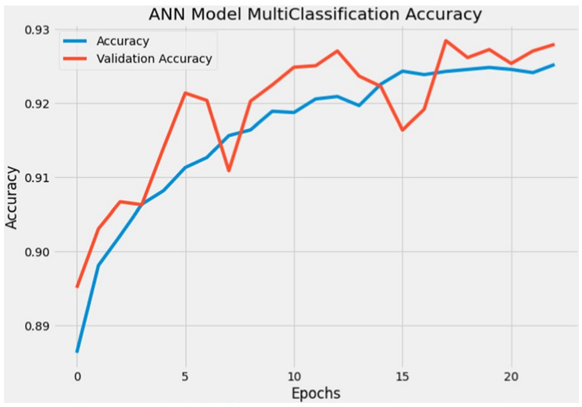


Fig. 11 Training and validation accuracy of ANN

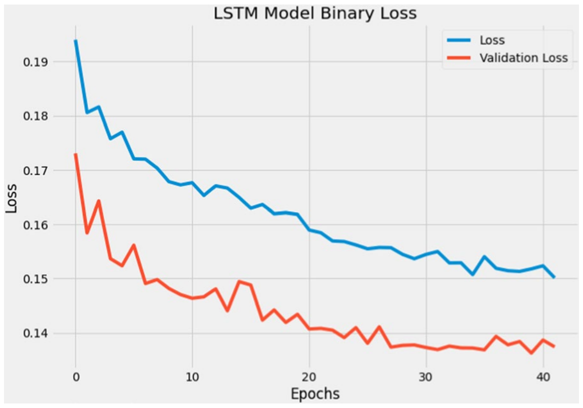


Fig. 10 Training and validation loss of LSTM

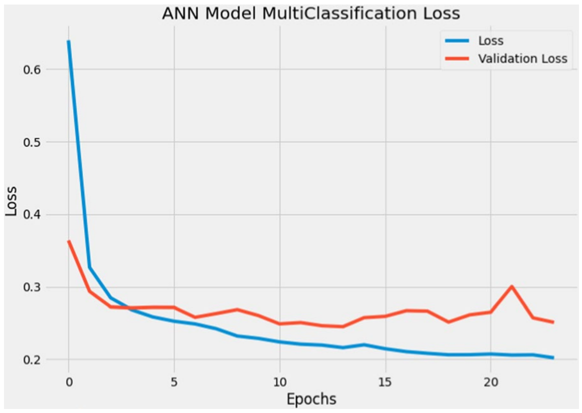


Fig. 12 Training and validation loss of ANN

classifier to analyze their performance in detecting various attack types in the testing dataset. The individual deployment was necessary to provide a rationale for choosing the ensemble-based approach. The training and validation accuracy and loss of the ANN classifier are reflected in Fig. 11 and 12 respectively, while that of CNN is shown in Figs. 13 and 14 respectively.

The training and validation accuracy and loss of the LSTM classifier are reflected in Figs. 15 and 16 respectively. It is evident that the performance of all three models in the multi-classification stage is close to each other. Moreover, the performance of LSTM is also not as good as it was in the binary classifier. This motivated us to go for an ensemble-based approach in the multi-class classification stage using all three deep learning models.

Table 7 provides performance details of the multi-class classifier with all three deep learning models individually deployed. It shows their performance in detecting the type of network attacks in an IoT environment.

We further experimented using all types of traffic, including the attack traffic and normal traffic (using 5 classes for traffic) in Table 8. We can see that the detection accuracy of splitting the detection into two stages, i.e. Binary followed by multi-class for attack-only traffic, is better than using the entire problem as a single multi-class problem with 4 types of attack and normal traffic.

Performance of ensemble-based approach

In order to further improve the accuracy of Multi-Classification, we have employed two variants of the ensemble (Majority Voting) and the weighted ensemble of three individual classifiers. The first one is the ensemble of three deep learning classifiers including ANN, CNN and LSTM, and the second variant is a hybrid ensemble that uses Random Forest, CNN and Decision Tree.

Although the time consumed by ensembles is much more than using only individual classifiers, however, since multi-classification is invoked only when the binary classifier has already detected an attack, the time consumption by the ensemble may be tolerated against the trade-off for better accuracy of detecting the attack type. The accuracy achieved by using the

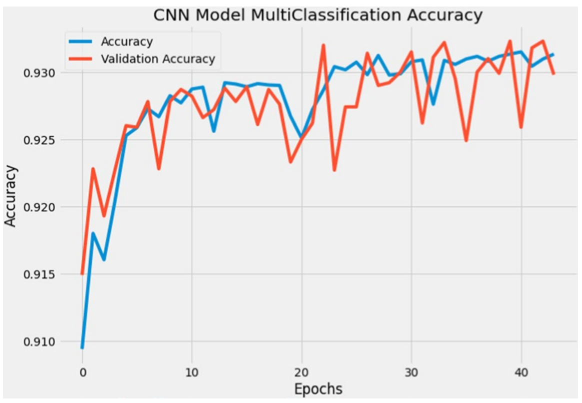


Fig. 13 Training and validation accuracy of CNN

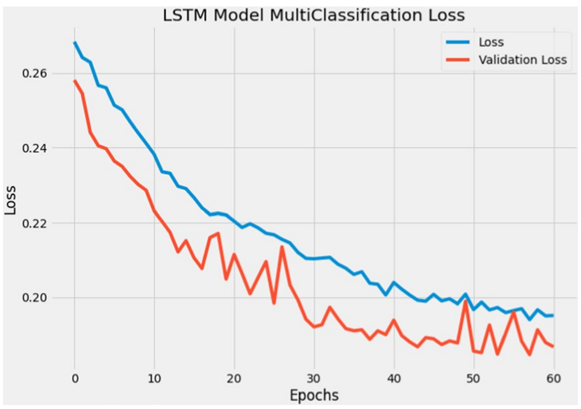


Fig. 16 Training and validation loss of LSTM

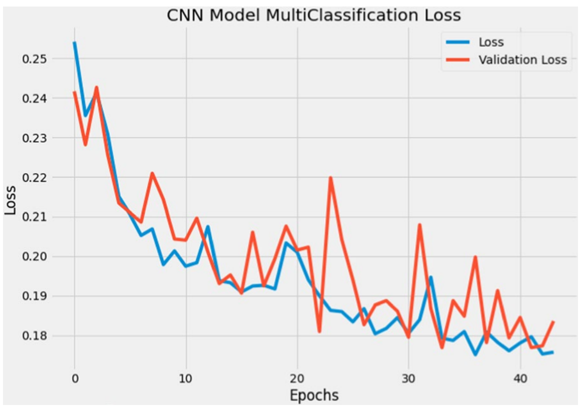


Fig. 14 Training and validation loss of CNN

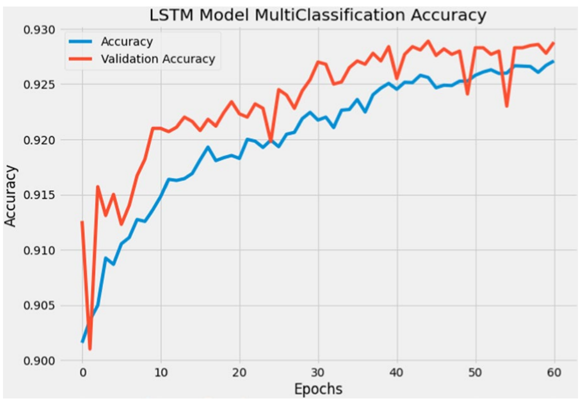


Fig. 15 Training and validation accuracy of LSTM

hybrid ensemble is 99.93% which is higher than the accuracy reported by individual classifiers. Table 6 shows the performance of applying the ensemble technique on Attack only data.

Analysing performance (throughput) of Lambda architecture

The prime intent of using the Lambda architecture was to enhance the system performance in terms of efficiency and throughput. We integrated the models into the Kafka-Spark streaming framework to generate a streaming process for evaluating processing time on the test dataset. The comparison of Tables 7 and 8 shows that the processing time in batch mode is slightly better than that for multi-classification without using Lambda architecture. Furthermore, the binary classification in binary mode, as shown in Table 5 is much quicker than batch mode or without using Lambda architecture.

As shown in Fig. 17, different time windows have been used to stream the data from 1 to 15 s. As expected, LSTM processes the data quickly as compared to other models in all window frame times for binary classification scenarios. However, in Fig. 18, it is evident that ANN works faster than other models and the LSTM performance decreases with increased length of the time window increases.

Table 6 Results for attack only data—attack classification in batch mode using ensemble classifiers

Model	Type	Precision	Recall	F1-score	Accuracy	Time (ms)
Weighted ensemble	DDoS	93.5	99.1	96.3	99.6	0.9
	Okiru	99.1	93.2	96.1		
	Port Scan	1.0	1	1		
	C &C	1	1	1		

Table 7 Results for attack only data—attack classification in batch mode

Model	Type	Precision	Recall	F1-score	Accuracy	Processing time (ms)
ANN	DDoS	89.5	99.6	94.3	96.9	0.09
	Okiru	99.6	88.2	93.5		
	Port Scan	99.9	1	99.9		
	C &C	1	1	1		
CNN	DDoS	89.6	99.7	94.4	97.0	0.05
	Okiru	99.7	88.8	93.6		
	Port Scan	99.9	1	99.9		
	C &C	1	1	1		
LSTM	DDoS	94.0	99.2	96.5	98.2	0.08
	Okiru	99.2	93.6	96.3		
	Port Scan	99.9	1.0	99.9		
	C &C	1.0	1.0	1.0		

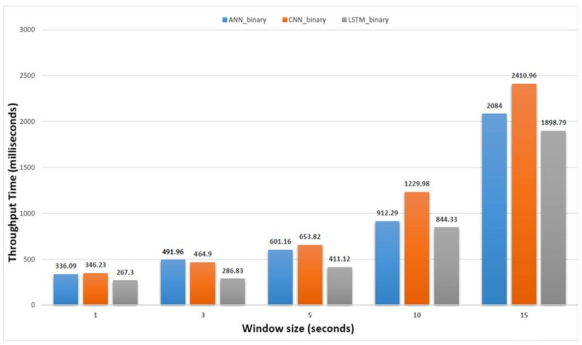


Fig. 17 Throughput of binary classification based on window size

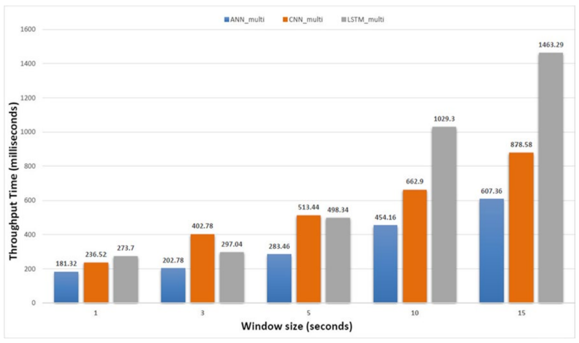


Fig. 18 Throughput of multi-classification based on window size

Table 8 Results for multi-classification without using lambda architecture—5 traffic types: 1 for benign and 4 for attacks

Model	Type	Precision	Recall	F1-score	Accuracy	Processing time (ms)
ANN	Normal	77.6	98.1	86.6	92.8	0.43
	DDoS	94.0	76.2	84.2		
	Okiru	98.8	90.4	94.4		
	Port Scan	99.4	99.0	99.2		
	C &C	99.6	1.0	99.8		
CNN	Normal	78.0	97.7	86.7	93.2	0.65
	DDoS	95.9	76.3	85.0		
	Okiru	98.6	91.9	95.1		
	Port Scan	98.8	99.7	99.3		
	C &C	99.9	1.0	1.0		
LSTM	Normal	78.7	96.35	86.6	92.8	0.13
	DDoS	91.1	79.1	84.6		
	Okiru	98.8	90.0	94.2		
	Port Scan	99.8	98.5	99.2		
	C &C	99.9	1.0	99.9		

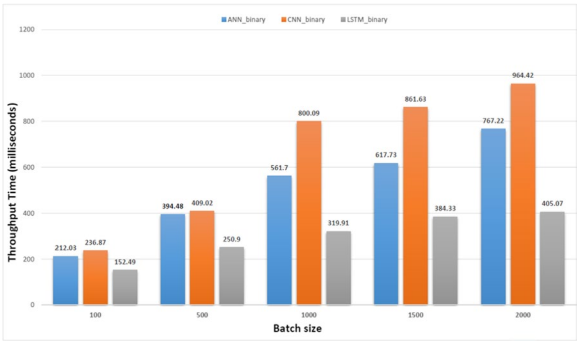


Fig. 19 Throughput of binary classification based on batch size

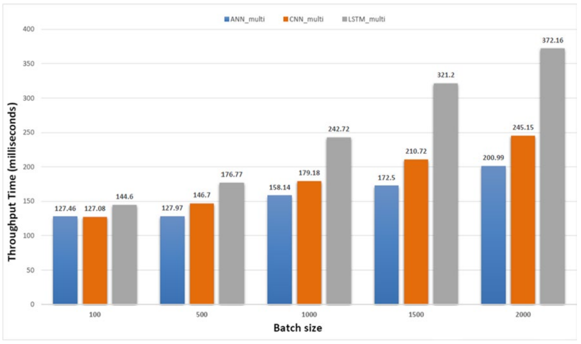


Fig. 20 Throughput of multi-classification based on batch size

We also evaluated all deep learning models based on batch processing time using the Spark framework. As shown in Fig. 19, batch sizes ranging from 100 until 2000 were used. It was noted that LSTM took less time to process the data than all other models in all batch sizes for binary classification scenarios, but ANN works faster than other models like LSTM, when the size of the batches increases, as shown in Fig. 20.

Conclusion and future work

The Internet of Things (IoT) has become a part of our daily lives because of the ease it brings in our everyday lives and in every field. This has enabled attackers and people with malicious intent to compromise the confidentiality, integrity and availability of these IoT devices and networks. In this research work, we have proposed a scalable and agile Intrusion Detection System (IDS) using multi-staged binary and multi-class classifiers using simple and ensemble-based deep learning approaches, respectively. The ensemble approach is based on both deep learning and hybrid machine and deep learning approaches to achieve very high detection accuracy. Moreover, we employed the Lambda architecture to add further efficiency and optimization in a way that training of both binary and multi-class classifiers takes place at the batch layer. At the same time, real-time IoT traffic is evaluated and analyzed in the low-latency speed layer with the help of model inferences. We also demonstrate that the ensemble approach results in higher detection accuracy and precision as compared to using the simple approach. We also demonstrate that using the Lambda architecture enhances system performance in terms of throughput.

In the future, we intend to employ more deep-learning approaches in the ensemble model to augment detection accuracy and system performance further. We also intend to test the proposed framework in a real-world production IoT environment to validate its performance further. Another important goal is to use Automated Machine Learning techniques for tuning the hyperparameters.

Author contributions

All authors read and approved the final manuscript.

Declarations

Competing Interests

The authors declare that they have no competing interests.

Received: 16 August 2022 Accepted: 8 December 2022
Published online: 02 March 2023

References

A labeled dataset with malicious and benign IoT network traffic, January 2020. https://www.stratosphereips.org/datasets-iot23
Agarwal V, Mishra P, Kumar S, Pilli ES (2022) A review on attack and security tools at network layer of IoT. Opt Wirel Technol 497–506
Ahmad R, Alsmadi I, Alhamdani W, Tawalbeh L (2022) A comprehensive deep learning benchmark for IoT IDS. Comput Secur 114:102588
Alenezi M, Nadeem M, Asif R (2021) SQL injection attacks countermeasures assessments. Indones J Electr Eng Comput Sci 21(2):1121–1131
Alghamdi R, Bellaiche M (2021) A deep intrusion detection system in Lambda architecture based on edge cloud computing for IoT. In: 2021 4th International conference on artificial intelligence and big data (ICAIBD), pp 561–566. IEEE
Ali O, Cotae P (2018) Towards DoS/DDoS attack detection using artificial neural networks. In: 2018 9th IEEE annual ubiquitous computing, electronics & mobile communication conference (UEMCON), pp 229–234. IEEE
Amanullah MA, Habeeb RAA, Nasaruddin FH, Gani A, Ahmed E, Nainar ASM, Akim NM, Imran M (2020) Deep learning and big data technologies for IoT security. Comput Commun 151:495–517
Aswale P, Shukla A, Bharati P, Bharambe S, Palve S (2019) An overview of internet of things: architecture, protocols and challenges. Inf Commun Technol Intell Syst 299–308
Azumah SW, Elsayed N, Adewopo V, Zaghloul ZS, Li C (2021) A deep LSTM based approach for intrusion detection IoT devices network in smart home. In: 2021 IEEE 7th world forum on internet of things (WF-IoT), pp 836–841. IEEE
Bisong E (2019) Building machine learning and deep learning models on Google cloud platform: a comprehensive guide for beginners. Apress
Carnero A, Martín C, Torres DR, Garrido D, Díaz M, Rubio B (2021) Managing and deploying distributed and deep neural models through Kafka-ML in the cloud-to-things continuum. IEEE Access 9:125478–125495
Chicco D, Jurman G (2020) The advantages of the Matthews correlation coefficient (MCC) over f1 score and accuracy in binary classification evaluation. BMC Genomics 21(1):1–13

Davis J, Goadrich M (2006) The relationship between precision-recall and ROC curves. In Proceedings of the 23rd international conference on Machine learning, pp 233–240
Diro A, Chilamkurti N (2018) Leveraging LSTM networks for attack detection in fog-to-things communications. IEEE Commun Mag 56(9):124–130
Panel Docs (2020) Man-in-the-middle attacks. https://docs.cpanel.net/knowledge-base/general-systems-administration/man-in-the-middle-attacks/. [Online]. Accessed 8 Oct 2021
Fang Y, Li Y, Liu L, Huang C (2018) Deepxss: cross site scripting detection based on deep learning. In: Proceedings of the 2018 international conference on computing and artificial intelligence, pp 47–51
Ghimire B, Rawat DB (2022) Recent advances on federated learning for cybersecurity and cybersecurity for federated learning for internet of things. IEEE Internet Things J
Grochowski E, Ronen R, Shen J, Wang H (2004) Best of both latency and throughput. In: IEEE international conference on computer design: VLSI in computers and processors, 2004. ICCD 2004. Proceedings, pp 236–243. IEEE
Gustavsson V (2019) Machine learning for a network-based intrusion detection system: an application using zeek and the cicids2017 dataset
Hertel L, Collado J, Sadowski P, Ott J, Baldi P (2020) Sherpa: robust hyperparameter optimization for machine learning. SoftwareX 12:100591
Idrissi I, Boukabous M, Azizi M, Moussaoui O, El Fadili H (2021) Toward a deep learning-based intrusion detection system for IoT against botnet attacks. IAES Int J Artif Intell 10(1):110
Kayode O (2020) A cloud based approach for data security in IoT
Khan AR, Kashif M, Jhaveri RH, Raut R, Saba T, Bahaj SA (2022) Deep learning for intrusion detection and security of internet of things (IoT): current analysis, challenges, and possible solutions. Secur Commun Netw 2022
Khattak HA, Shah MA, Khan S, Ali I, Imran M (2019) Perception layer security in internet of things. Future Gener Comput Syst 100:144–164
Lahasan B, Samma H (2022) Optimized deep autoencoder model for internet of things intruder detection. IEEE Access 10:8434–8448
Lata M, Kumar V (2022) IoT network security in smart homes. In: Cybersecurity in smart homes: architectures, solutions and technologies, pp 155–176
Lopez MA, Lobato AGP, Duarte OCMB, Pujolle G (2018) An evaluation of a virtual network function for real-time threat detection using stream processing. In: 2018 Fourth international conference on mobile and secure services (MobiSecServ), pp 1–5. IEEE
Ma W (2020) Analysis of anomaly detection method for internet of things based on deep learning. Trans Emerg Telecommun Technol 31(12):e3893
Malik R, Singh Y, Sheikh ZA, Anand P, Singh PK, Workneh TC (2022) An improved deep belief network IDS on IoT-based network for traffic systems. J Adv Transp 2022
Maniath S, Ashok A, Poornachandran P, Sujadevi VG, Prem Sankar AU, Jan S (2017) Deep learning LSTM based ransomware detection. In: 2017 Recent developments in control, automation & power engineering (RDCAPE), pp 442–446. IEEE
Martins I, Resende J, Sousa PR, Silva S, Antunes L, Gama J (2022) Host-based IDS: a review and open issues of an anomaly detection system in IoT. Future Gener Comput Syst
Mehedi ST, Anwar A, Rahman Z, Ahmed K, Rafiqul I (2022) Dependable intrusion detection system for IoT: a deep transfer learning-based approach. IEEE Trans Ind Inform
Miao J, Zhu W (2021) Precision–recall curve (PRC) classification trees. Evol Intell 15:1545–1569
Nair S (2019) Web application firewall (WAF) solutions. https://mobisoftinfotech.com/resources/wp-content/uploads/2018/05/AWS-WAF-Banner.png. [Online]. Accessed 21 March 2020
Otoum Y, Liu D, Nayak A (2022) Dl-IDS: a deep learning-based intrusion detection framework for securing IoT. Trans Emerg Telecommun Technol 33(3):e3803
Pan J, Ye N, Hanxiao Y, Hong T, Al-Rubaye S, Mumtaz S, Al-Dulaimi A, Chih-Lin I (2022) AI-driven blind signature classification for IoT connectivity: a deep learning approach. IEEE Trans Wirel Commun
Patan R, Rajasekhara Babu M (2018) A novel performance aware real-time data handling for big data platforms on Lambda architecture. Int J Comput Aided Eng Technol 10(4):418–430
Prabha PS, Kumar SM (2022) A novel cyber-attack leads prediction system using cascaded R2CNN model. Int J Adv Comput Sci Appl 13(2)
Roopak M, Tian GY, Chambers J (2019) Deep learning models for cyber security in IoT networks. In: 2019 IEEE 9th annual computing and communication workshop and conference (CCWC), pp 0452–0457. IEEE
Saha A, Subramanya A, Pirsiavash H (2020) Hidden trigger backdoor attacks. Proc AAAI Conf Artif Intell 34:11957–11965
Sarhan M, Layeghy S, Moustafa N, Portmann M (2020) Netflow datasets for machine learning-based network intrusion detection systems. In: Big data technologies and applications, pp 117–135. Springer
Sarker IH, Khan AI, Abushark YB, Alsolami F (2022) Internet of things (IoT) security intelligence: a comprehensive overview, machine learning solutions and research directions. Mob Netw Appl 1–17
Shahid WB, Baber A, Haider A, Khalid SB, Hammad A (2022) An enhanced deep learning based framework for web attacks detection, mitigation and attacker profiling. J Netw Comput Appl 198:103270
Siddique K, Akhtar Z, Khan MA, Jung Y-H, Kim Y (2018) Developing an intrusion detection framework for high-speed big data networks: a comprehensive approach. KSII Trans Internet Inf Syst 12:4021–4037. https://doi.org/10.3837/tiis.2018.08.026
Tangsatjatham P, Nupairoj N (2016) Hybrid big data architecture for high-speed log anomaly detection. In: 2016 13th International joint conference on computer science and software engineering (JCSSE), pp 1–6. IEEE
Visa S, Ramsay B, Ralescu AL, Van Der Knaap E (2011) Confusion matrix-based feature selection. MAICS 710(1):120–127
Yang F, Merlino G, Ray N, Léauté X, Gupta H, Eric T (2017) Open source lambda architecture for interactive analytics, The RADStack

Publisher's Note