Sharmila and Nagapadma *Cybersecurity (2023)6:41*
https://doi.org/10.1186/s42400-023-00178-5

Cybersecurity

RESEARCH

Open Access

Quantized autoencoder (QAE) intrusion detection system for anomaly detection in resource-constrained IoT devices using RT-IoT2022 dataset

B S Sharmila[1*†] and Rohini Nagapadma[1†]

Abstract

In recent years, many researchers focused on unsupervised learning for network anomaly detection in edge devices to identify attacks. The deployment of the unsupervised autoencoder model is computationally expensive in resource-constrained edge devices. This study proposes quantized autoencoder (QAE) model for intrusion detection systems to detect anomalies. QAE is an optimization model derived from autoencoders that incorporate pruning, clustering, and integer quantization techniques. Quantized autoencoder uint8 (QAE-u8) and quantized autoencoder float16 (QAE-f16) are two variants of QAE built to deploy computationally expensive AI models into Edge devices. First, we have generated a Real-Time Internet of Things 2022 dataset for normal and attack traffic. The autoencoder model operates on normal traffic during the training phase. The same model is then used to reconstruct anomaly traffic under the assumption that the reconstruction error (RE) of the anomaly will be high, which helps to identify the attacks. Furthermore, we study the performance of the autoencoders, QAE-u8, and QAE-f16 using accuracy, precision, recall, and F1 score through an extensive experimental study. We showed that QAE-u8 outperforms all other models with a reduction of 70.01% in average memory utilization, 92.23% in memory size compression, and 27.94% in peak CPU utilization. Thus, the proposed QAE-u8 model is more suitable for deployment on resource-constrained IoT edge devices.

Introduction

The adoption of IoT technology in the fields of healthcare, manufacturing, and agriculture requires constant network connectivity and data sharing. Because of this, cybercriminals could easily target IoT devices for exploitation and exploit any other devices sharing the same network infrastructure (Sobin 2020). According to the IBM data breach report for 2022, the average cost of a data breach reached a record-breaking USD 4.35 million, demonstrating a 2.6% rise from the previous year and a notable 12.7% surge since the 2020 report (Mansfield-Devine 2022). The Verkada breach in 2021 exposed the live feeds of 150 million surveillance cameras (Higgins 2022). In the same year, the attacker gained access to a water treatment organization in Florida and changed the chemical composition to contaminate the water. This kind of catastrophic data breach due to anomaly attacks in the IoT infrastructure is increasing significantly (Radanliev et al. 2018).

Our study aimed to develop a method for identifying anomalous attacks in IoT network traffic. Since spotting an anomaly requires careful observation of various IoT network traffic. In addition, the network traffic of each

†B. S. Sharmila and Rohini Nagapadma contributed equally to this work

*Correspondence:
B S Sharmila
sharmilabs@nie.ac.in
[1] Depatment of Electronics and Communication Engineering, The National Institute of Engineering, Mysore, Karnataka 570008, India

IoT device varies. So we utilize the autoencoder algorithm for anomaly detection. The training of the model particularly utilized benign network traffic, expecting that any anomalous traffic would lead to substantial RE. Therefore, in this research work, we considered the real-time IoT network traffic of four major devices: ThingSpeak-LED, MQTT-Temp, Amazon-Alexa, and Wipro-Bulb.

The distributed denial of service (DDoS) attack is one of the most critical threats to Internet of Things devices (Hummel Richard 2021). The vulnerability and pervasiveness of non-legacy IoT devices such as webcams, baby monitoring devices, and printers are primary targets for launching this volumetric DDoS attack to form a botnet (Salim et al. 2020). These compromised IoT devices redirect a large amount of traffic to the servers, which causes them to malfunction. According to the Kaspersky report, there was a tremendous increase in the proportion of intelligent DDoS attacks in 2022. The length of a DDoS attack increased 100-fold, reaching 3000 min. Over the last 4 years, the ratio of DDoS attacks has increased to about 50%. After the recent cryptocurrency market crash, experts predict an increase in DDoS attacks (Gutnikov 2022).

The Secure Shell (SSH) brute-force attack is another prominent cyberattack that uses trial and error to try all possible combinations to break the password (Fahrnberger 2022). Most IoT devices provide remote access over the SSH protocol with default passwords. The attackers utilize SSH brute-force attacks with known credentials to gain access and exploit IoT devices. In addition, brute-force attacks on SSH surpassed Telnet attacks by threefold. In 2020, Linux-based IoT devices were infected with malware named Kaiji Botnet and launched DDoS attacks via SSH brute-force attacks (Cimpanu 2020). In the middle of June 2022, a new IoT botnet virus known as RapperBot aggressively expanded its powers (Lakshmanan 2022). According to a report by Fortinet FortiGuard Labs, this particular category of IoT malware demonstrates a significant reliance on the original Mirai source code. However, what distinguishes it from other IoT malware families is its capability to perform credential brute-forcing and exploit SSH servers instead of the Telnet approach employed by the original Mirai attack.

IoT devices are vulnerable to SSH brute-force attacks because of their default credentials, which in turn cause DDoS exploitation, and these two attacks are the gateway for further exploitation. Hence, this study focuses on the two most significant attacks on IoT devices.

In recent years, IDS frameworks for IoT devices have integrated unsupervised learning techniques and other advances. In unsupervised learning, anomaly detection is a popular technique utilized for the identification of rare observations that deviate from normal events. The potential for significant environmental damage resulting from such anomalies establishes it as a crucial issue within the cyber domain. Several studies proposed different unsupervised learning techniques through various network traffic datasets to validate their IDS framework (Khraisat and Alazab 2021). However, the availability of IoT traces in most of the datasets is limited (Ring et al. 2019). Hence, it is difficult to identify an anomaly in IoT infrastructure. In this proposed work, we have generated network traces using real-time IoT devices for normal and attack patterns.

Real-time IoT devices, in contrast to general-purpose GPUs and CPUs, are designed for applications with limited physical resources. The deployment of an AI-based IDS framework in these devices imposes computational challenges such as restricted memory, fewer ALUs, high CPU time, etc. In such conditions, direct implementation of memory-consuming AI algorithms is unreliable without optimization (Lee et al. 2022; Garifulla et al. 2021). A solution to this problem is optimizing AI algorithms for IoT devices. Neural networks in AI optimization include network pruning and integer quantization (Shomron et al. 2021; Liang et al. 2021). The network pruning process includes removing redundant neurons that do not significantly affect model accuracy. This helps to decrease memory size and leads to a reduction in energy consumption (Hoefler et al. 2021). In addition, we reduced the numerical weights and activation function by converting commonly used floating-point 32-bit representation to floating-point 16-bit and 8-bit integer precision. This process represents a post-quantization technique (Finotti and Albertini 2021). The operations on integers reduce the overhead compared to floating point operations and reduce computational time and complexity.

Existing work

This section discusses different benchmark network traffic datasets developed for IDS and optimization strategies implemented to deploy Artificial Intelligence (AI) algorithms for edge devices.

In AI, the critical analysis and evaluation of the IDS framework depend on the type of dataset selected during the training and testing phases. The DARPA 1999 dataset that the MIT Lincoln Laboratory created is the most popular dataset for network IDS (McHugh 2000). DARPA contained 41 features in a packet-based format. The top attacks include DoS, port scanning, R2L, and U2R. Even though the distribution is high, DARPA suffers from high redundancy (Tavallaee et al. 2009). Sharafaldin et al. (2018) proposed the CICIDS2017 dataset. This dataset has 80 attributes and bidirectional flow-based network traces. The dataset includes botnet, DoS,

SSH brute-force, infiltration, and web attack traffic. Both servers and personal computers generated these attacks.

Cybersecurity researchers at the University of New South Wales Canberra developed the UNSW-NB15 dataset in 2015 (Moustafa and Slay 2015). Nour et al. used the IXIA Perfect Storm tool to produce this UNSW-NB15 dataset. They have extracted normal and attack traffic from the simulation tool. This dataset contains both packet-based and time-based features. The attack family includes DDoS, DoS, reconnaissance, and theft. In addition, UNSW proposed a new BoT-IoT dataset of 72 million traces in 2019 (Koroniotis et al. 2019). In this data collection, Node-RED, a simulation tool for middleware, was used to simulate Internet of Things (IoT) traffic. The authors developed JavaScript for virtual weather stations using IoT sensors like pressure, humidity, and temperature and communicated through the Message Queuing Telemetry Transport (MQTT) protocol. The prime attacks in this dataset include DoS and information theft. Sebastian Garcia and Erquiaga (2020) captured the IoT-23 dataset in the stratosphere lab at CTU University. The dataset has 23 captures of different networks generated in Raspberry Pi devices. IoT-23 comprised benign traffic collected from Amazon Echo, the Philips HUE LED Light and the Somfy Smart door lock devices. In this dataset, benign and malicious traffic was captured on different devices, leading to separate networks.

Table 1 summarizes the publicly available benchmark datasets for researchers. The survey shows that most of the examined datasets are not part of the IoT infrastructure. Even though IoT-23 utilized a real-time environment, the attack traces and benign traces generated were in a different environment. In addition, in anomaly-based IDS, we need to analyze the entire behavior of all IoT devices to identify the novel anomaly in the network. Therefore, in this research work, we have captured normal and attack traffic on the same IoT-based network infrastructure.

Thudumu et al. (2020) proposed a technique by deriving a generalized locally relevant subspace from a high dimensional dataset using a correlation score. Yang et al. (2021) presented a multi-tiered hybrid IDS to secure intravehicle networks (IVN) and external vehicular systems. Their proposed learning model comprises a four-tiered network. They used tree-based ML models such as decision tree (DT), extra trees (ET), random forest, and extreme gradient boosting (XGBoost) for the detection of known attacks. Furthermore, developing an anomaly detection system integrates CL-k-means, Bayesian optimization with the Gaussian process (BO-GP), and biased classifiers. They focused on detecting intrusions on the Internet of Vehicles using two publicly available CIC-IDS2017 and CAN-intrusion datasets. High dimensionality refers to a dataset that has attributes that are more independent. In this scenario, it is hard to identify outliers due to the problems associated with the 'curse of dimensionality'. In another study, Dutt et al. (2020) proposed a new statistical modeling based anomaly detection (SMAD) by exploiting the innate immune system for training and feature selection using the predefined value F in an ANOVA. This approach builds a naive activation function by adopting T-cells and B-cells.

Eskandari et al. (2020) proposed Passband, an intelligent end-to-end design for IDS for IoT gateways. This method utilized the Isolation Forest (IF) and Local Outlier Factor (LOF) algorithms for anomaly detection. This method was tested against port scanning, SYN flood, HTTP, and SSH brute-force attacks. With an F1 score of 0.79% for SYN flood, this article shows that Isolation Forest (IF) is more stable than the LOF technique. Nevertheless, the model has a high CPU utilization of 47.17%. Shyla et al. (2022) proposed the Nesterov-accelerated adaptive moment estimation-stochastic gradient descent (HNADAM-SGD) algorithm. This work utilized the UNSW-NB15 dataset for validation. The regression model selected to build the proposed algorithm involves different hyperparameters for optimization. For further investigation, this work compared the Ridge classifier, Logistic Regression, and an ensemble method with respect to recorded time complexity and accuracy. In

Table 1 Dataset

Dataset	Features type	No of features	IoT attack traces	IoT device	IoT normal traces	Year
DARPA 1998	Packet based	41	No	No	No	1998
CICIDS 2017	Bi-directional flow based	80	No	No	No	2017
UNSW-NB15	Packet based/flow based	49	No	No	No	2015
BoT-IoT	Packet based	35	Yes	Yes (Virtual)	Yes	2019
Aposemat IoT23	Packet based	–	Yes	Yes	Yes	2020
RT-IoT2022	Bi-directional flow based	24	Yes	Yes	Yes	2022

contrast, performance depends on the nature of the dataset and the parameters used during testing.

Ogundokun et al. (2021) proposed IDS using particle swarm optimization (PSO). This work applied particle swarm-based feature extraction techniques to optimize Machine learning (ML) algorithms. In this work, the author deployed PSO to the Decision Tree (PSO + DT) algorithm and the K-Nearest Neighbor (PSO + KNN) algorithm for the KDD-CUP99 dataset. The PSO + KNN algorithm performed better compared to other algorithms. Since the dataset does not contain IoT traces, deployment on IoT devices is inappropriate. Tang et al. (2020) proposed the Stacked Attention Autoencoder (SAAE) IDS model. The authors inserted an attention mechanism layer in the middle of the encoder and a latent layer. This layer calculates the attention vector of all features to identify the contribution of each attribute. The proposed algorithm can achieve 98.12% in a simulated environment.

Popoola et al. (2020) proposed a hybrid deep learning technique for botnets by combining a Long Short autoencoder (LAE) with deep Bidirectional Long Short Term Memory (BLSTM) for detecting intrusions in security cameras. The classification of Mirai and BASHLITE IoT attacks is performed by evaluating and testing network traffic on various cameras, including Samsung SNH 1011N, XCS7-1.003-WHT, Simple Home XCS7-1.002-WHT, and Provision PT-838. They have achieved better performance in terms of accuracy and precision. However, the model experienced high losses for classifying TCP and SCAN attacks (Zhang et al. 2021). Predić et al. (2022) discussed the possibility of optimizing the deep neural network (DNN) algorithm. Different compression techniques such as pruning, clustering, and quantization were discussed in this work. For validating the compression capacity and correctness of ResNet18, this study presents the sparsity and cluster preserving quantization (PCQAT) technique.

The AS-IDS model performs both signature-based and anomaly-based detection using the NSL-KDD dataset. This framework includes a Deep Q-Learning framework in which the output layer utilizes Signal to Noise Ratio (SNR) and bandwidth for classification (Otoum and Nayak 2021). Saba et al. (2022) proposed the convolution neural network (CNN) algorithm for IDS for outlier detection. This model utilized NID dataset and the BoT-IoT dataset for validation. However, the CNN model is far too complicated for deployment with limited resources. Gong et al. (2020) employed the VecQ model for compressing DNN algorithms using MNIST, ImageNet, CIFAR, THUCNews text, and IMDB movie review datasets. Parameterizing the probability estimate method used in the quantization process allowed this model to achieve higher accuracy. Hu et al. (2021) proposed the One-shot pruning quantization (OPQ) compression technique for the DNN algorithm. This method addresses the problem of manual tuning by utilizing pretrained weight parameters for computing the compression allocation and sharing a common codebook for all channels at each layer to replace conventional channel-wise quantization. However, these research works do not address the computational complexity of AI algorithms on real-time IoT devices, which includes memory consumption, CPU load, and processing time.

Table 2 shows the different optimization approaches for IDS. In summary, the deployment of AI algorithms in attack detection will elevate the performance of the IDS (Lakhan et al. 2022; Verhelst and Moons 2017). Besides the enormous advantage of AI techniques, the deployment of IDS in IoT devices is complex. Because the IoT devices suffer from limited storage capacity, low processing ability, and low power (Thakkar and Chaudhari 2021; Imteaj et al. 2021). Hence, this research work presents two QAE models, such as QAE-u8 and QAE-f16, by exploiting the combination of sparsity-based pruning (Anwar et al. 2017; Zeng et al. 2019), clustering, and quantization (Fang et al. 2020) methods. The empirical analysis also includes Raspberry Pi devices in its investigation.

Paper contribution

In this research work, the major contribution is as follows:

(a) We generated RT-IoT2022 datasets for normal and attack network traffic using IoT infrastructure with the deployment of real-time IoT devices like ThingSpeak-LED, MQTT-Temp, Amazon Alexa, Wipro bulb, and Raspberry Pi.
(b) We proposed optimized QAE-u8 and QAE-f16 models for IDS to support resource-constrained IoT devices, aiming to reduce the complexity of AI.
(c) Regarding the evaluation of the proposed model, QAE-u8 achieved better performance than QAE-f16 and the autoencoder model in terms of F1 score.
(d) We measure the computational complexity of the proposed model in terms of execution time, CPU, and memory utilization for deployment in computationally constrained devices. The proposed QAE-u8 model significantly outperforms the QAE-f16 and benchmark autoencoder models. For this purpose, we simulated all three models on a Raspberry Pi device.

Table 2 Summary of optimization techniques of different anomaly detection models using AI

Methods/references	Learning model	Security threat	Optimization method	Applied to IDS	Dataset of cyber attacks	Remarks
SAAE-DNN (Tang et al. 2020)	Stacked autoencoder	DoS, Probe, R2L and U2R	Attention vectors	✓	NSL-KDD	Limited to simulation
Passban (Eskandari et al. 2020)	IF, LOF	Port scan, SYS flood, HTTP and SSH brute-force	×	×	×	High CPU 47.17%
VecQ (Gong et al. 2020)	DNN	×	Quantization	×	×	Not applied for IDS
Hybrid deep learning technique (Popoola et al. 2020)	LAE, BLSTM	Mirai, BASHLITE	×	×	N_BaIoT	High false alarm rate to correlate TCP and SCAN attacks
PSO + DT, PSO_KNN (Ogundokun et al. 2021)	DT KNN	DoS, Probe, R2L and U2R	Particle swarm	✓	KDDCUP99	Limited to simulation
AS-IDS (Otoum and Nayak 2021)	DNN	DoS, Probe, R2L and U2R	Q-Learning	✓	NSL-KDD	Limited to simulation
OPQ (Hu et al. 2021)	DNN	×	Pruning quantization	×	×	Not applied for IDS
HNADAM-SDG (Shyla et al. 2022)	Regression	DoS and information theft	Hyper parameters	✓	UNSW-NB15	Performance depend on hyperparameters and type of dataset
Deep learning models (Saba et al. 2022)	CNN	DoS, DDoS, reconnaissance	×	✓	NID and BoT-IoT	CNN is too complex for IoT devices
Quantized autoencoder (QAE-u8, QAE-f16)	Autoencoder	DDoS and SSH brute-force	Pruning, clustering and quantization	✓	RT-IoT2022	Proposed method

Proposed framework

In this section, we introduce our proposed QAE IDS Framework for anomaly detection in resource-constrained IoT devices, as shown in Fig. 1. The work contribution mainly consists of four stages that offer: (a) dataset generation, (b) feature engineering, (c) autoencoder framework, (d) post-training quantization.

Dataset generation

The generation of the dataset is one of the crucial parts of unsupervised learning. In this section, we propose the RT-IoT2022 dataset for the training and testing of QAE-based IDS. Figure 2 shows the testbed infrastructure for RT-IoT2022 dataset generation.

The infrastructure consists of two parts, namely IoT victim devices and IoT attacker devices, both connected through a router. We collect the network traffic through a router using Wireshark, which is an open-source monitoring tool for network traffic that helps extract traces and convert them into a PCAP file. Table 3 shows the list of devices, operating systems, and related configurations. In this dataset, we created four normal profiles and two attack profiles. The details are as follows.

ThingSpeak-LED

ThingSpeak is an open-source IoT cloud platform to visualize sensor data and control actuator data. In this work, we established an interface between the Intel Galileo Gen 2 board and an RGB LED module. Subsequently, the LED status is monitored using the ThingSpeak platform. The router recorded the data communication of the LED module interfaced with the IoT device.

MQTT-Temp

MQTT protocol is a publish/subscribe protocol that aims to support resource-constrained IoT devices for communicating low-bandwidth data. First, we established an interface between the Raspberry Pi device and the temperature sensor. Consequently, the Raspberry Pi device publishes the temperature value to the MQTT Mosquitto Broker using the Paho MQTT library over the internet. Using the Wireshark tool, we passively monitor and capture the backend traffic, resulting in the collection of a dataset that includes MQTT-Temp.

Wipro-Bulb

Recently, Wipro released the NS9400 9-Watt B22 Smart Bulb as an IoT Smart Home Integrated Solution. Mobile devices can remotely control these bulbs using the WiFi protocol. Our Wipro-Bulb dataset includes the complete Wipro-Bulb communication.

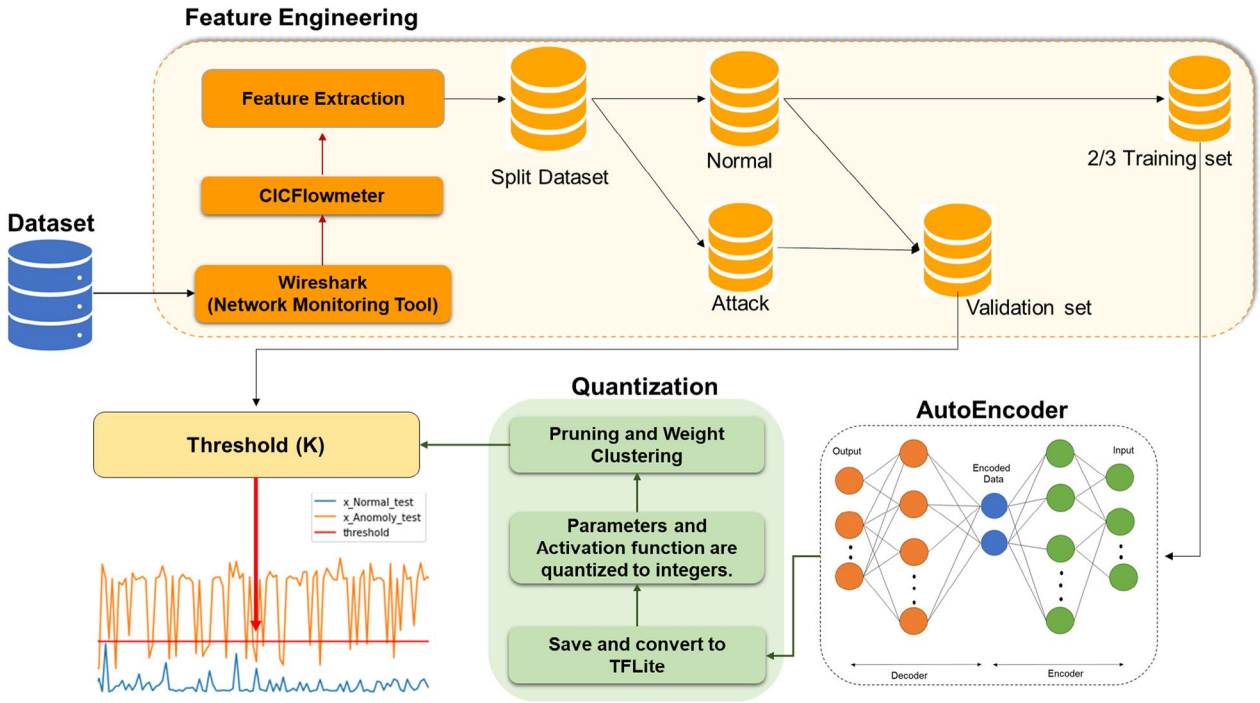


Fig. 1 Proposed QAE IDS framework for anomaly detection

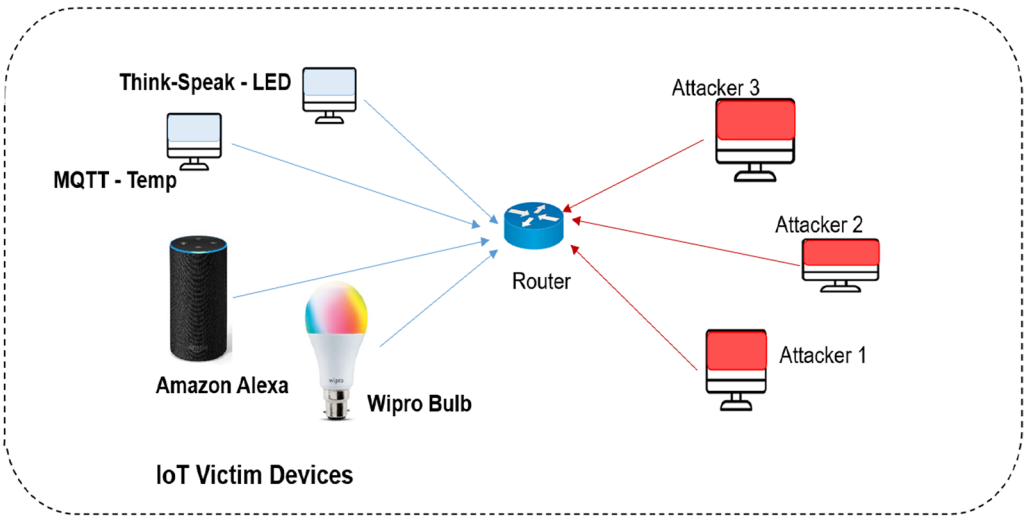


Fig. 2 RT-IoT2022 dataset infrastructure

Amazon Alexa

An Alexa device, developed by Amazon, is connected to a router and captured complete communications on the router. This device works as a cloud-based voice service (Barceló-Armada et al. 2022).

SSH brute-force attack

The authentication mechanism of IoT devices is prone to SSH brute-force attacks due to weak passwords. This attack not only discovers the passwords but also enters the system and implants malicious code to control and attack other connected devices. The SSH login auxiliary modules available in Metasploit, an open-source tool, are exploited to generate brute-force attack traces. The initial phase of the SSH brute-force process involves conducting a port scan using the Network Mapper (Nmap) tool to identify the machines having open SSH ports. Following this, we selected the 'Scanner SSH' auxiliary module of

Table 3 Victim-attacker device configurations

	Devices	Types	Operating system
Victim network	ThingSpeak LED	Intel Galileo Gen-2	Debian
	MQTT-Temp	Raspberry Pi	Rasbian OS
	Wipro Bulb	NS9400 9-Watt	–
	Amazon Alexa	ARM Cortex-A8	Fire OS
	Router	Raspberry Pi	Kali Linux
Attacker network	Attacker-1	Raspberry Pi	Kali Linux
	Attacker-2	Virtual machine	Kali Linux
	Attacker-3	Virtual machine	Kali Linux

msfconsole in Metasploit to initiate a brute-force attack. Further, the msfconsole is configured with the victim's IP address, default username, and password files to carry out the attack. Once the auxiliary module successfully cracks the login credentials, remote connectivity is established, allowing entry into the victim's computer.

DDoS attack

Further, we generate DDoS attack traces using the Hping3 tool from multiple Kali Linux IoT devices. Hping3 is an open-source tool available in Kali Linux OS utilized to launch DDoS attacks for victim machines. In the initial step, we established the hping3 command by specifying the IP address and domain name of the targeted victim. Subsequently, we opted for the 'SYN' scan feature provided by hping3. Additionally, we personalized the configuration by setting up source ports, destination ports, and fragmentation. Lastly, we transmitted 30,000 TCP SYN packets to the victim's device, utilizing random sources during the attack process. The ability of these attacks to change the IP address of source devices randomly every time poses difficulty to administrators in identifying the source of attacks (Jia et al. 2022). The procedure was extended for a period of 120 s, resulting in the generation of a significant volume of packets that overwhelmed the resources of the victim machine.

Feature Engineering

The collected PCAP files from Wireshark are converted and dumped as CSV files using the CICFlowmeter tool. Table 4 illustrates the complete RT-IoT2022 dataset. This tool generates a bidirectional flow of time-related features that help to distinguish DDoS attacks (Lashkari et al. 2017). To prevent over-fitting during training, features such as the address of the source, destination, and FlowID were eliminated (Aouini and Pekar 2022). In addition, numerical values were encoded for categorical features such as protocol and service. Then, we standardized the complete dataset by transforming it into a distribution with a mean value of zero and a standard deviation of one. The normal dataset had been labeled as zero and the attack dataset as one for anomaly detection. Finally, the training phase utilizes 70% of the dataset, and the validation phase utilizes 30%.

Autoencoder framework

This section discusses the construction of the autoencoder framework for training the normal network traffic behavior of IoT devices. Autoencoder is an unsupervised learning work on reconstruction methods. The mechanism of the autoencoder algorithm is to reconstruct the same input data into output data. In anomaly detection, the idea behind deploying the autoencoder is that anomalous traffic will fail to reconstruct its input traffic at the output. So, a significant error arises because the framework is trained exclusively on regular network traffic. The autoencoder framework consist of hidden layers, having encoder ϕ that converts the original dataset χ to latent space F for compression and a decoder θ reconstructs the original dataset from latent space as shown in Fig. 1. In addition, the hidden layers in the encoder had decreased, which helped extract only essential information and ignore the noise. This model consists of six hidden layers with three equal encoders and decoders, along with the 'relu' activation function. Since IoT devices are resource-constrained devices, the autoencoder model incorporates optimization procedures, which include pruning, clustering, and quantization. The following section describes the complete steps of optimization.

Table 4 RT-IOT2022 dataset

Dataset	PCAP	CICFlowmeter	Protocol	Service	SubCategory	Category
ThingSpeak-LED	10,526	8108	TCP. UDP	DNS, HTTP	ThingSpeak	0
MQTT-Temp	8162	4146	TCP	MQTT	MQTT	0
Amazon-Alexa	6056	5023	TCP, UDP, ICMP	DNS, HTTP	Alexa	0
Wipro-Bulb	1265	253	TCP, UDP	SSL, DNS, IRC	Wipro_Bulb	0
SSH brute-force	1564	526	TCP, UDP	DNS, SSH	SSH_Brute_Force	1
DDoS	1786	534	TCP	HTTP	DDoS	1

Algorithm 1 Proposed Quantized Autoencoder framework for anomaly detection

Input: Let training set $x_1, ...x_n \in \mathbb{R}$**, encoder** ϕ **and decoder** θ
Output: Find threshold

procedure Q-AUTOENCODER
  Repeat:
    Construct $\phi : \chi \rightarrow F$, $\theta : F \rightarrow \chi^{'}$
      Where, $\phi = h(W\chi + b)$ and $\theta = g(W^{'}\phi + b^{'})$
    Compute Loss $\mathcal{L} = \| \chi - \chi^{'} \|^2$

  Until $\underset{\phi \ \theta}{argmin} = \| \chi - (\phi \cdot \theta)\chi \|^2$ ▷ min RE is achieved
  Compute threshold by Eq. 2
  Pruning
  Compute $w' = \underset{w}{argmin} \| \chi - \chi' \|^2 + \lambda \sum_{i=0}^{k} | w_i |$ ▷ L1 Normalization
  if Weights are leading zero **then**
      Prune Weights
  Retrain the model
  Apply sparsity preserving clustering
  for `each layer` **do**
    Form group of weights into 8 clusters
    Update weights based on their clusters centroids
  end for
  Quantize parameters and the activation function to float16 and uint8
end procedure

Post-training quantization

The main goal of this research is to compress the model and reduce CPU and memory utilization as much as possible while retaining the model's accuracy. Therefore, we implemented the Post-Training Quantization technique to the trained autoencoder model for optimization. This technique includes pruning, clustering, and quantization. The steps involved in post-training quantization are as follows:

(a) The first step for model compression is pruning. Pruning means the selection of the most significant neurons while skipping redundant or zero-value tensors. The constant pruning involves: (1) The method of pruning: In this step, we considered retaining constant sparsity throughout training. (2) Fine tuning: Following this pruning process, the model undergoes retraining again.
(b) In the second step, we applied the weight clustering method. In this method, the layer weights are clustered and subsequently changed based on their cluster centroids. This approach helps in further reducing the size of the model. Based on a trade-off analysis between accuracy and the number of clusters, we selected eight clusters as the optimal choice for weight clustering in this research work.
(c) The third step involves converting the clustered model to TFlite format, aiming to improve the performance of the model in terms of CPU processing, model size, and memory utilization. This step, known as Quantization, involves converting all float32 weights and bias into unsigned 8-bit integer and floating-point 16-bit representations.

In the final stage, we trained and tested the quantized model before loading it onto the IoT device for the final predictions. The algorithm 1 shows the pseudo-code for Q-Autoencoder.

Experimental analysis

In this section, we first train the dataset for feature extraction. Then we demonstrate the autoencoder performance and compare it with other optimized autoencoders.

Feature extraction

First, we trained four different normal IoT datasets using an autoencoder: (a) ThingSpeak-LED, (b) MQTT-Temp, (c) Amazon Alexa, (d) Wipro-Bulb. Next, we applied the dataset to a trained model to reconstruct both normal and anomalous traffic. Then we calculated RE by employing Eq. 1 for Mean Absolute Error (MAE) and Eq. 2 for mean square error (MSE).

$$MAE_{RE} = \frac{1}{n}\sum_{i=1}^{n} \| \chi - \chi^{'} \|^2 \quad (1)$$

$$MSE_{RE} = \frac{1}{n}\sum_{i=1}^{n} \| \chi - \chi^{'} \| \quad (2)$$

where n represents the number of data points. According to the autoencoder model, if the model is trained only for normal traffic, it has to generate a high RE for anomalous traffic during the prediction. Therefore, we extracted features with high RE for all datasets, as shown in Table 5.

Subsequently, with the help of reconstruction error, we calculated the threshold to differentiate between normal and anomalous traffic using Eq. 3.

Table 5 Reconstruction error and threshold

Dataset	Features	Reconstruction error	Threshold
ThingSpeak-LED	bwd_pkts_payload.avg, flow_pkts_payload.avg, bwd_iat.avg, flow_iat.max, bwd_init_window_size	0.1	0.02
MQTT-Temp	id.orig_p, idle.avg, idle.std, fwd_last_window_size	0.4	0.06
Amazon-Alexa	id.orig_p, id.resp_p, down_up_ratio, fwd_header_size_max,flow_ECE_flag_count, fwd_pkts_payload.std,bwd_pkts_payload.std	0.9	0.06
Wipro-Bulb	flow_RST_flag_count, fwd_pkts_payload.avg, bwd_pkts_payload.max, bwd_pkts_payload.avg, bwd_pkts_payload.std,flow_pkts_payload.max, flow_pkts_payload.avg, flow_pkts_payload.std, flow_iat.std,payload_bytes_per_second, fwd_subflow_bytes,bwd_subflow_bytes	0.1	0.04
Final RT-IOT2022	All above features	–	0.02

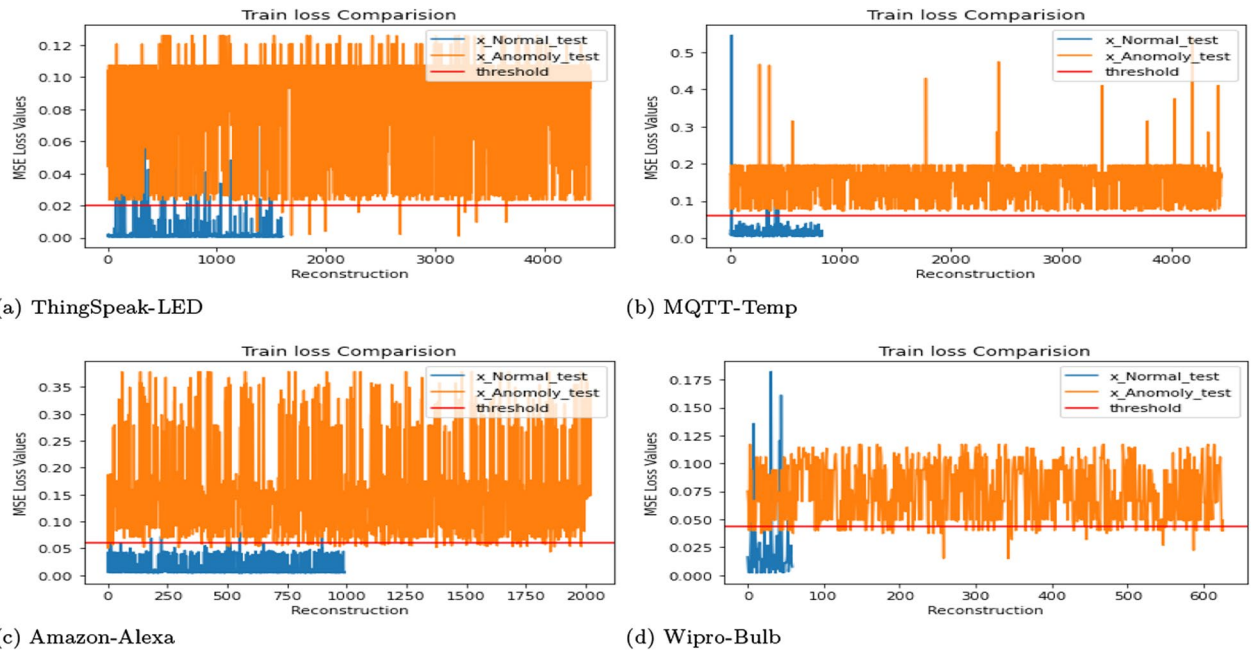


Fig. 3 Anomaly threshold measurement for various normal datasets

$$threshold = \mu(RE_{Normal}) + \sigma(RE_{Normal}) \quad (3)$$

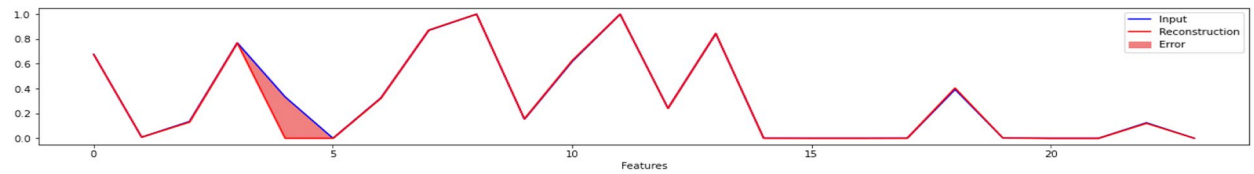
Where μ is mean and σ is the standard deviation, and RE_{Normal} is the RE of normal traffic. Figure 3 shows the graphical representation of discriminating between normal and anomalous traffic using a threshold value calculated from reconstruction error after decoding the normal dataset.

Hereafter, we combined all traces of normal network traffic from different IoT devices to form the final RT-IoT2022 dataset. This final dataset contains features with a high RE only in order to reduce computational complexity. Table 5 depicts the extracted features from each dataset to detect network anomalies.

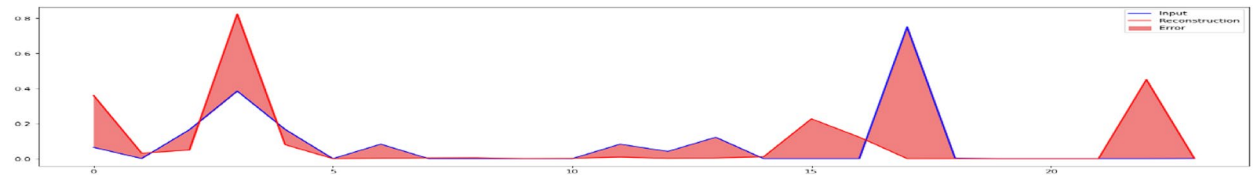
Next, we conducted training on RT-IoT2022 using an autoencoder and recalculated the RE for both normal and anomalous traffic. Figure 4a and b illustrate the correctness of RE by plotting against all extracted features. The figure shows that RE is large in anomaly traffic in all datasets compared to normal traffic. Finally, we calculated the threshold to detect anomalous traffic using Eq. 2. Figure 4c shows the error generation in normal and anomalous traffic and, finally, the threshold level to distinguish anomalies.

Metrics for evaluation of the QAE model

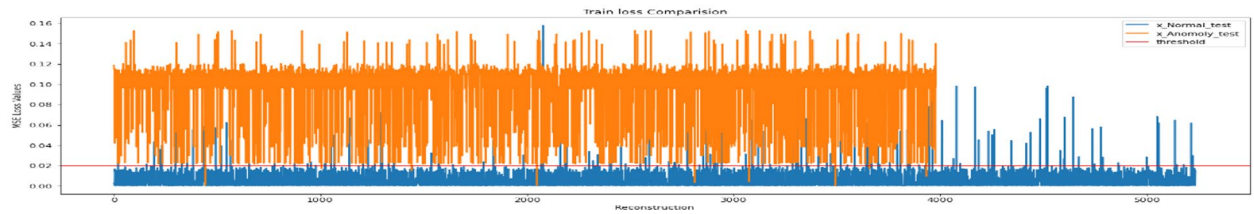
In this section, we performed an experimental evaluation by training an autoencoder model with normal traffic to detect anomalies. The anomalous traffic considered in this research work is the SSH brute-force attack and the DDoS attack. First, we considered individual normal traffic to extract the features for detecting anomalies. Then, we aggregated all the features for the proposed RT-IoT2022 dataset to evaluate performance. We investigated various evaluation metrics such as accuracy, precision, recall, and F1 score to validate the proposed training model. The accuracy measure measures the proportion of correctly predicted normal or anomalous traffic. Due to the recommendation against relying only on accuracy when there is an imbalance in the number of instances between classes, we also considered the precision metric. This metric measures the proportion of accurately predicted true positives. The sensitivity, or recall, of the model against the ground, or truth, provides a proportion of the predicted true positives against all true positives. Additionally, we also investigated the F1 score measure to enhance precision and recall. F1 score calculates the harmonic mean of precision and recall to provide more



(a) Reconstruction Error of RT-IoT2022 Normal dataset



(b) Reconstruction Error of RT-IoT2022 Anomaly dataset



(c) Threshold of RT-IoT2022 dataset

Fig. 4 Reconstruction error and threshold of RT-IoT2022

Table 6 Autoencoder evaluation metrics

Dataset	Accuracy (%)	Precision (%)	Recall (%)	F1 score (%)
ThingSpeak-LED	98.98	99.10	98.98	99.10
MQTT-Temp	99.63	100	99.9	99.98
Amazon-Alexa	98.33	98.33	98.34	98.34
Wipro-Bulb	91.5	91.5	91.5	95.00
Final RT-IOT2022	98.40	98.40	98.59	99.19

insight into the results in an imbalanced dataset. Table 6 reports all the numerical results.

Furthermore, we evaluated and compared the performance of two quantized autoencoder models, namely QAE-f16 and QAE-u8, with the non-optimized autoencoder model. During the training and testing phases, we considered only the extracted features. Table 7 shows scores for evaluation metrics and the calculation of different thresholds for each method. In the MAE error metric, the accuracy of the QAE-u8 is reduced marginally compared to other models. However, the F1 score of the QAE-u8 shows that the model is better than other proposed models.

Results and discussions

In this section, we test our models on state-of-the-art Raspberry Pi devices. The configuration of this resource-constrained IoT device is 2 GB of RAM and a 1.2 GHz quad-core ARM Cortex-A53 64-bit processor. The AI packages involved in this research work are TensorFlow-1.2 and the tflite supporting package. On Raspberry Pi device, we tested ground truth-values of time consumption, CPU utilization, and memory utilization for autoencoders, QAE-f16 and QAE-u8. Table 8 shows the complete analysis of all three proposed models. The summary of the results pertaining to all three proposed models is:

- The experimental results demonstrate a notable reduction in the memory size of the QAE-u8 model, reaching 92.23% and 26.29% in comparison to the autoencoder and QAE-f16, respectively, as shown in Table 8.
- QAE-u8 model compressed the average memory utilization to 70.01% and 10.45% with respect to autoencoder and QAE-f16 models, as shown in Table 8.
- 27.94% of the reduction is recorded for the CPU peak utilization parameter.
- We recorded the time consumption for predicting a single traffic instance. The observations in Table 8 show that the QAE-u8 consumes less time when compared to other models.

Figures 5 and 6 show memory consumption and CPU utilization on resource-constrained devices. We also reported the peak memory utilization and CPU peak utilization of Raspberry Pi devices in Table 8. The results show that the QAE-u8 autoencoder model consumes less memory and CPU. Therefore, from the above discussion, we can prove that QAE-u8 outperforms in

Table 7 Experimental results of autoencoder with and without optimization

Methods	Error metric	Threshold	Accuracy (%)	Precision (%)	Recall (%)	F1 score (%)
Autoencoder	MSE	0.006	91.04	91.04	91.05	95.1
	MAE	0.02	98.40	98.39	98.40	98.39
QAE-f16	MSE	0.073	78.07	78.07	78.69	86.10
	MAE	0.145	97.25	97.24	97.25	97.24
QAE-u8	MSE	6153.01	92.08	92.10	92.08	95.10
	MAE	35.02	96.35	96.35	96.36	98.10

Table 8 Performance evaluation of autoenocder, QAE-f16 and QAE-u8 models on Raspberry Pi IoT device

Methods	Memory size (bytes)	Average memory utilization (MiB)	Peak memory utilization(MiB)	Average CPU utilization	Peak CPU utilization	Time consumption (s)
Autoencoder	79,432	154.23	219.19	115.87	184	34
QAE-f16	8368	51.64	63.84	107.31	201	24
QAE-u8	6168	46.24	57.06	111.51	147	15

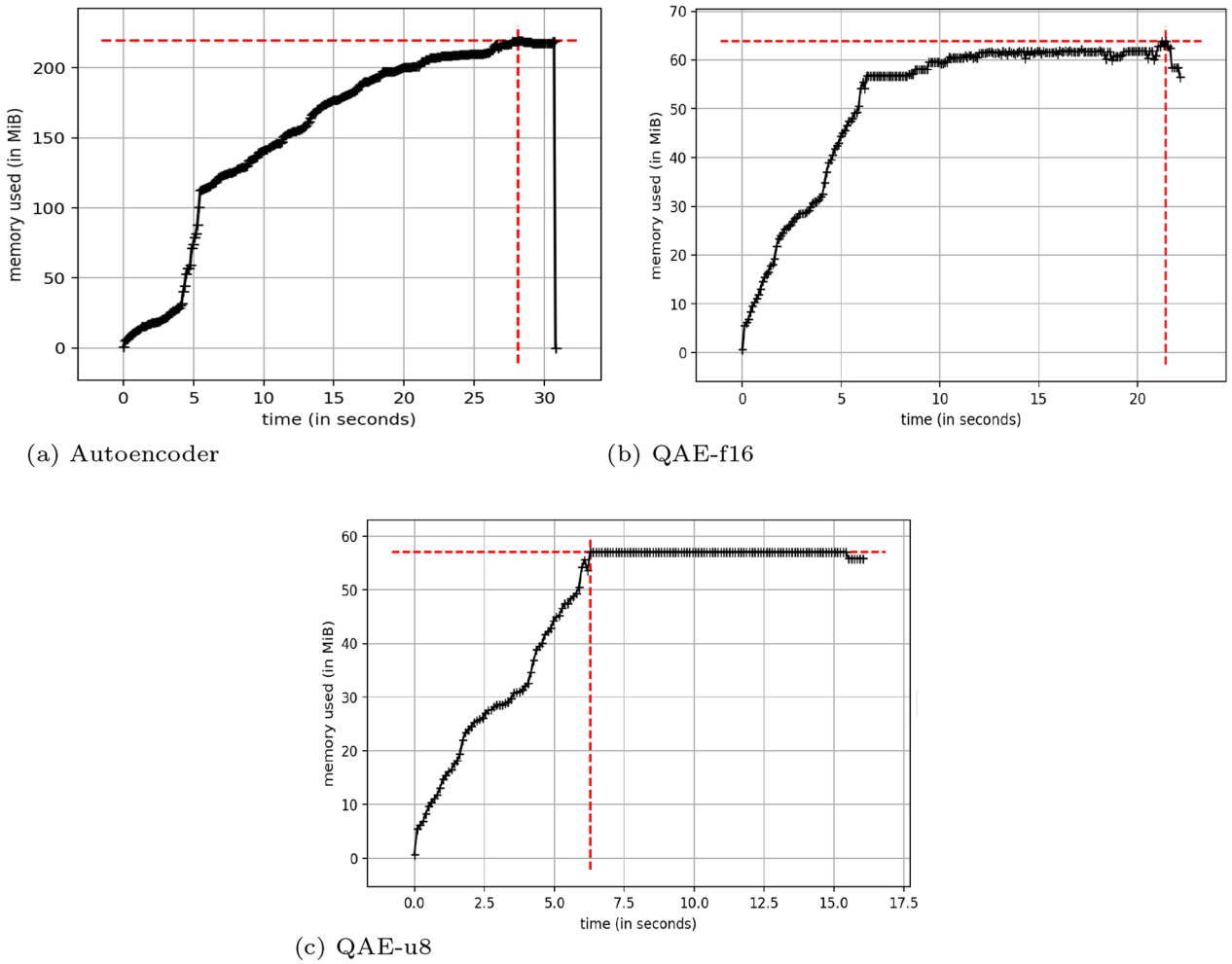
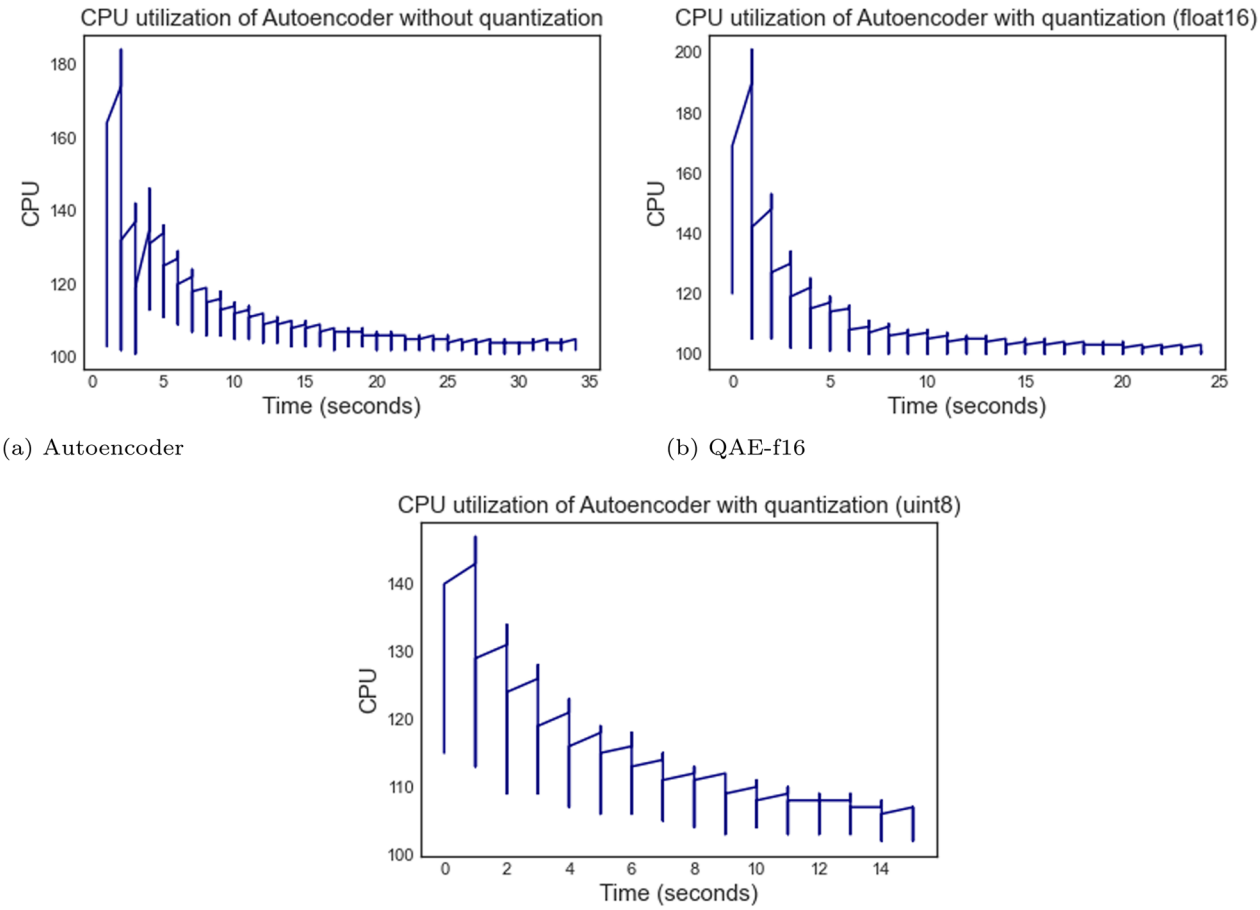


Fig. 5 Memory utilization of autoencoder, QAE-f16, QAE-u8 models

resource-constrained environments. Figure 7 shows the plots of numerical results. Further, the proposed QAE model not only provides performance benefits but also increases power efficiency by reducing storage memory costs and increasing computational efficiency. This is essentially beneficial for the widespread deployment of IDS based on QAE-u8 in agriculture and healthcare IoT devices since these devices require computationally inexpensive AI technologies.

Conclusion and future work

In this research, we generated the RT-IoT2022 dataset using an IoT environment detecting anomalies using an autoencoder framework. The dataset includes Amazon Alexa, ThingSpeak-LED, MQTT-Temp, Wipro-Bulb, SSH brute-force, and DDoS network traffic. The autoencoder model calculates RE and thresholds for detecting anomalies in IoT infrastructure. However, the major drawback of constructing autoencoder-based IDS in IoT devices is that they are resource-constrained, specifically for memory, processing ability, and power. Therefore, in this research, we proposed two optimized autoencoder models, the QAE-u8 and QAE-f16, for constructing the IDS framework. The optimization of the autoencoder models involves pruning, clustering, and quantization techniques. We conducted the final model prediction on a Raspberry Pi device, where the observation of QAE-u8 shows that the memory size has compressed to 92.23% and memory utilization reduced to 70.01%. The CPU peak utilization is reduced to 27.94%. In addition, there is a significant decrease in the time consumed for predictions, from 35 to 15s. The results indicate that our proposed QAE-u8 model can outperform the original autoencoder model in the context of reduced memory size, CPU, and memory utilization. Hence, this proves to be suitable for resource-constrained IoT devices. We also



(a) Autoencoder (b) QAE-f16

(c) QAE-u8

Fig. 6 CPU utilization of autoencoder, QAE-f16, QAE-u8 models

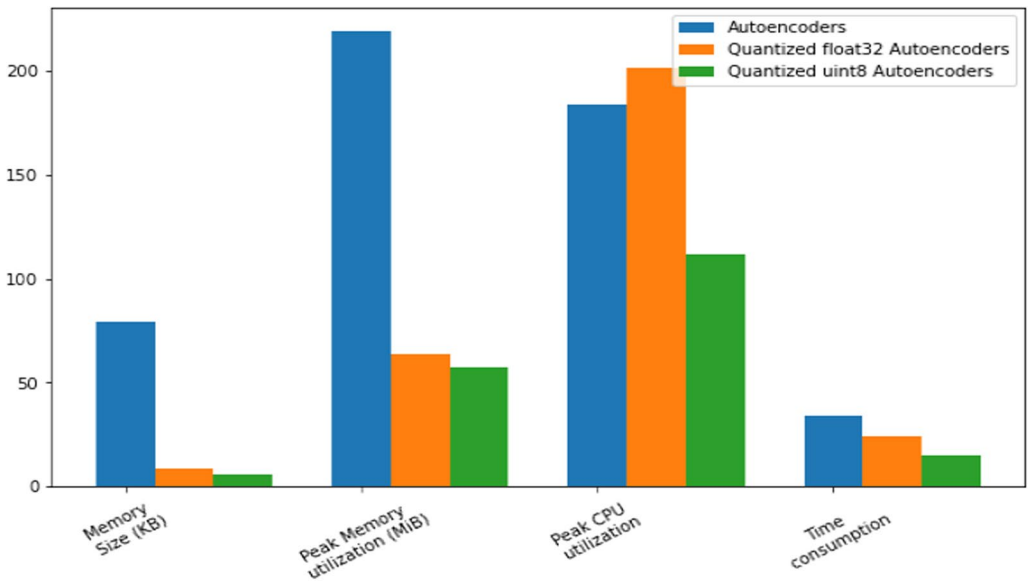


Fig. 7 CPU utilization of autoencoder, QAE-f16 and QAE-u8 models

investigated accuracy, precision, recall, and F1 score for all three models. The results depict that QAE-u8 with MAE achieved promising performance in evaluation metrics over the QAE-f16 model but slightly lower performance when compared to the baseline autoencoder model. Therefore, we conclude that there is a trade-off between the autoencoder and the QAE-u8 model in the context of accuracy and processor evaluation parameters like memory and CPU. For future studies, we will focus on other vulnerabilities of IoT devices to develop a more secure IoT infrastructure.

Authors' contributions
All the authors read and approved the final manuscript.

Declarations

Competing interests
The authors declare that they have no competing interests.

Received: 28 November 2022 Accepted: 5 July 2023
Published online: 05 September 2023

References

Anwar S, Hwang K, Sung W (2017) Structured pruning of deep convolutional neural networks. ACM J Emerg Technol Comput Syst (JETC) 13(3):1–18
Aouini Z, Pekar A (2022) Nfstream: a flexible network data analysis framework. Comput Netw 204:108719
Barceló-Armada R, Castell-Uroz I, Barlet-Ros P (2022) Amazon Alexa traffic traces. Comput Netw 205:108782
Cimpanu C (2020) New kaiji malware targets IoT devices via SSH bruteforce attacks zdnet.com. https://www.zdnet.com/article/new-kaiji-malware-targets-iot-devices-via-ssh-brute-force-attacks/. Accessed 13 Jun 2023
Dutt I, Borah S, Maitra IK (2020) Immune system based intrusion detection system (IS-IDS): a proposed model. IEEE Access 8:34929–34941
Eskandari M, Janjua ZH, Vecchio M et al (2020) Passban IDS: an intelligent anomaly-based intrusion detection system for IoT edge devices. IEEE Internet Things J 7(8):6882–6897
Fahrnberger G (2022) Realtime risk monitoring of SSH brute force attacks. In: Innovations for community services: 22nd international conference, I4CS 2022, Delft, The Netherlands, June 13–15, 2022, Proceedings. Springer, pp 75–95
Fang X, Liu H, Xie G et al (2020) Deep neural network compression method based on product quantization. In: 2020 39th Chinese control conference (CCC). IEEE, pp 7035–7040
Finotti V, Albertini B (2021) Simulating quantized inference on convolutional neural networks. Comput Electr Eng 95:107446
Garifulla M, Shin J, Kim C et al (2021) A case study of quantizing convolutional neural networks for fast disease diagnosis on portable medical devices. Sensors 22(1):219
Gong C, Chen Y, Lu Y et al (2020) VecQ: minimal loss DNN model compression with vectorized weight quantization. IEEE Trans Comput 70(5):696–710
Gutnikov A (2022) Crypto-collapse and rising smart attacks: Kaspersky reports on DDoS in Q2. https://www.kaspersky.com/about/press-releases/2022_crypto-collapse-and-rising-smart-attacks-kaspersky-reports-on-ddos-in-q2. Accessed on 13 Jun 2023
Higgins D (2022) Cyber attacks from 2021 we need to talk about. https://technative.io/cyber-attacks-from-2021-which-we-need-to-talk-about/. Accessed 14 May 2023
Hoefler T, Alistarh D, Ben-Nun T et al (2021) Sparsity in deep learning: pruning and growth for efficient inference and training in neural networks. J Mach Learn Res 22(241):1–124
Hu P, Peng X, Zhu H et al (2021) Opq: compressing deep neural networks with one-shot pruning-quantization. In: Proceedings of the AAAI conference on artificial intelligence, pp 7780–7788
Hummel Richard HC (2021) Crossing the 10 million mark: DDoS attacks in 2020. https://www.netscout.com/blog/asert/crossing-10-million-mark-ddos-attacks-2020. Accessed 13 Jun 2023
Imteaj A, Thakker U, Wang S et al (2021) A survey on federated learning for resource-constrained IoT devices. IEEE Internet Things J 9(1):1–24
Jia K, Liu C, Liu Q et al (2022) A lightweight DDoS detection scheme under SDN context. Cybersecurity 5(1):1–15
Khraisat A, Alazab A (2021) 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):1–27
Koroniotis N, Moustafa N, Sitnikova E et al (2019) Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-IoT dataset. Future Gener Comput Syst 100:779–796
Lakhan A, Mastoi QUA, Elhoseny M et al (2022) Deep neural network-based application partitioning and scheduling for hospitals and medical enterprises using IoT assisted mobile fog cloud. Enterp Inf Syst 16(7):1883122
Lakshmanan R (2022) New IoT Rapperbot malware targeting Linux servers via SSH brute-forcing attack. https://thehackernews.com/2022/08/new-iot-rapperbot-malware-targeting.html. Accessed 13 Jun 2023
Lashkari AH, Draper-Gil G, Mamun MSI et al (2017) Characterization of tor traffic using time based features. In: ICISSp, pp 253–262
Lee J, Yu M, Kwon Y et al (2022) Quantune: post-training quantization of convolutional neural networks using extreme gradient boosting for fast deployment. Future Gener Comput Syst 132:124–135
Liang T, Glossner J, Wang L et al (2021) Pruning and quantization for deep neural network acceleration: a survey. Neurocomputing 461:370–403
Mansfield-Devine S (2022) IBM: cost of a data breach. https://www.tripwire.com/state-of-security/key-points-ibm-cost-data-breach-report. Accessed 14 May 2023
McHugh J (2000) Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. ACM Trans Inf Syst Secur (TISSEC) 3(4):262–294
Moustafa N, Slay J (2015) 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). IEEE, pp 1–6
Ogundokun RO, Awotunde JB, Sadiku P et al (2021) An enhanced intrusion detection system using particle swarm optimization feature extraction technique. Procedia Comput Sci 193:504–512
Otoum Y, Nayak A (2021) As-ids: anomaly and signature based ids for the internet of things. J Netw Syst Manag 29:1–26
Popoola SI, Adebisi B, Hammoudeh M et al (2020) Hybrid deep learning for botnet attack detection in the internet of things networks. IEEE IoT J. https://doi.org/10.1109/JIOT.2020.3034156
Predić B, Vukić U, Saračević M et al (2022) The possibility of combining and implementing deep neural network compression methods. Axioms 11(5):229
Radanliev P, De Roure D, Cannady S et al (2018) Economic impact of IoT cyber risk-analysing past and present to predict the future developments in IoT risk analysis and IoT cyber insurance. In: Living in the internet of things: cybersecurity of the IoT—2018. https://doi.org/10.1049/cp.2018.0003
Ring M, Wunderlich S, Scheuring D et al (2019) A survey of network-based intrusion detection data sets. Comput Secur 86:147–167
Saba T, Rehman A, Sadad T et al (2022) Anomaly-based intrusion detection system for IoT networks through deep learning model. Comput Electr Eng 99:107810
Salim MM, Rathore S, Park JH (2020) Distributed denial of service attacks and its defenses in IoT: a survey. J Supercomput 76:5320–5363
Sebastian Garcia AP, Erquiaga MJ (2020) IoT-23 dataset: a labeled dataset of malware and benign IoT traffic (version 1.0.0). https://www.stratosphereips.org/datasets-iot23. Accessed 13 Jun 2023
Sharafaldin I, Lashkari AH, Ghorbani AA (2018) Toward generating a new intrusion detection dataset and intrusion traffic characterization. ICISSp 1:108–116
Shomron G, Gabbay F, Kurzum S et al (2021) Post-training sparsity-aware quantization. Adv Neural Inf Process Syst 34:17737–17748

Shyla S, Bhatnagar V, Bali V et al (2022) Optimization of intrusion detection systems determined by ameliorated HNADAM-SGD algorithm. Electronics 11(4):507
Sobin C (2020) A survey on architecture, protocols and challenges in IoT. Wirel Pers Commun 112(3):1383–1429
Tang C, Luktarhan N, Zhao Y (2020) SAAE-DNN: deep learning method on intrusion detection. Symmetry 12(10):1695
Tavallaee M, Bagheri E, Lu W et al (2009) A detailed analysis of the KDD CUP 99 data set. In: 2009 IEEE symposium on computational intelligence for security and defense applications. IEEE, pp 1–6
Thakkar A, Chaudhari K (2021) A comprehensive survey on deep neural networks for stock market: the need, challenges, and future directions. Expert Syst Appl 177:114800
Thudumu S, Branch P, Jin J et al (2020) Estimation of locally relevant subspace in high-dimensional data. In: Proceedings of the Australasian computer science week multiconference, pp 1–6
Verhelst M, Moons B (2017) Embedded deep neural network processing: algorithmic and processor techniques bring deep learning to IoT and edge devices. IEEE Solid State Circuits Mag 9(4):55–65
Yang L, Moubayed A, Shami A (2021) MTH-IDS: a multitiered hybrid intrusion detection system for internet of vehicles. IEEE Internet Things J 9(1):616–632
Zeng L, Chen S, Zeng S (2019) An efficient end-to-end channel level pruning method for deep neural networks compression. In: 2019 IEEE 10th international conference on software engineering and service science (ICSESS). IEEE, pp 43–46
Zhang C, Liu J, Chen W et al (2021) Unsupervised anomaly detection based on deep autoencoding and clustering. Secur Commun Netw. https://doi.org/10.1155/2021/7389943

Publisher's Note